Package ‘ideq’

December 19, 2019

Title Bayesian Dynamic Spatio-Temporal Models, Including the
Integrodifference Equation Model

Version 0.1.4

Description In contrast to other methods of modeling spatio-temporal data,
dynamic spatio-temporal models (DSTMs) directly model the dynamic
data-generating process.
'ideq' supports two main classes of DSTMs:
(1) empirical orthogonal function (EOF) models and
(2) integrodifference equation (IDE) models.
EOF models do not directly use any spatial information;
instead, they make use of observed relationships in the data
(the principal components) to model the underlying process.
In contrast, IDE models are based on diffusion dynamics and the process
evolution is governed by a (typically Gaussian) redistribution kernel.
Both types have a variety of options for specifying the model components,
including the process matrix, process error, and observation error.
The classic reference for DSTMs is
For IDE models specifically, see
and

Depends R (>= 3.5.0)
License GPL-2
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2
LinkingTo Rcpp, RcppArmadillo, rgen
Imports Rcpp, matrixcalc, pdist, mvtnorm
BugReports https://github.com/eastonhuch/ideq/issues
NeedsCompilation yes
Author Easton Huch [aut, cre],
Robert Richardson [ths]
Description

Fits a dynamic spatio-temporal model using empirical orthogonal functions (EOFs). The model does not require the spatial locations because the process model is based on the principal components of the data matrix. Three broad model types are supported:

1. RW: A random walk model for which the process matrix is the identity.
2. AR: An auto-regressive model for which the process matrix is diagonal and its elements are estimated.
3. Dense: A model in which the process matrix is a dense, estimated matrix.

For each broad model type, users can specify a variety of options including the size of the state space, the form of the process error, and whether to sample the observation error. Users can specify prior distributions for all sampled quantities using the ‘params’ argument.

Each model type mentioned above is a dynamic linear model (DLM), and the state vectors can be estimated using the forward filtering backward sampling algorithm. The other parameters are estimated with conditionally conjugate updates.

Usage

dstm_eof(
  Y,
  proc_model = "Dense",
  P = 4L,
  proc_error = "IW",
  n_samples = 1L,
  sample_sigma2 = TRUE,
)
verbose = FALSE,
    params = NULL
)

Arguments

Y (numeric matrix) S by T data matrix containing the response variable at S spatial
locations and T time points. The t-th column (NOT row) corresponds to the t-th
observation vector.

proc_model (character string) Process model: one of "RW" (identity process matrix), "AR"
diagonal process matrix), or "Dense" (dense process matrix).

P (integer) Number of EOFs or, in other words, the state space size.

proc_error (character string) Process error: "IW" (inverse-Wishart) or "Discount" (discount
factor).

n_samples (numeric scalar) Number of samples to draw

sample_sigma2 (logical) Whether to sample the variance of the iid observation error.

verbose (logical) Whether to print additional information; e.g., iteration in sampling al-
gorithm.

params (list) List of hyperparameter values; see details.

Details

This section explains how to specify custom hyperparameters using the ‘params’ argument. For
each distribution referenced below, we use the scale parameterization found on the distribution’s
Wikipedia page. You may specify the following as named elements of the ‘params’ list:

m_0: (numeric vector) The prior mean of the state vector at time zero (\(\theta_0\)).

C_0: (numeric matrix) The prior variance-covariance matrix of the state vector at time zero (\(\theta_0\)).

alpha_sigma2, beta_sigma2: (numeric scalars) The inverse-Gamma parameters (scale parameteri-
zation) of the prior distribution on the observation error (\(\sigma^2\)).

sigma2: (numeric scalar) The value to use for the observation error (\(\sigma^2\)) if ‘sample_sigma2’ =
FALSE.

mu_G: (numeric matrix) The prior mean for the process matrix G. If ‘proc_model’ = "AR", then
‘mu_G’ must be a diagonal matrix. If ‘proc_model’ = "Dense", then ‘mu_G’ has no constraints.

Sigma_G: (numeric matrix) The prior variance-covariance matrix for the process matrix. If proc_model
= "AR", then Sigma_G should be P by P and is the variance-covariance matrix for diag(G). If
proc_model = "Dense", then Sigma_G should be P^2 by P^2 and is the variance-covariance matrix
for vec(G).

alpha_lambda, beta_lambda: (numeric scalars) The inverse-Gamma parameters (scale parameteri-
zation) of the prior distribution on \(\lambda = (1 - \delta)/\delta\), where \(\delta\) is the discount factor.

scale_W: (numeric matrix) The scale matrix for the inverse-Wishart prior distribution on the variance-
covariance matrix of the process error (‘W’).

df_W: (numeric scalar) The degrees of freedom for the inverse-Wishart prior distribution on the variance-covariance matrix of the process error (‘W’).
References


See Also

[dstm_ide]

Examples

```
# Load example data
data("ide_standard")

# Illustrate methods
rw_model <- dstm_eof(ide_standard, proc_model="RW", verbose=TRUE)
summary(rw_model)
predict(rw_model)

# Other model types
dstm_eof(ide_standard, proc_model="AR") # Diagonal process matrix
dstm_eof(ide_standard, proc_model="Dense") # Dense process matrix

# Specify hyperparameters
P <- 4
dstm_eof(ide_standard, P=P,
         params=list(m_0=rep(1, P), C_0=diag(0.01, P),
                      scale_W=diag(P), df_W=100))
```

dstm_ide
Integrodifference equation (IDE) model

Description

dstm_ide fits a type of dynamic spatio-temporal model called an integrodifference equation (IDE) model. It estimates a redistribution kernel—a probability distribution controlling diffusion across time and space. Currently, only Gaussian redistribution kernels are supported.

The process model is decomposed with an orthonormal basis function expansion (a Fourier series). It can then be estimated as a special case of a dynamic linear model (DLM), using the forward filtering backward sampling algorithm to estimate the state vector. The kernel parameters are estimated with a random walk Metropolis-Hastings update. The other parameters are estimated with conditionally conjugate updates.
Usage

dstm_ide(
  Y,
  locs = NULL,
  knot_locs = NULL,
  proc_error = "IW",
  J = 1L,
  n_samples = 1L,
  sample_sigma2 = TRUE,
  verbose = FALSE,
  params = NULL
)

Arguments

Y (numeric matrix) S by T data matrix containing response variable at S spatial locations and T time points. The t-th column (NOT row) corresponds to the t-th observation vector.

locs (numeric matrix) S by 2 matrix containing the spatial locations of the observed data. The rows of 'locs' correspond with the rows of 'Y'.

knot_locs (integer or numeric matrix) Knot locations for the spatially varying IDE model. The kernel parameters are estimated at these locations and then mapped to the spatial locations of the observed data via process convolution. If an integer is provided, then the knots are located on an equally spaced grid with dimension ('knot_locs', 'knot_locs'). If a matrix is provided, then each row of the matrix corresponds to a knot location. If NULL, then the standard (spatially constant) IDE is fit.

proc_error (character string) Process error: "IW" (inverse-Wishart) or "Discount" (discount factor). "IW" is recommended because it is more computationally stable.

J (integer) Extent of the Fourier approximation. The size of the state space is \((2*J + 1)^2\).

n_samples (integer) Number of posterior samples to draw.

sample_sigma2 (logical) Whether to sample the variance of the iid observation error.

verbose (logical) Whether to print additional information; e.g., iteration in sampling algorithm.

params (list) List of hyperparameter values; see details.

Details

This section explains how to specify custom hyperparameters using the 'params' argument. For each distribution referenced below, we use the scale parameterization found on the distribution’s Wikipedia page. You may specify the following as named elements of the 'params' list:

m_0: (numeric vector) The prior mean of the state vector at time zero \((\theta_0)\).

C_0: (numeric matrix) The prior variance-covariance matrix of the state vector at time zero \((\theta_0)\).
alpha_sigma2, beta_sigma2: (numeric scalars) The inverse-Gamma parameters (scale parameterization) of the prior distribution on the observation error ($\sigma^2$).

sigma2: (numeric scalar) The value to use for the observation error ($\sigma^2$) if ‘sample_sigma2’ = FALSE.

alpha_lambda, beta_lambda: (numeric scalars) The inverse-Gamma parameters (scale parameterization) of the prior distribution on $\lambda = (1 - \delta)/\delta$, where $\delta$ is the discount factor.

scale_W: (numeric matrix) The scale matrix for the inverse-Wishart prior distribution on the variance-covariance matrix of the process error (‘W’).

df_W: (numeric scalar) The degrees of freedom for the inverse-Wishart prior distribution on the variance-covariance matrix of the process error (‘W’).

L: (numeric scalar) The period of the Fourier series approximation. The spatial locations and knot locations are rescaled to range from -$'L'$/4 to $'L'$/4 because the Fourier decomposition assumes that the spatial surface is periodic. Regardless of the value of ‘L’, kernel parameter estimates are back-transformed to the original scale.

smoothing: (numeric scalar) Controls the degree of smoothing in the process convolution for models with spatially varying kernel parameters. The values in the process convolution matrix are proportional to exp(d/‘smoothing’) where d is the distance between spatial locations before rescaling with ‘L’

mean_mu_kernel: (numeric vector) The mean of the normal prior distribution on ‘mu_kernel’, the mean of the redistribution kernel. In the spatially varying case, the prior distribution for ‘mu_kernel’ is assumed to be the same at every knot location.

var_mu_kernel: (numeric matrix) The variance of the normal prior distribution on ‘mu_kernel’, the mean of the redistribution kernel.

scale_Sigma_kernel: (numeric matrix) The scale matrix for the inverse-Wishart prior distribution on ‘Sigma_kernel’, the variance-covariance matrix of the redistribution kernel.

df_Sigma_kernel: (numeric scalar) The degrees of freedom for the inverse-Wishart prior distribution on ‘Sigma_kernel’, the variance-covariance matrix of the redistribution kernel.

proposal_factor_mu: (numeric scalar) Controls the variance of the proposal distribution for ‘mu_kernel’. The proposals have a variance of ‘proposal_factor_mu’$^2 \times ‘var_mu_kernel’. ‘proposal_factor_mu’ must generally be set lower for spatially varying models.

proposal_factor_Sigma: (numeric scalar) Controls the variance of the proposal distribution for ‘Sigma_kernel’. As is the case with ‘proposal_factor_mu’, a higher value corresponds to a higher variance. The degrees of freedom for the proposal distribution for ‘Sigma_kernel’ is ncol(‘locs’) + ‘df_Sigma_kernel’ / ‘proposal_factor_Sigma’. ‘proposal_factor_Sigma’ must generally be set lower for spatially varying models.

kernel_samples_per_iter: (numeric scalar) Number of times to update the kernel parameters per iteration of the sampling loop.

References


See Also

[dstm_eof]

Examples

# Load example data
data("ide_standard", "ide_spatially_varying", "ide_locations")

# Basic IDE model with one kernel
mod <- dstm_ide(ide_standard, ide_locations)
predict(mod)
summary(mod)

# IDE model with spatially varying kernel
dstm_ide(ide_spatially_varying, ide_locations, knot_locs=4)

# Fix sigma2

dstm_ide(ide_standard, ide_locations,
  sample_sigma2=FALSE, params=list(sigma2=1))

# Set proposal scaling factors, number of kernel updates per iteration,
# and prior distribution on kernel parameters

dstm_ide(ide_standard, ide_locations,
  params=list(proposal_factor_mu=2, proposal_factor_Sigma=3,
  kernel_updates_per_iter=2,
  scale_Sigma_kernel=diag(2), df_Sigma_kernel=100,
  mean_mu_kernel=c(0.2, 0.4), var_mu_kernel=diag(2)))

# Set priors on state vector, process error, and observation error

J <- 1
P <- (2*J + 1)^2

dstm_ide(ide_standard, ide_locations,
  params=list(m_0=rep(1, P), C_0=diag(0.01, P),
  alpha_sigma2=20, beta_sigma2=20,
  scale_W=diag(P), df_W=100))
ide_spatially_varying

Description
A matrix containing the 400 two-dimensional spatial locations of the following data sets: `ide_standard` and `ide_spatially_varying`. The rows of these two data sets correspond with the rows of `ide_locations`.

Usage
ide_locations

Format
A numeric matrix with 400 rows and 2 columns

Source
Generated as part of Easton Huch’s MS project at BYU

See Also
`[ide_standard]/[ide_spatially_varying]`, the data sets that correspond to these spatial locations

ide_spatially_varying  Simulated data from a spatially varying IDE model

Description
A matrix containing simulated data points on a 20 X 20 grid at 20 time points. The rows correspond to the 400 spatial locations, and the columns, to the 20 time points.

Usage
ide_spatially_varying

Format
A numeric matrix with 400 rows and 20 columns

Source
Randomly generated as part of Easton Huch’s MS project at BYU

See Also
`[ide_locations]` for a matrix containing the spatial locations corresponding to the rows of `ide_spatially_varying` and `[ide_standard]` for a similar data set generated from a standard ide model
ide_standard

Simulated data from a standard IDE model

Description
A matrix containing simulated data points on a 20 X 20 grid at 20 time points. The rows correspond to the 400 spatial locations, and the columns, to the 20 time points.

Usage
ide_standard

Format
A numeric matrix with 400 rows and 20 columns

Source
Randomly generated as part of Easton Huch’s MS project at BYU

See Also
[ide_locations] for a matrix containing the spatial locations corresponding to the rows of ‘ide_standard’ and [ide_spatially_varying] for a similar data set generated from a spatially varying ide model

predict.dstm

Predict Method for DSTM Fits

Description
Generates samples from the posterior predictive distribution at future time points for (1) the observation vector and (2) the state vector.

Usage
## S3 method for class 'dstm'
predict(
  object,
  K = 1,
  only_K = FALSE,
  return_ys = TRUE,
  return_thetas = FALSE,
  burnin = NULL,
  ...
)
Arguments

- **object**: A `dstm` object
- **K**: (integer scalar) The number of future time periods for which to generate predictions
- **only_K**: (logical scalar) Whether to return predictions for time period \( T+K \) only (as opposed to \( T+1, T+2, \ldots, T+K \))
- **return_ys**: (logical scalar) Whether to return samples from the posterior predictive distribution of the observation vector (ys)
- **return_thetas**: (logical scalar) Whether to return samples from the posterior predictive distribution of the state vector (thetas)
- **burnin**: (integer scalar) The number of samples to discard as burn-in. If `object$burnin` exists, this argument will override it.
- ... Arguments passed to other methods (necessary for S3 generic compatibility)

Details

The posterior predictive samples are returned in a matrix or 3-D array, depending on whether samples from multiple time points are requested. The dimensions are always in the following order:

1. The index of the value within the state or observation vector.
2. The time period
3. The sample number

Examples

```r
data("ide_standard", "ide_locations")

# IDE example
mod_ide <- dstm_ide(ide_standard, ide_locations)
predict(mod_ide)
predict(mod_ide, K=4, return_thetas=TRUE)

# EOF example
mod_eof <- dstm_eof(ide_standard, n_samples=2)
predict(mod_eof, K=2, only_K=TRUE, burnin=1)
```

print.dstm

Print Method for DSTM Fits

**Description**

Prints a summary for a `dstm` object by calling `summary.dstm()`.

**Usage**

```r
## S3 method for class 'dstm'
print(x, x_name = deparse(substitute(x)), ...)
```
Arguments

- **x**: A `dstm` object
- **x_name**: (optional) Object name to display
- **...**: Arguments passed to other methods (necessary for S3 generic compatibility)

See Also

- `summary.dstm()`

---

**summary.dstm**

*Summary Method for DSTM Fits*

**Description**

Prints summary information for `dstm` objects.

**Usage**

```r
## S3 method for class 'dstm'
summary(object, object_name = deparse(substitute(object)), ...)
```

**Arguments**

- **object**: A `dstm` object
- **object_name**: The name to be printed in the summary (if desired)
- **...**: Arguments passed to other methods (necessary for S3 generic compatibility)

**See Also**

- `print.summary()`

**Examples**

```r
# Load example data
data("ide_standard", "ide_locations")
mod_ide <- dstm_ide(ide_standard, ide_locations)
summary(mod_ide)
```
Index

*Topic **datasets**
  ide_locations, 7
  ide_spatially_varying, 8
  ide_standard, 9

dstm_eof, 2
dstm_ide, 4

ide_locations, 7
ide_spatially_varying, 8
ide_standard, 9

predict.dstm, 9
print.dstm, 10

summary.dstm, 11