# Package ‘mssm’

**Type** Package  
**Title** Multivariate State Space Models  
**Version** 0.1.3  
**Description** Provides methods to perform parameter estimation and make analysis of multivariate observed outcomes through time which depends on a latent state variable. All methods scale well in the dimension of the observed outcomes at each time point. The package contains an implementation of a Laplace approximation, particle filters like suggested by Lin, Zhang, Cheng, & Chen (2005) [doi:10.1198/016214505000000349], and the gradient and observed information matrix approximation suggested by Poyiadjis, Doucet, & Singh (2011) [doi:10.1093/biomet/asq062].

**License** GPL-2  
**Encoding** UTF-8  
**LazyData** true  
**Depends** R (>= 3.5.0), stats, graphics  
**LinkingTo** Rcpp, RcppArmadillo, testthat, nloptr (>= 1.2.0)  
**Imports** Rcpp, nloptr (>= 1.2.0)  
**RoxygenNote** 6.1.1  
**SystemRequirements** C++11  
**Suggests** testthat, microbenchmark, Ecdat  
**NeedsCompilation** yes  
**Author** Benjamin Christoffersen [cre, aut], Anthony Williams [cph]  
**Maintainer** Benjamin Christoffersen <boennecd@gmail.com>  
**Repository** CRAN  
**Date/Publication** 2019-11-07 00:20:02 UTC
R topics documented:

mssm-package ..........................................................  2
get_ess .................................................................  2
logLik.mssm ............................................................  3
mssm .................................................................  4
mssm-Laplace ..........................................................  6
mssm-pf ...............................................................  8
mssm-smoother ........................................................  9
mssm_control .......................................................... 10
plot.mssm ............................................................. 12
plot.mssmEss .......................................................... 13

Index 15

mssm-package  Multivariate State Space Models

Description

This package contains particle filter methods for multivariate observed outcomes and low dimensional state vectors. The methods are intended to scale well in the dimension of the observed outcomes. The main function in the package is the mssm function. The package also includes a method to estimate the parameters using a Laplace approximation.

The README contains an example of the features in the package. See https://github.com/boennecd/mssm.

The package is still under development and the API and results of the methods may change.

get_ess  Effective Sample Sizes of a mssm Object

Description

Extracts the effective sample size at each time point from a mssm object.

Usage

get_ess(object)

Arguments

object an object of class mssm.

Value

An object of class mssmEss with the effective sample sizes.
Examples

```r
if(require(Ecdat)){
  # load data and fit glm to get some parameters to use in an illustration
  data("Gasoline", package = "Ecdat")
  glm_fit <- glm(lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
                  Gamma("log"), Gasoline)

  # get object to run particle filter
  library(mssm)
  ll_func <- mssm(
      fixed = formula(glm_fit), random = "1", family = Gamma("log"),
      data = Gasoline, ti = year, control = mssm_control(
        N_part = 1000L, n_threads = 1L))

  # run particle filter
  pf <- ll_func$pf_filter(
      cfix = coef(glm_fit), disp = summary(glm_fit)$dispersion,
      F. = as.matrix(.0001), Q = as.matrix(.0001^2))

  # summary statistics for effective sample sizes
  print(ess <- get_ess(pf))
}
```

### logLik.mssm

**Approximate Log-likelihood for a mssm Object**

**Description**

Function to extract the log-likelihood from a mssm or mssmLaplace object.

**Usage**

```r
## S3 method for class 'mssm'
logLik(object, ...)

## S3 method for class 'mssmLaplace'
logLik(object, ...)
```

**Arguments**

- `object` an object of class mssm or mssmLaplace.
- `...` un-used.

**Value**

A logLik object. The log_lilk_terms attribute contains the log-likelihood contributions from each time point.

The degrees of freedom assumes that all parameters are free. The number of observations may be invalid for some models (e.g., discrete survival analysis).
Examples

```r
if(require(Ecdat)){
    # load data and fit glm to get starting values
    data("Gasoline", package = "Ecdat")
    glm_fit <- glm(lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
                Gamma("log"), Gasoline)

    # get object to perform estimation
    library(mssm)
    ll_func <- mssm(
        fixed = formula(glm_fit), random = ~ 1, family = Gamma("log"),
        data = Gasoline, ti = year, control = mssm_control(
            N_part = 1000L, n_threads = 1L))

    # fit model with time-varying intercept with Laplace approximation
    disp <- summary(glm_fit)$dispersion
    laplace <- ll_func$Laplace(
        cfix = coef(glm_fit), disp = disp, F. = diag(.5, 1), Q = diag(1))

    # run particle filter
    pf <- ll_func$pf_filter(
        cfix = laplace$cfix, disp = laplace$disp, F. = laplace$F., Q = laplace$Q)

    # compare approximate log-likelihoods
    print(logLik(pf))
    print(logLik(laplace))
}
```

---

mssm  
Get Multivariate State Space Model Functions

Description

Returns an object with a function that can be used to run a particle filter, a function to perform parameter estimation using a Laplace approximation, and a function to perform smoothing of particle weights.

Usage

```r
mssm(fixed, family, data, random, weights, offsets, ti,
     control = mssm_control())
```

Arguments

- `fixed` formula with outcome variable on the left hand side and covariates with fixed effects on the right hand side.
- `family` family for the observed outcome given the state variables and covariates.
- `data` data.frame or environment containing the variables in fixed and random.
random formula for covariates with a random effect. Left hand side is ignored.
weights optional prior weights.
offsets optional a priori known component in the linear predictor.
ti integer vector with time indices matching with each observation of fixed and random.
control list with arguments passed to mssm_control.

Value
An object of class mssmFunc with the following elements

- **pf_filter** function to perform particle filtering. See mssm-pf.
- **Laplace** function to perform parameter estimation with a Laplace approximation. See mssm-Laplace.
- **smoother** function to compute smoothing weights for an mssm object returned by the pf_filter function. See mssm-smoother.
- **terms_fixed** terms.object for the covariates with fixed effects.
- **terms_random** terms.object for the covariates with random effects.
- **y** vector with outcomes.
- **X** covariates with fixed effects.
- **Z** covariates with random effects.
- **ti** time indices for each observation.
- **weights** prior weights for each observation.
- **offsets** a priori known component in the linear predictor for each observation.
- **call** the matched call.
- **family** character describing the conditional distribution of the outcomes.

See Also
The README of the package contains examples of how to use this function. See https://github.com/boennecd/mssm.

Examples
if(require(Ecdat)){
  # load data and fit glm to get starting values
  . <- print
data("Gasoline", package = "Ecdat")
glm_fit <- glm(lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
            Gamma("log"), Gasoline)
  # get object to perform estimation
library(mssm)
ll_func <- mssm(
  fixed = formula(glm_fit), random = ~ 1, family = Gamma("log"),
  data = Gasoline, ti = year, control = mssm_control(}
Parameter Estimation with Laplace Approximation for Multivariate State Space Model

**Description**

Function returned from `mssm` which can be used to perform parameter estimation with a Laplace approximation.

**Arguments**

- `cfix`: starting values for coefficient for the fixed effects.
- `disp`: starting value for additional parameters for the family (e.g., a dispersion parameter).
- `F.`: starting values for matrix in the transition density of the state vector.
- `Q`: starting values for covariance matrix in the transition density of the state vector.
Q0 un-used.
mu0 un-used.
trace integer controlling whether information should be printed during parameter estimation. Zero yields no information.

Value
An object of class mssmLaplace with the following elements

- F. estimate of F.
- Q estimate of Q.
- cfix estimate of cfix.
- logLik approximate log-likelihood at estimates.
- n_it number of Laplace approximations.
- code returned code from nlopt.
- disp estimated dispersion parameter.

Remaining elements are the same as returned by mssm.

See Also
mssm.

Examples

```r
if(require(Ecdat)){
  # load data and fit glm to get starting values
data("Gasoline", package = "Ecdat")
glm_fit <- glm(lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
    Gamma("log"), Gasoline)

  # get object to perform estimation
library(mssm)
ll_func <- mssm(
  fixed = formula(glm_fit), random = ~ 1, family = Gamma("log"),
data = Gasoline, ti = year, control = mssm_control(
    N_part = 1000L, n_threads = 1L))

  # fit model with time-varying intercept with Laplace approximation
disp <- summary(glm_fit)$dispersion
laplace <- ll_func$Laplace(
  cfix = coef(glm_fit), disp = disp, F. = diag(.5, 1), Q = diag(1))
  print(laplace)
}
```
Particle Filter Function for Multivariate State Space Model

Description

Function returned from `mssm` which can be used to perform particle filtering given values for the parameters in the model.

Arguments

- `cfix` values for for coefficient for the fixed effects.
- `disp` additional parameters for the family (e.g., a dispersion parameter).
- `F` matrix in the transition density of the state vector.
- `Q` covariance matrix in the transition density of the state vector.
- `Q0` optional covariance matrix at the first time point. Default is the covariance matrix in the time invariant distribution.
- `mu0` optional mean at the first time point. Default is the zero vector.
- `trace` integer controlling whether information should be printed during particle filtering. Zero yields no information.
- `seed` integer to pass to `set.seed`. The seed is not set if the argument is `NULL`.
- `what, N_part` same as in `mssm_control`.

Value

An object of class `mssm` with the following elements

- `pf_output` A list with an element for each time period. Each element is a list with `particles`: the sampled particles, `stats`: additional object that is requested to be computed with each particle, `ws`: unnormalized log particle weights for the filtering distribution, and `ws_normalized`: normalized log particle weights for the filtering distribution.

Remaining elements are the same as returned by `mssm`.

If gradient approximation is requested then the first elements of `stats` are w.r.t. the fixed coefficients, the next elements are w.r.t. the matrix in the map from the previous state vector to the mean of the next, and the last element is w.r.t. the covariance matrix. Only the lower triangular matrix is kept for the covariance matrix. See the examples in the README at [https://github.com/boennecd/mssm](https://github.com/boennecd/mssm). There will be an additional element for the dispersion parameter if the family has a dispersion parameter.

If the Hessian is requested then the $\tilde{\beta}_i$’s in Poyiadjis et al. (2011) are returned after the gradient elements. These can be used to approximate the observed information matrix. That is, using that the approximation of the observed information matrix is

$$\tilde{S}_n \tilde{S}_n^\top - \sum_{i=1}^n \tilde{W}_n^{(i)} (\tilde{\alpha}_n^{(i)} \tilde{\alpha}_n^{(i)\top} + \tilde{\beta}_n^{(i)}) = \sum_{i=1}^n \tilde{W}_n^{(i)} \tilde{\alpha}_n^{(i)}$$

as in Poyiadjis et al. (2011). See the README for an example.
mssm-smoother

Computes Smoothed Particle Weights for Multivariate State Space Model

Description
Computes smoothed weights using the backward smoothing formula for a mssm object. The k-d dual tree approximation is also used if it used for the mssm object.

Arguments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>an object of class mssm from mssm-pf.</td>
</tr>
</tbody>
</table>

Value
Same as mssm-pf but where the pf_output’s list elements has an additional element called ws_normalized_smooth. This contains the normalized log smoothing weights.

References

See Also

mssm.

Examples

```r
if(require(Ecdat)){
  # load data and get object to perform particle filtering
data("Gasoline", package = "Ecdat")

library(mssm)
ll_func <- mssm(
  fixed = lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
  random = ~ 1, family = Gamma("log"), data = Gasoline, ti = year,
  control = mssm_control(N_part = 1000L, n_threads = 1L))

# run particle filter
cfix <- c(0.612, -0.015, 0.214, 0.048, -0.013, -0.016, -0.022, 0.047,
  -0.046, 0.007, -0.001, 0.008, -0.117, 0.075, 0.048, -0.054, 0.017,
  0.228, 0.077, -0.056, -0.139)
pf <- ll_func$pf_filter(
  cfix = cfix, Q = as.matrix(2.163e-05), F. = as.matrix(0.9792),
  disp = 0.000291)
print(pf)
}
```
See Also

mssm.

Examples

```r
if(require(Ecdat)){
  # load data and get object to perform particle filtering
data("Gasoline", package = "Ecdat")

library(mssm)
l1_func <- mssm(
  fixed = lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
  random = ~ 1, family = Gamma("log"), data = Gasoline, ti = year,
  control = mssm_control(N_part = 1000L, n_threads = 1L))

  # run particle filter
cfix <- c(0.612, -0.015, 0.214, 0.048, -0.013, -0.016, -0.022, 0.047,
    -0.046, 0.007, -0.001, 0.008, -0.117, 0.075, 0.048, -0.054, 0.017,
    0.228, 0.077, -0.056, -0.139)
pf <- ll_func$pf_filter(
  cfix = cfix, Q = as.matrix(2.163e-05), F. = as.matrix(0.9792),
  disp = 0.000291)

print(is.null(pf$pfilteroutput[[1L]]$ws_normalized_smooth))
pf <- ll_func$smoother(pf)
print(is.null(pf$pfilteroutput[[1L]]$ws_normalized_smooth))
}
```

---

**mssm_control**  
Auxiliary for Controlling Multivariate State Space Model Fitting

**Description**

Auxiliary function for mssm.

**Usage**

```r
mssm_control(N_part = 1000L, n_threads = 1L, covar_fac = 1.2,
              ftol_rel = 1e-06, nu = 8, what = "log_density",
              which_sampler = "mode_aprx", which_ll_cp = "no_aprx", seed = 1L,
              KD_N_max = 10L, aprx_eps = 0.001, ftol_abs = 1e-04,
              ftol_abs_inner = 1e-04, la_ftol_rel = -1, la_ftol_rel_inner = -1,
              maxeval = 10000L, maxeval_inner = 10000L, use_antithetic = FALSE)
```

**Arguments**

- `N_part` integer greater than zero for the number of particles to use.
- `n_threads` integer greater than zero for the number of threads to use.
covar_fac positive numeric scalar used to scale the covariance matrix in the proposal distribution.

ftol_rel positive numeric scalar with convergence threshold passed to nloptr if the mode approximation method is used for the proposal distribution.

nu degrees of freedom to use for the multivariate t-distribution that is used as the proposal distribution. A multivariate normal distribution is used if nu <= 2.

what character indicating what to approximate. "log_density" implies only the log-likelihood. "gradient" also yields a gradient approximation. "Hessian" also yields an approximation of the observed information matrix.

which_sampler character indicating what type of proposal distribution to use. "mode_aprx" yields a Taylor approximation at the mode. "bootstrap" yields a proposal distribution similar to the common bootstrap filter.

which_ll_cp character indicating what type of computation should be performed in each iteration of the particle filter. "no_aprx" yields no approximation. "KD" yields an approximation using a dual k-d tree method.

seed integer with seed to pass to set.seed.

KD_N_max integer greater than zero with the maximum number of particles to include in each leaf of the two k-d trees if the dual k-d trees method is used.

aprx_eps positive numeric scalar with the maximum error if the dual k-d tree method is used.

ftol_abs, ftol_abs_inner, la_ftol, la_ftol_inner, maxeval, maxeval_inner scalars passed to nlopt when estimating parameters with a Laplace approximation. The _inner denotes the values passed in the inner mode estimation. The mode estimation is done with a custom Newton–Raphson method.

use_antithetic logical which is true if antithetic variables should be used.

See Also

mssm.

See the README of the package for details of the dual k-d tree method at https://github.com/boennecdmssm.

Examples

library(mssm)
str(mssm_control())
str(mssm_control(N_part = 2000L))
plot.mssm

Plot Predicted State Variables for mssm Object.

Description

Plots the predicted mean and pointwise prediction interval of the state variables for the filtering distribution or smoothing distribution.

Usage

```
## S3 method for class 'mssm'
plot(x, y, qs = c(0.05, 0.95), do_plot = TRUE,
     which_weights = c("filter", "smooth"), ...)
```

Arguments

- `x`: an object of class mssm.
- `y`: un-used.
- `qs`: two-dimensional numeric vector with bounds of the prediction interval.
- `do_plot`: TRUE to create a plot with the mean and quantiles.
- `which_weights`: character of which weights to use. Either "filter" for filter weights or "smooth" for smooth weights. The latter requires that smooth element has been used.
- `...`: un-used.

Value

List with means and quantiles.

Examples

```r
if(require(Ecdat)){
  # load data and get object to perform particle filtering
  data("Gasoline", package = "Ecdat")

  library(mssm)
  ll_func <- mssm(
    fixed = lgaspcar ~ factor(country) + lincomep + lrpmg + lcarpcap,
    random = ~ 1, family = Gamma("log"), data = Gasoline, ti = year,
    control = mssm_control(N_part = 1000L, n_threads = 1L))

  # run particle filter
  cfix <- c(0.612, -0.015, 0.214, 0.048, -0.013, -0.016, -0.022, 0.047,
            -0.046, 0.007, -0.001, 0.008, -0.117, 0.075, 0.048, -0.054, 0.017,
            0.228, 0.077, -0.056, -0.139)
  pf <- ll_func$pf_filter(
    cfix = cfix, Q = as.matrix(2.163e-05), F. = as.matrix(0.9792),
```
## plot.mssmEss

### Plot Effective Sample Sizes

#### Description

Plots the effective sample sizes.

#### Usage

```r
## S3 method for class 'mssmEss'
plot(x, y, ...)
```

#### Arguments

- `x`: an object of class `mssmEss`.
- `y`: un-used.
- `...`: un-used.

#### Value

The plotted x-values, y-values, and maximum possible effective sample size.

#### Examples

```r
if(require(Ecdat)){
  # load data and fit glm to get some parameters to use in an illustration
  data("Gasoline", package = "Ecdat")
  glm_fit <- glm(lgaspcar ~ factor(country) + lincomep + lrpme + lcarpcap,
    Gamma("log"), Gasoline)

  # get object to run particle filter
  library(mssm)
  ll_func <- mssm(  
    fixed = formula(glm_fit),  
    random = -1,  
    family = Gamma("log"),  
    data = Gasoline,  
    ti = year,  
    control = mssm_control(  
      N_part = 1000L,  
      n_threads = 1L)  
  )

  # run particle filter
  pf <- ll_func$pf_filter(    
    cfix = coef(glm_fit),  
    disp = summary(glm_fit)$dispersion,
```
F. = as.matrix(.0001), Q = as.matrix(.0001^2))

# plot effective samples sizes
plot(get_ess(pf))
}
Index

data.frame, 4
formula, 4, 5
get_ess, 2
logLik.mssm, 3
logLik.mssmLaplace (logLik.mssm), 3
mssm, 2, 4, 6–11
mssm-Laplace, 5, 6
mssm-package, 2
mssm-pf, 5, 8, 9
mssm-smoother, 5, 9
mssm_control, 5, 8, 10
nloptr, 11
plot.mssm, 12
plot.mssmEss, 13
set.seed, 8, 11
terms.object, 5