Package: spDBL (via r-universe)

June 9, 2026

Title Dynamic Bayesian Learning for Spatiotemporal Mechanistic Models

Version 1.0.2

Description Provides tools for Bayesian learning of spatiotemporal dynamical mechanistic models. Includes methods for parameter estimation, simulation, and inference using hierarchical and state-space modeling approaches, following Banerjee, Chen, Frankenburg and Zhou (2025) <https://jmlr.org/papers/v26/22-0896.html>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0)

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppEigen

Imports Rcpp, matrixsampling, invgamma, deSolve, ReacTran, LaplacesDemon, matrixcalc, mniw, utils, stats, ggpubr, ggplot2, readr, magrittr, rlang, scales

Suggests testthat (>= 3.0.0), here, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Author Xiang Chen [aut, cre], Sudipto Banerjee [aut]

Maintainer Xiang Chen <xiangchen@ucla.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-09 08:44:41 UTC

RemoteUrl https://github.com/cran/spDBL

RemoteRef HEAD

RemoteSha ce91035c804b0c25514ecc18212184d23b5d285b

Contents

`BS` *Backward sampler for the Forward Filter Backward Sampler (FFBS)*

Description

Performs the backward sampling pass of the FFBS algorithm using the filtered distributions produced by `FF`. Iterates from time T back to 1, drawing smoothed state samples at each step via `BS_1step_cpp`.

Usage

```
BS(res_ff, G_ls, nT, delta = 1, verbose = FALSE)
```

Arguments

res_ff	List of length nT (plus a prior element) returned by FF. Each element res_ff[[t]] must contain mt, Mt, at, and At.
G_ls	Either a single numeric matrix (state transition matrix G, constant over time) or a list of nT matrices (time-varying).
nT	Integer. Number of time steps.
delta	Numeric scalar. Right-variance discount factor. Defaults to 1.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list with:

st Array of dimension c(p, q, nT) containing the smoothed state means.

St Array of dimension c(p, p, nT) containing the smoothed state left-covariance matrices.

See Also

FF, FFBS

---

cal_Bt_bt *Update the posterior mean and covariance of the discrepancy field*

Description

Computes the posterior mean vector $B_t b_t$ and covariance matrix B_t of the discrepancy u_t given the current predictive mean μ_t, observations $z_t - u_t$, and variance τ_t^2.

Usage

```
cal_Bt_bt(mut, invSigma, tau2t, zt_ut, I_Stilde)
```

Arguments

mut	Numeric vector. Predictive mean at calibration locations.
invSigma	Numeric matrix. Inverse of the predictive spatial covariance.
tau2t	Numeric scalar. Current variance τ_t^2.
zt_ut	Numeric vector. Observation residuals $z_t - u_t$.
I_Stilde	Numeric matrix. Identity-like matrix for the discrepancy prior precision (N_sp_train x N_sp_train).

Value

A named list with:

Bt Numeric matrix. Posterior covariance of the discrepancy.

Btbt Numeric vector. Posterior mean of the discrepancy.

cal_errorbar *Compute median and 95% credible interval across rows*

Description

For each row of the matrix `X`, computes the median and the 2.5th and 97.5th percentiles across columns (samples).

Usage

```
cal_errorbar(X)
```

Arguments

`X`	Numeric matrix. Rows correspond to spatial locations or parameters; columns correspond to posterior samples.

Value

A data frame with columns:

med Row-wise median.

lower Row-wise 2.5th percentile.

upper Row-wise 97.5th percentile.

See Also

`cal_errorbar_mean`

cal_errorbar_mean *Compute mean and 95% credible interval across rows*

Description

For each row of the matrix `X`, computes the mean and the 2.5th and 97.5th percentiles across columns (samples).

Usage

```
cal_errorbar_mean(X)
```

Arguments

`X`	Numeric matrix. Rows correspond to spatial locations or parameters; columns correspond to posterior samples.

Value

A data frame with columns:

med Row-wise mean.

lower Row-wise 2.5th percentile.

upper Row-wise 97.5th percentile.

See Also

cal_errorbar

`cal_jacobian_logit_uniform` *Log absolute Jacobian of the logit-uniform transformation*

Description

Computes the log (or raw) absolute Jacobian of the transformation from the constrained parameter `eta` (bounded in `[eta_limit_low, eta_limit_high]`) to the unconstrained logit scale. Zero entries of `eta` are excluded from the product.

Usage

```
cal_jacobian_logit_uniform(eta, eta_limit_low, eta_limit_high, log = FALSE)
```

Arguments

`eta`	Numeric vector. Constrained parameter values.
`eta_limit_low`	Numeric vector. Lower bounds for each element of `eta`.
`eta_limit_high`	Numeric vector. Upper bounds for each element of `eta`.
`log`	Logical. If `TRUE`, returns the log Jacobian. Defaults to `FALSE`.

Value

Numeric scalar (log absolute Jacobian or absolute Jacobian).

check_pds *Check and repair a matrix to be positive definite and symmetric*

Description

Verifies that a matrix is approximately symmetric (within a relative and absolute tolerance). If so, symmetrises it and, if still not positive definite, adds a small diagonal ridge. Raises an error if the asymmetry exceeds the tolerance.

Usage

```
check_pds(C, eps = 10^(-5), per = 0.05)
```

Arguments

`C`	Numeric matrix to check.
`eps`	Numeric scalar. Absolute tolerance for the Frobenius norm of `C - t(C)`. Defaults to `1e-5`.
`per`	Numeric scalar. Relative tolerance: the ratio of the asymmetry norm to the mean of `C`. Defaults to `0.05`.

Value

The (possibly corrected) positive definite symmetric matrix.

See Also

`make_pds`

dMTig *Log density of the matrix-T distribution with inverse-gamma right covariance*

Description

Evaluates the log (or raw) density of the matrix-T distribution that arises when the right covariance is $\sigma^2 R$ and $\sigma^2 \sim IG(\nu, d)$.

Usage

```
dMTig(Y, m, M, nu, d, R, log = TRUE)
```

Arguments

`Y`	Numeric matrix. Observation matrix ($p \times S$).
`m`	Numeric matrix. Mean matrix ($p \times S$).
`M`	Numeric matrix. Left covariance matrix ($p \times p$, positive definite).
`nu`	Numeric scalar. Degrees of freedom (shape parameter of the inverse-gamma on σ^2).
`d`	Numeric scalar. Rate parameter of the inverse-gamma on σ^2).
`R`	Numeric matrix. Right correlation matrix ($S \times S$, positive definite).
`log`	Logical. If `TRUE` (default), returns the log density.

Value

Numeric scalar. Log density (or density if `log = FALSE`).

`dt_emulation` *Example emulation dataset*

Description

A list of example training and testing objects used to illustrate the emulation workflow in the package.

Usage

```
dt_emulation
```

Format

A list with 4 elements:

pde_para_train Training set of PDE parameter values.

pde_para_test Test set of PDE parameter values.

dt_pde_train Training set of PDE-based outputs corresponding to `pde_para_train`.

dt_pde_test Test set of PDE-based outputs corresponding to `pde_para_test`.

Source

Simulated data / package example data

`emulator_learn` *Fit an FFBS-based dynamic emulator*

Description

Fits a dynamic linear emulator to training PDE output using Forward Filtering Backward Sampling (FFBS). The wrapper constructs the input-parameter Gaussian process covariance, partitions spatial output into episode blocks, builds autoregressive design matrices, runs `FFBS`, and draws posterior state samples with `FFBS_sampling`.

Usage

```
emulator_learn(
  pde_para_train,
  dt_pde_train,
  Nx,
  Ny,
  F_ls_train = "default",
  F_ls_test = "default",
  N_people = 10000,
  nsam = 10,
  AR_choice = 2,
  episode_window = 1,
  gp_tune = 0.5,
  gp_sigma2 = 1.1,
  gp_tau2 = 10^(-4)
)
```

Arguments

`pde_para_train`	Numeric matrix. Training input parameters, with rows corresponding to simulator runs and columns to parameters.
`dt_pde_train`	List of length `nT`. Each element is a numeric matrix of training PDE outputs at one time step, with rows corresponding to simulator runs and columns to spatial locations.
`Nx`	Integer. Number of spatial grid cells in the x-direction.
`Ny`	Integer. Number of spatial grid cells in the y-direction.
`F_ls_train`	Either `"default"` or a list of autoregressive design matrices for the training data. If `"default"`, the design matrices are generated from `dt_pde_train`.
`F_ls_test`	Deprecated/unused argument retained for compatibility.
`N_people`	Numeric scalar. Population size associated with the PDE simulator. Stored in the returned setup list. Defaults to `10000`.
`nsam`	Integer. Number of posterior state samples drawn by `FFBS_sampling`. Defaults to `10`.

AR_choice	Integer. Autoregressive design order. Use 1 for gen_F_ls_AR1_EP or 2 for gen_F_ls_AR2_EP. Defaults to 2.
episode_window	Integer. Number of y-direction columns per episode block. Defaults to 1.
gp_tune	Numeric scalar. Tuning factor used to set the exponential kernel range from the maximum pairwise input distance. Defaults to 0.5.
gp_sigma2	Numeric scalar. Marginal variance parameter for the input Gaussian process kernel. Defaults to 1.1.
gp_tau2	Numeric scalar. Nugget variance parameter for the input Gaussian process kernel. Defaults to 10^(-4).

Value

A named list with:

para_ffbs Output of FFBS.

res_ffbs Posterior state samples returned by FFBS_sampling.

setup A list of training data, blocking information, kernel parameters, generated design matrices, and other quantities needed by emulator_predict.

See Also

emulator_predict, FFBS, FFBS_sampling, gen_F_ls_AR1_EP, gen_F_ls_AR2_EP

---

emulator_predict *Predict PDE output from a fitted FFBS emulator*

Description

Uses a fitted emulator returned by emulator_learn to predict PDE output at new input parameter values. Prediction can use either the exact posterior predictive mean from FFBS_predict_exact or Monte Carlo posterior predictive draws from FFBS_predict_MC.

Usage

```
emulator_predict(
  emulator,
  input_new,
  dt_pde_test,
  F_new_ls = "default",
  MC = FALSE
)
```

Arguments

`emulator`	Named list. Fitted emulator object returned by `emulator_learn`.
`input_new`	Numeric matrix. New input parameter values at which to predict, with rows corresponding to new simulator settings.
`dt_pde_test`	List of length `nT_ori`. PDE output matrices for the new inputs, used to construct default autoregressive design matrices when `F_new_ls = "default"`.
`F_new_ls`	Either `"default"` or a list of autoregressive design matrices for the new inputs. Defaults to `"default"`.
`MC`	Logical. If `FALSE`, return exact posterior predictive means. If `TRUE`, return Monte Carlo posterior predictive draws. Defaults to `FALSE`.

Details

If `F_new_ls = "default"`, the function constructs the required autoregressive design matrices for the new inputs from `dt_pde_test` using the autoregressive order and episode blocking structure stored in `emulator`.

Value

If `MC = FALSE`, a named list of length `nT_ori`, where each element is a matrix of exact posterior predictive means for one original time step. If `MC = TRUE`, a named list of length `nT_ori`, where each element is an array of Monte Carlo predictive draws for one original time step.

See Also

`emulator_learn`, `FFBS_predict_exact`, `FFBS_predict_MC`, `recover_from_EP_exact`, `recover_from_EP_MC`

`expit` *Logistic (expit) function*

Description

Computes the logistic sigmoid $1/(1+e^{-x})$.

Usage

```
expit(x)
```

Arguments

`x`	Numeric vector or matrix.

Value

Numeric vector or matrix with values in $(0, 1)$.

FF *Forward Filter for the MNIW dynamic linear model*

Description

Runs the full forward filtering pass over `nT` time steps under the Matrix Normal Inverse Wishart (MNIW) model. At each step, calls `FF_1step_cpp` to update the state mean and covariance as well as the inverse-Wishart parameters for the right-covariance matrix Σ.

Usage

```
FF(Y, F_ls, G_ls, W_ls, V_ls, m0, M0, n0, D0, nT, delta = 1, verbose = FALSE)
```

Arguments

`Y`	List of length `nT`. Each element `Y[[t]]` is the $N \times q$ data matrix at time t.
`F_ls`	Either a single $N \times p$ covariate matrix (constant over time) or a list of `nT` such matrices (time-varying).
`G_ls`	Either a single $p \times p$ state transition matrix or a list of `nT` such matrices.
`W_ls`	Either a single $p \times p$ state noise left-covariance matrix or a list of `nT` such matrices.
`V_ls`	Either a single $N \times N$ observation noise left-covariance matrix or a list of `nT` such matrices.
`m0`	Numeric matrix. Prior mean of the state $\beta_0 \mid D_0$ ($p \times q$).
`M0`	Numeric matrix. Prior left-covariance of $\beta_0 \mid D_0$ ($p \times p$).
`n0`	Numeric scalar. Prior degrees of freedom of the inverse-Wishart on $\Sigma \mid D_0$.
`D0`	Numeric matrix. Prior scale matrix of the inverse-Wishart on $\Sigma \mid D_0$ ($q \times q$).
`nT`	Integer. Number of time steps.
`delta`	Numeric scalar. Discount factor for the right-variance matrix. Defaults to `1.0`.
`verbose`	Logical. If `TRUE`, emits progress messages. Defaults to `FALSE`.

Value

A named list of length `nT + 1`. Elements `"T1"` through `"T<nT>"` each contain a list with filtered parameters `nt`, `Dt`, `at`, `At`, `mt`, `Mt`. The additional element `prior` stores `m0`, `M0`, `n0`, `D0`.

See Also

BS, FFBS

`FF_1step_R_I` *Single forward filter step (identity right-covariance)*

Description

Performs one step of the forward filter under the model where the right covariance of the state and observation is the identity matrix (no inverse-Wishart update for Σ).

Usage

```
FF_1step_R_I(Yt, Ft, Gt, Wt, Vt, mt_1, Mt_1, delta = 1)
```

Arguments

`Yt`	Numeric matrix. Data matrix at time t ($N \times S$).
`Ft`	Numeric matrix. Covariate matrix at time t ($N \times p$).
`Gt`	Numeric matrix. State transition matrix ($p \times p$).
`Wt`	Numeric matrix. State noise left-covariance ($p \times p$).
`Vt`	Numeric matrix. Observation noise left-covariance ($N \times N$).
`mt_1`	Numeric matrix. Filtered state mean at $t-1$ ($p \times S$).
`Mt_1`	Numeric matrix. Filtered state left-covariance at $t-1$ ($p \times p$).
`delta`	Numeric scalar. Discount factor. Defaults to `1.0`.

Value

A named list with:

at One-step-ahead state mean.

At One-step-ahead state left-covariance.

mt Filtered state mean.

Mt Filtered state left-covariance.

delta The discount factor passed in.

See Also

`FF_I`

`FF_1step_R_sigma2R` *Single forward filter step (scalar right-covariance, sigma-squared times R)*

Description

Performs one step of the forward filter under the model where the right covariance of Y is $\sigma^2 R$ with $\sigma^2 \sim IG(n, d)$. The inverse of R (`Rinv`) must be precomputed and passed in.

Usage

```
FF_1step_R_sigma2R(Yt, Ft, Gt, Wt, Vt, mt_1, Mt_1, nt_1, Dt_1, Rinv, delta = 1)
```

Arguments

`Yt`	Numeric matrix. Data matrix at time t ($N \times S$).
`Ft`	Numeric matrix. Covariate matrix at time t ($N \times p$).
`Gt`	Numeric matrix. State transition matrix at time t ($p \times p$).
`Wt`	Numeric matrix. State noise left-covariance at time t ($p \times p$).
`Vt`	Numeric matrix. Observation noise left-covariance at time t ($N \times N$).
`mt_1`	Numeric matrix. Filtered state mean at $t-1$ ($p \times S$).
`Mt_1`	Numeric matrix. Filtered state left-covariance at $t-1$ ($p \times p$).
`nt_1`	Numeric scalar. Shape parameter of $\sigma^2 \mid D_{t-1}$.
`Dt_1`	Numeric scalar. Rate parameter of $\sigma^2 \mid D_{t-1}$.
`Rinv`	Numeric matrix. Precomputed inverse of the spatial correlation matrix R ($S \times S$).
`delta`	Numeric scalar. Right-variance discount factor. Defaults to `1.0`.

Value

A named list with updated filtering parameters:

nt Updated shape parameter.

Dt Updated rate parameter.

at One-step-ahead state mean ($p \times S$).

At One-step-ahead state left-covariance ($p \times p$).

mt Filtered state mean ($p \times S$).

Mt Filtered state left-covariance ($p \times p$).

delta The discount factor passed in.

See Also

`FF_sigma2R`

FF_bigdata_R *Forward Filter for big data stored in CSV files (MNIW model)*

Description

Runs the forward filtering pass for large datasets that are partitioned into spatial blocks and stored as CSV files on disk. At each time step the spatial domain is traversed block-by-block using a snake traversal order (see `generate.grid.rowsnake`), and results are written to disk to avoid exhausting memory.

Usage

```
FF_bigdata_R(
  Y_ls,
  F_ls,
  G_ls,
  W_ls,
  V_ls,
  m0,
  M0,
  n0,
  D0,
  nT,
  fnrow,
  fncol,
  bnrow,
  bncol,
  path_out,
  delta = 1,
  verbose = FALSE
)
```

Arguments

`Y_ls`	Character vector of length `nT`. File paths to the response CSV files, one per time step.
`F_ls`	Either a single file path (constant F_t) or a character vector of length `nT` (time-varying). Can also contain a second set of columns (AR2 lags) for the same file at offset `fncol`.
`G_ls`	Either a single file path (constant G_t) or a character vector of length `nT` (time-varying).
`W_ls`	Either a single file path (constant W_t) or a character vector of length `nT` (time-varying).
`V_ls`	Either a single file path (constant V_t) or a character vector of length `nT` (time-varying). Expected to be a square block submatrix indexed by the row range of each spatial block.

m0	Numeric matrix. Prior state mean ($p \times q$).
M0	Numeric matrix. Prior left-covariance of the state ($p \times p$).
n0	Numeric scalar. Prior degrees of freedom of the inverse-Wishart.
D0	Numeric matrix. Prior scale matrix of the inverse-Wishart.
nT	Integer. Number of time steps (files).
fnrow	Integer. Total number of rows in each data file.
fncol	Integer. Total number of columns in each data file.
bnrow	Integer. Number of rows per spatial block.
bncol	Integer. Number of columns per spatial block.
path_out	Character. Directory path where output CSV files are written.
delta	Numeric scalar. Right-variance discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list of character vectors with paths to the output CSV files:

nt_ls Paths to filtered degrees-of-freedom files.

Dt_ls Paths to filtered scale-matrix files.

mt_ls Paths to filtered state-mean files.

Mt_ls Paths to filtered left-covariance files.

at_ls Paths to predicted state-mean files.

At_ls Paths to predicted left-covariance files.

FT Path to the covariate matrix at the final time step.

VT Path to the noise covariance matrix at the final time step.

See Also

FF, generate.grid.rowsnake

FF_I *Forward Filter with identity right-covariance*

Description

Runs the full forward filtering pass over nT time steps under the model where the right covariance is fixed at the identity (no inverse-Wishart update). Calls FF_1step_R_I at each step.

Usage

```
FF_I(Y, F_ls, G_ls, W_ls, V_ls, m0, M0, nT, delta = 1, verbose = FALSE)
```

Arguments

Y	List of length nT. Each element is the data matrix at time t.
F_ls	Covariate matrix or list of matrices (see FF).
G_ls	State transition matrix or list (see FF).
W_ls	State noise left-covariance matrix or list (see FF).
V_ls	Observation noise left-covariance matrix or list (see FF).
m0	Numeric matrix. Prior state mean.
M0	Numeric matrix. Prior state left-covariance.
nT	Integer. Number of time steps.
delta	Numeric scalar. Discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list of length nT + 1. Elements "T1" through "T<nT>" contain at, At, mt, Mt. The element prior stores m0 and M0.

See Also

FF_1step_R_I, FFBS_I

FF_sigma2R *Forward Filter for the scalar-sigma-squared-times-R model*

Description

Runs the full forward filtering pass over nT time steps under the model where the right covariance of Y is $\sigma^2 R$ with $\sigma^2 \sim IG(n_0, d_0)$.

Usage

```
FF_sigma2R(
  Y,
  F_ls,
  G_ls,
  W_ls,
  V_ls,
  m0,
  M0,
  n0,
  D0,
  nT,
  R,
  delta = 1,
  verbose = FALSE
)
```

Arguments

Y	List of length nT. Each element is the $N \times S$ data matrix at the corresponding time step.
F_ls	Covariate matrix or list of matrices (see FF).
G_ls	State transition matrix or list (see FF).
W_ls	State noise left-covariance matrix or list (see FF).
V_ls	Observation noise left-covariance matrix or list (see FF).
m0	Numeric matrix. Prior state mean ($p \times S$).
M0	Numeric matrix. Prior state left-covariance ($p \times p$).
n0	Numeric scalar. Prior shape of σ^2.
D0	Numeric scalar. Prior rate of σ^2.
nT	Integer. Number of time steps.
R	Numeric matrix. Fixed spatial correlation matrix ($S \times S$). Its Cholesky inverse is used internally.
delta	Numeric scalar. Right-variance discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list of length nT + 1 (same structure as FF) with an additional scalar Dt in place of a matrix.

See Also

FF_1step_R_sigma2R, FFBS_sigma2R

FFBS *Forward Filter Backward Sampler (MNIW model)*

Description

Runs the complete FFBS algorithm under the Matrix Normal Inverse Wishart (MNIW) model: first applies the forward filter (FF) and then the backward sampler (BS).

Usage

```
FFBS(Y, F_ls, G_ls, W_ls, V_ls, m0, M0, n0, D0, nT, delta = 1, verbose = FALSE)
```

Arguments

Y	List of length nT. Each element is the $N \times q$ data matrix at the corresponding time step.
F_ls	Covariate matrix or list of matrices (see FF).
G_ls	State transition matrix or list (see FF).
W_ls	State noise left-covariance matrix or list (see FF).
V_ls	Observation noise left-covariance matrix or list (see FF).
m0	Numeric matrix. Prior state mean ($p \times q$).
M0	Numeric matrix. Prior state left-covariance ($p \times p$).
n0	Numeric scalar. Prior degrees of freedom of the inverse-Wishart.
D0	Numeric matrix. Prior scale matrix of the inverse-Wishart.
nT	Integer. Number of time steps.
delta	Numeric scalar. Right-variance discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list with:

ff Output of FF: filtered distributions for each time step.

bs Output of BS: smoothed state means (st) and left-covariances (St).

See Also

FF, BS, FFBS_sampling

FFBS_I *Forward Filter Backward Sampler (identity right-covariance)*

Description

Runs the complete FFBS algorithm with the right covariance fixed at the identity matrix (no inverse-Wishart update for Σ).

Usage

```
FFBS_I(Y, F_ls, G_ls, W_ls, V_ls, m0, M0, nT, delta = 1, verbose = FALSE)
```

Arguments

Y	List of length nT. Data matrices.
F_ls	Covariate matrix or list (see FF).
G_ls	State transition matrix or list (see FF).
W_ls	State noise left-covariance matrix or list (see FF).
V_ls	Observation noise left-covariance matrix or list (see FF).
m0	Numeric matrix. Prior state mean.
M0	Numeric matrix. Prior state left-covariance.
nT	Integer. Number of time steps.
delta	Numeric scalar. Discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list with ff (output of FF_I) and bs (output of BS).

See Also

FF_I, BS, FFBS_sampling_I

FFBS_predict_exact *Exact posterior predictive mean using FFBS smoothed states (MNIW model)*

Description

Computes the analytical posterior predictive mean at new spatial locations using the smoothed state means s_t from FFBS and an exponential GP kernel for the cross-covariance.

Usage

```
FFBS_predict_exact(
  Y,
  para_ffbs,
  F_ls,
  F_new_ls,
  input,
  input_new,
  nT,
  phi_para,
  gp_sigma2 = 1.1,
  gp_tau2 = 10^(-4),
  delta = 1
)
```

Arguments

Y	List of length nT. Observed data matrices.
para_ffbs	List. Output of FFBS, containing ff and bs.
F_ls	Covariate matrix or list for observed locations.
F_new_ls	Covariate matrix or list for new locations.
input	Numeric matrix. Coordinates of observed locations.
input_new	Numeric matrix or vector. Coordinates of new locations.
nT	Integer. Number of time steps.
phi_para	Numeric scalar. Range parameter of the exponential kernel.
gp_sigma2	Numeric scalar. Variance parameter of the GP kernel. Defaults to 1.1.
gp_tau2	Numeric scalar. Nugget variance of the GP kernel. Defaults to 1e-4.
delta	Numeric scalar. Discount factor. Defaults to 1.0.

Value

A named list of length nT. Each element "T<t>" is a matrix of dimension c(N_new, q) with the posterior predictive mean at the new locations.

See Also

FFBS_predict_MC, gen_exp_kernel

FFBS_predict_MC *Monte Carlo prediction using FFBS output (MNIW model)*

Description

Estimates posterior predictive means at new spatial locations using Monte Carlo integration over the posterior samples of the state Θ_t. Uses an exponential GP kernel to compute the cross-covariance between observed and new locations.

Usage

```
FFBS_predict_MC(
  nsam,
  Y,
  res_ffbs,
  F_ls,
  F_new_ls,
  input,
  input_new,
  nT,
  phi_para,
  gp_sigma2 = 1.1,
  gp_tau2 = 10^(-4),
  delta = 1
)
```

Arguments

nsam	Integer. Number of posterior samples to average over.
Y	List of length nT. Observed data matrices.
res_ffbs	List. Posterior samples of Θ_t, as returned by FFBS_sampling.
F_ls	Covariate matrix or list for observed locations.
F_new_ls	Covariate matrix or list for new (prediction) locations.
input	Numeric matrix. Coordinates of observed locations ($N \times d$).
input_new	Numeric matrix or vector. Coordinates of new locations.
nT	Integer. Number of time steps.
phi_para	Numeric scalar. Range parameter of the exponential kernel.
gp_sigma2	Numeric scalar. Variance parameter of the GP kernel. Defaults to 1.1.
gp_tau2	Numeric scalar. Nugget variance of the GP kernel. Defaults to 1e-4.
delta	Numeric scalar. Discount factor. Defaults to 1.0.

Value

A named list of length nT. Each element "T<t>" is an array of dimension c(N_new, q, nsam) with posterior predictive mean samples at the new locations.

See Also

FFBS_predict_exact, gen_exp_kernel

FFBS_sampling *Draw posterior samples from FFBS output (MNIW model)*

Description

Given the filtered and smoothed distributions from FFBS, draws nsam joint posterior samples of the state matrices $\Theta_1, \ldots, \Theta_T$ and the covariance Σ using the MNIW sampler.

Usage

```
FFBS_sampling(nsam, para_ffbs, F_ls, G_ls, nT, delta = 1, verbose = FALSE)
```

Arguments

nsam	Integer. Number of posterior samples to draw.
para_ffbs	List. Output of FFBS, containing elements ff and bs.
F_ls	Covariate matrix or list of matrices (see FF).
G_ls	State transition matrix or list (see FF).
nT	Integer. Number of time steps.
delta	Numeric scalar. Discount factor. Defaults to 1.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list of length `nT + 1`. Elements `"T1"` through `"T<nT>"` are arrays of dimension `c(p, q, nsam)` containing posterior samples of Θ_t. The element `Sigma` is an array of dimension `c(q, q, nsam)` containing posterior samples of Σ.

See Also

`FFBS`, `MNIW_sampler`

`FFBS_sampling_I` *Draw posterior samples from FFBS output (identity right-covariance)*

Description

Draws `nsam` posterior samples of the state Θ_t from the smoothed distributions produced by `FFBS_I`, assuming an identity right-covariance matrix.

Usage

```
FFBS_sampling_I(nsam, para_ffbs, F_ls, G_ls, nT, delta = 1, verbose = FALSE)
```

Arguments

`nsam`	Integer. Number of posterior samples.
`para_ffbs`	List. Output of `FFBS_I`.
`F_ls`	Covariate matrix or list (see `FF`).
`G_ls`	State transition matrix or list (see `FF`).
`nT`	Integer. Number of time steps.
`delta`	Numeric scalar. Discount factor. Defaults to `1`.
`verbose`	Logical. If `TRUE`, emits progress messages. Defaults to `FALSE`.

Value

A named list of length `nT`. Each element `"T<t>"` is an array of dimension `c(p, q, nsam)` with posterior samples of the state at time t.

See Also

`FFBS_I`

FFBS_sampling_sigma2R *Draw posterior samples from FFBS output (scalar sigma-squared-times-R model)*

Description

Draws `nsam` posterior samples of the state Θ_t and the scalar variance σ^2 using the MNIG sampler, given the FFBS output under the scalar-sigma model.

Usage

```
FFBS_sampling_sigma2R(
  nsam,
  para_ffbs,
  F_ls,
  G_ls,
  nT,
  R,
  delta = 1,
  verbose = FALSE
)
```

Arguments

`nsam`	Integer. Number of posterior samples.
`para_ffbs`	List. Output of `FFBS_sigma2R`.
`F_ls`	Covariate matrix or list (see `FF`).
`G_ls`	State transition matrix or list (see `FF`).
`nT`	Integer. Number of time steps.
`R`	Numeric matrix. Fixed spatial correlation matrix.
`delta`	Numeric scalar. Discount factor. Defaults to `1`.
`verbose`	Logical. If `TRUE`, emits progress messages. Defaults to `FALSE`.

Value

A named list of length `nT + 1`. Elements `"T1"` through `"T<nT>"` are arrays of dimension `c(p, q, nsam)` with state samples. The element `Sigma` is a vector of length `nsam` with samples of σ^2.

See Also

`FFBS_sigma2R`, `MNIG_sampler`

FFBS_sigma2R *Forward Filter Backward Sampler (scalar sigma-squared-times-R model)*

Description

Runs the complete FFBS algorithm under the model where the right covariance is $\sigma^2 R$ with $\sigma^2 \sim IG(n_0, d_0)$.

Usage

```
FFBS_sigma2R(
  Y,
  F_ls,
  G_ls,
  W_ls,
  V_ls,
  m0,
  M0,
  n0,
  D0,
  nT,
  R,
  delta = 1,
  verbose = FALSE
)
```

Arguments

Y	List of length nT. Data matrices.
F_ls	Covariate matrix or list (see FF).
G_ls	State transition matrix or list (see FF).
W_ls	State noise left-covariance matrix or list (see FF).
V_ls	Observation noise left-covariance matrix or list (see FF).
m0	Numeric matrix. Prior state mean.
M0	Numeric matrix. Prior state left-covariance.
n0	Numeric scalar. Prior shape of σ^2.
D0	Numeric scalar. Prior rate of σ^2.
nT	Integer. Number of time steps.
R	Numeric matrix. Fixed spatial correlation matrix.
delta	Numeric scalar. Discount factor. Defaults to 1.0.
verbose	Logical. If TRUE, emits progress messages. Defaults to FALSE.

Value

A named list with `ff` (output of `FF_sigma2R`) and `bs` (output of `BS`).

See Also

`FF_sigma2R`, `BS`, `FFBS_sampling_sigma2R`

`gen_calibrate_data` *Generate synthetic calibration data with correlated discrepancy*

Description

Simulates observed calibration data z_t by adding a spatially correlated random discrepancy u_t and an independent noise term to the computer model output y_t. The variance τ_t^2 decays over time via the discount factor b.

Usage

```
gen_calibrate_data(ycal_mat, para_gen_cal, U_gen)
```

Arguments

`ycal_mat`	Numeric matrix of dimension `c(t_cal, s_cal)`. Computer model output at calibration locations and time steps.
`para_gen_cal`	Named list with elements:
	b Numeric scalar. Discount factor in $(0, 1)$.
	n0 Numeric scalar. Prior shape of τ_0^2.
	d0 Numeric scalar. Prior scale of τ_0^2.
	u0 Numeric vector of length `s_cal`. Initial discrepancy.
`U_gen`	Numeric matrix. Spatial covariance matrix for the discrepancy (`s_cal` x `s_cal`).

Value

A named list with:

u Matrix of dimension `c(t_cal, s_cal)` of discrepancy realisations.

z Matrix of dimension `c(t_cal, s_cal)` of simulated observations.

tau2 Numeric vector of length `t_cal` with sampled variances.

See Also

`gen_prior_u_tau2`

gen_calibrate_data_uncorr *Generate synthetic calibration data with uncorrelated discrepancy*

Description

Like gen_calibrate_data but uses independent (uncorrelated) discrepancy increments drawn from a univariate normal with variance $\tau_t^2 \cdot U_{11}$.

Usage

```
gen_calibrate_data_uncorr(ycal_mat, para_gen_cal, U_gen)
```

Arguments

ycal_mat	Numeric matrix. Computer model output (t_cal x s_cal).
para_gen_cal	Named list (same structure as in gen_calibrate_data).
U_gen	Numeric matrix. Only the [1,1] entry is used as the marginal variance scale.

Value

A named list with:

z Matrix of dimension c(t_cal, s_cal) of simulated observations.

tau2 Numeric vector of length t_cal.

See Also

gen_calibrate_data

gen_exp_kernel *Compute an exponential GP kernel matrix*

Description

Evaluates the exponential kernel $k(s, s') = \sigma^2 \exp(-\phi\|s - s'\|) + \tau^2 \mathbf{1}_{s=s'}$ for all pairs of rows in loc.

Usage

```
gen_exp_kernel(loc, phi, sigma2 = 1, tau2 = 0)
```

Arguments

`loc`	Numeric matrix. Spatial locations ($n \times d$).
`phi`	Numeric scalar. Range (decay) parameter.
`sigma2`	Numeric scalar. Marginal variance. Defaults to `1`.
`tau2`	Numeric scalar. Nugget variance. Defaults to `0`.

Value

A symmetric positive definite matrix of dimension `c(n, n)`.

See Also

`gen_gp_kernel`, `gen_expsq_kernel`

`gen_expsq_kernel` *Compute a squared-exponential (Gaussian) GP kernel matrix*

Description

Evaluates the squared-exponential kernel $k(s, s') = \sigma^2 \exp(-\phi \|s - s'\|^2) + \tau^2 \mathbf{1}_{s=s'}$ for all pairs of rows in `loc`.

Usage

```
gen_expsq_kernel(loc, phi, sigma2 = 1, tau2 = 0)
```

Arguments

`loc`	Numeric matrix. Spatial locations ($n \times d$).
`phi`	Numeric scalar. Inverse length-scale parameter.
`sigma2`	Numeric scalar. Marginal variance. Defaults to `1`.
`tau2`	Numeric scalar. Nugget variance. Defaults to `0`.

Value

A symmetric positive definite matrix of dimension `c(n, n)`.

See Also

`gen_gp_kernel`, `gen_exp_kernel`

gen_F_ls_AR1 *Build AR(1) covariate list from a list of response matrices*

Description

Constructs the time-varying covariate list F_t for a first-order autoregressive model: $F_t = Y_{t-1}$ (with $F_1 = Y_1$).

Usage

```
gen_F_ls_AR1(Y, nT)
```

Arguments

Y	List of length nT. Each element is the $N \times S$ data matrix at time t.
nT	Integer. Number of time steps.

Value

A list of length nT where element t is the covariate matrix F_t.

See Also

gen_F_ls_AR2, gen_F_ls_AR1_EP

gen_F_ls_AR1_EP *Build AR(1) covariate list for the episode-block model*

Description

Generates a flattened covariate list for the episode-partition (EP) model with AR(1) lags. The output list is indexed by $(t-1) \times n_b + j$, where j is the block index and t is the time index.

Usage

```
gen_F_ls_AR1_EP(Y, nT, n_b, ind)
```

Arguments

Y	List of length nT. Each element is the full spatial data matrix at time t.
nT	Integer. Number of time steps.
n_b	Integer. Number of spatial blocks.
ind	Data frame of block indices as returned by generate_grid.

Value

A list of length nT * n_b with block-wise AR(1) covariate matrices.

See Also

gen_F_ls_AR1, gen_F_ls_AR2_EP

---

gen_F_ls_AR2 *Build AR(2) covariate list from a list of response matrices*

Description

Constructs the time-varying covariate list F_t for a second-order autoregressive model by column-binding the two most recent lags: $F_t = [Y_{t-2}, Y_{t-1}]$ (with special handling for $t = 1, 2$).

Usage

```
gen_F_ls_AR2(Y, nT)
```

Arguments

Y	List of length nT. Each element is the $N \times S$ data matrix at time t.
nT	Integer. Number of time steps.

Value

A list of length nT where element t is the $N \times 2S$ covariate matrix F_t.

See Also

gen_F_ls_AR1, gen_F_ls_AR2_EP

---

gen_F_ls_AR2_EP *Build AR(2) covariate list for the episode-block model*

Description

Generates a flattened covariate list for the episode-partition (EP) model with AR(2) lags. Both lag-1 and lag-2 column blocks are extracted and column-bound for each spatial block.

Usage

```
gen_F_ls_AR2_EP(Y, nT, n_b, ind)
```

Arguments

Y	List of length nT. Each element is the full spatial data matrix at time t.
nT	Integer. Number of time steps.
n_b	Integer. Number of spatial blocks.
ind	Data frame of block indices as returned by generate_grid.

Value

A list of length nT * n_b with block-wise AR(2) covariate matrices (each of width $2 \times$ bncol).

See Also

gen_F_ls_AR2, gen_F_ls_AR1_EP

---

gen_ffbs_csv *Generate synthetic FFBS data and write to CSV files*

Description

Simulates data from the MNIW dynamic linear model for nT time steps and writes the observations $Y_1, \ldots, Y_T$ and all model parameters to CSV files in path.

Usage

```
gen_ffbs_csv(N, S, p, nT, path)
```

Arguments

N	Integer. Number of spatial locations (rows of Y_t).
S	Integer. Number of response variables (columns of Y_t).
p	Integer. Dimension of the state vector (must be ≥ 2).
nT	Integer. Number of time steps.
path	Character. Directory path (with trailing /) where CSV files are written.

Value

Invisibly returns NULL. Files written: m0.csv, MM0.csv, G0.csv, F0.csv, Sigma.csv, RSigma.csv, V0.csv, RV0.csv, W0.csv, and Y1.csv through Y<nT>.csv.

See Also

gen_ffbs_data

gen_ffbs_data *Generate synthetic FFBS data in memory*

Description

Simulates data from the MNIW dynamic linear model for `nT` time steps and returns the generated observations and model parameters as in-memory objects.

Usage

```
gen_ffbs_data(N, S, p, nT)
```

Arguments

`N`	Integer. Number of spatial locations (rows of Y_t).
`S`	Integer. Number of response variables (columns of Y_t).
`p`	Integer. Dimension of the state vector (must be ≥ 2).
`nT`	Integer. Number of time steps.

Value

A named list with:

Y Named list of length `nT` (`"Y1"`, ..., `"Y<nT>"`), each element an $N \times S$ data matrix.

para Named list of model parameters: `loc`, `dist`, `m0`, `M0`, `G0`, `F0`, `Sigma`, `RSigma`, `V0`, `RV0`, `W0`, `n0`, `D0`.

See Also

`gen_ffbs_csv`

gen_gp_kernel *Compute a Gaussian Process covariance kernel matrix*

Description

Evaluates the isotropic GP exponential kernel $k(s, s') = \sigma^2 \exp(-\phi\|s - s'\|) + \tau^2 \mathbf{1}_{s=s'}$ for all pairs of rows in `loc`.

Usage

```
gen_gp_kernel(loc, phi, sigma2, tau2)
```

Arguments

loc	Numeric matrix. Spatial locations ($n \times d$).
phi	Numeric scalar. Range (decay) parameter of the exponential kernel.
sigma2	Numeric scalar. Marginal variance parameter.
tau2	Numeric scalar. Nugget (white-noise) variance.

Value

A symmetric positive definite matrix of dimension `c(n, n)`.

See Also

gen_exp_kernel, gen_expsq_kernel

---

gen_Jt *Compute the cross-covariance matrix between observed and new locations*

Description

Builds the full covariance matrix over observed and new locations using an exponential GP kernel, then extracts the $N \times N_{\text{new}}$ cross-covariance block J_t.

Usage

```
gen_Jt(input, input_new, phi_para, gp_sigma2 = 1.1, gp_tau2 = 10^(-4))
```

Arguments

input	Numeric matrix. Coordinates of observed locations ($N \times d$).
input_new	Numeric matrix or vector. Coordinates of new locations ($N_{\text{new}} \times d$).
phi_para	Numeric scalar. Range parameter of the exponential kernel.
gp_sigma2	Numeric scalar. Marginal variance of the kernel. Defaults to `1.1`.
gp_tau2	Numeric scalar. Nugget variance. Defaults to `1e-4`.

Value

Numeric matrix of dimension `c(N, N_new)`.

See Also

gen_exp_kernel

gen_pd_matrix *Generate a random positive definite matrix*

Description

Constructs a positive definite matrix of dimension `dim` by forming $A^\top A$ from a random normal matrix A and then rescaling so the maximum entry is 1.

Usage

```
gen_pd_matrix(dim)
```

Arguments

`dim`	Integer. Number of rows (and columns) of the output matrix.

Value

A symmetric positive definite numeric matrix of dimension `c(dim, dim)`.

gen_pde *Simulate a spatially extended SIR PDE model*

Description

Numerically integrates the spatial SIR ODE (`SIR`) on a two-dimensional grid using `deSolve::ode.2D` and returns the log(1 + I) values of the infected compartment.

Usage

```
gen_pde(eta, Nx, Ny, N, nT_ori)
```

Arguments

`eta`	Numeric vector of length 5. PDE parameters: transmission rate, recovery rate, and diffusion coefficients for S, I, R.
`Nx`	Integer. Number of grid cells in the x-direction (rows).
`Ny`	Integer. Number of grid cells in the y-direction (columns).
`N`	Numeric. Total population size.
`nT_ori`	Integer. Number of time steps to simulate (times $0, 1, \ldots,$ nT_ori $- 1$).

Value

Numeric matrix of dimension `c(nT_ori, Nx * Ny)` containing $\log(1 + I_t)$ for each time step and grid cell.

See Also

SIR, update_y_eta

---

gen_prior_u_tau2 *Sample prior discrepancy trajectory and variance sequence*

Description

Simulates a prior trajectory $u_1, \ldots, u_T$ of the spatial discrepancy field and the associated inverse-gamma variance sequence $\tau_1^2, \ldots, \tau_T^2$ from the dynamic calibration model.

Usage

```
gen_prior_u_tau2(n0, d0, b, m0, M0, U, nT_cal)
```

Arguments

n0	Numeric scalar. Prior shape of τ_0^2.
d0	Numeric scalar. Prior scale of τ_0^2.
b	Numeric scalar. Discount factor in $(0, 1)$.
m0	Numeric vector. Prior mean of the initial discrepancy u_0.
M0	Numeric matrix. Prior covariance of u_0.
U	Numeric matrix. Spatial covariance matrix for the discrepancy increments.
nT_cal	Integer. Number of calibration time steps.

Value

A named list with:

u0 Numeric vector. Sampled initial discrepancy.

u Matrix of dimension `c(nT_cal, s_cal)` of discrepancy realisations.

tau2_0 Numeric scalar. Sampled τ_0^2.

tau2 Numeric vector of length `nT_cal`.

See Also

gen_calibrate_data

gen_ran_matrix *Generate a random matrix with entries scaled to* $[-1, 1]$

Description

Draws a matrix of independent standard normal entries and divides by the absolute maximum, so all entries lie in $[-1, 1]$.

Usage

```
gen_ran_matrix(nrow, ncol)
```

Arguments

nrow	Integer. Number of rows.
ncol	Integer. Number of columns.

Value

A numeric matrix of dimension `c(nrow, ncol)`.

generate.grid.exact *Generate an exact block grid analytically*

Description

Computes block indices when block dimensions divide file dimensions exactly. Uses vectorised operations rather than a loop, making it fast for large grids.

Usage

```
generate.grid.exact(fnrow, fncol, bnrow, bncol, traversal.mode)
```

Arguments

fnrow	Integer. Number of rows in the file or matrix.
fncol	Integer. Number of columns in the file or matrix.
bnrow	Integer. Number of rows per block. Must divide `fnrow` exactly.
bncol	Integer. Number of columns per block. Must divide `fncol` exactly.
traversal.mode	Character. `"rowsnake"` or `"lr"`. See `generate_grid` for details.

Value

A `data.frame` with columns `block_no`, `U`, `D`, `L`, `R`. See `generate_grid` for column descriptions.

See Also

generate_grid, generate.grid.rowsnake, generate.grid.lr

`generate.grid.lr` *Generate a flexible block grid with left-to-right traversal*

Description

Iteratively computes block indices using a strict left-to-right traversal order across all rows. Handles cases where block dimensions do not evenly divide file dimensions by allowing the last block in each row or column to be smaller.

Usage

```
generate.grid.lr(fnrow, fncol, bnrow, bncol)
```

Arguments

`fnrow`	Integer. Number of rows in the file or matrix.
`fncol`	Integer. Number of columns in the file or matrix.
`bnrow`	Integer. Number of rows per block.
`bncol`	Integer. Number of columns per block.

Value

A `data.frame` with columns `block_no`, `U`, `D`, `L`, `R`. See generate_grid for column descriptions.

See Also

generate_grid, generate.grid.exact, generate.grid.rowsnake

`generate.grid.rowsnake`
Generate a flexible block grid with snake traversal

Description

Iteratively computes block indices using a snake (boustrophedon) traversal order. Handles cases where block dimensions do not evenly divide file dimensions by allowing the last block in each row or column to be smaller.

Usage

```
generate.grid.rowsnake(fnrow, fncol, bnrow, bncol)
```

Arguments

fnrow	Integer. Number of rows in the file or matrix.
fncol	Integer. Number of columns in the file or matrix.
bnrow	Integer. Number of rows per block.
bncol	Integer. Number of columns per block.

Value

A data.frame with columns block_no, U, D, L, R. See generate_grid for column descriptions.

See Also

generate_grid, generate.grid.exact, generate.grid.lr

---

generate_grid *Generate block indices for big data grid traversal*

Description

Partitions a matrix or file into rectangular blocks and returns a data frame of block indices numbered from 1 to K (total number of blocks). Supports exact and flexible partitioning, and multiple traversal orders.

Usage

```
generate_grid(
  fnrow,
  fncol,
  bnrow,
  bncol,
  traversal.mode = "rowsnake",
  is.flexible = NULL,
  is.flexible.row = FALSE,
  is.flexible.col = FALSE
)
```

Arguments

fnrow	Integer. Number of rows in the file or matrix.
fncol	Integer. Number of columns in the file or matrix.
bnrow	Integer. Number of rows per block.
bncol	Integer. Number of columns per block.
traversal.mode	Character. Traversal order for the grid. "rowsnake" uses alternating left-right traversal (snake/boustrophedon order); "lr" uses strict left-to-right traversal throughout. Defaults to "rowsnake".

`is.flexible` Logical or `NULL`. If non-`NULL`, overrides both `is.flexible.row` and `is.flexible.col`. When `TRUE`, allows block dimensions that do not evenly divide the file dimensions. Defaults to `NULL`.

`is.flexible.row` Logical. If `FALSE`, an error is thrown when `bnrow` does not evenly divide `fnrow`. Defaults to `FALSE`.

`is.flexible.col` Logical. If `FALSE`, an error is thrown when `bncol` does not evenly divide `fncol`. Defaults to `FALSE`.

Value

A `data.frame` with columns:

block_no Block index from 1 to K.

U Upper (first) row index of the block.

D Lower (last) row index of the block.

L Left (first) column index of the block.

R Right (last) column index of the block.

Examples

```
# Exact partition with snake traversal
generate_grid(100, 100, 10, 10)

# Flexible partition (block size does not divide file size evenly)
generate_grid(105, 103, 10, 10, is.flexible = TRUE)
```

---

inv_chol *Invert a matrix via its Cholesky factorisation*

Description

Computes X^{-1} using `chol` followed by `chol2inv` for numerical stability.

Usage

```
inv_chol(X)
```

Arguments

`X` Numeric symmetric positive definite matrix.

Value

The inverse of `X`.

lppd_id_1t — *One-step log posterior predictive density (identity right-covariance model)*

Description

Computes the log posterior predictive density of Y_t at a single time step under the model with an identity right-covariance matrix, using a matrix-normal density.

Usage

```
lppd_id_1t(Yt, Ft, Vt, st, St)
```

Arguments

Yt	Numeric matrix. Observed data at time t ($N \times S$).
Ft	Numeric matrix. Covariate matrix at time t ($N \times p$).
Vt	Numeric matrix. Observation noise left-covariance ($N \times N$).
st	Numeric matrix. Smoothed state mean at time t ($p \times S$).
St	Numeric matrix. Smoothed state left-covariance at time t ($p \times p$).

Value

Numeric scalar. Log posterior predictive density.

See Also

lppd_IW_1t, lppd_IG_1t

lppd_IG_1t — *One-step log posterior predictive density (scalar sigma-squared-times-R model)*

Description

Computes the log posterior predictive density of Y_t at a single time step under the scalar-sigma model ($\sigma^2 \sim IG$), using dMTig.

Usage

```
lppd_IG_1t(Yt, Ft, Vt, st, St, nt, Dt, R)
```

Arguments

Yt	Numeric matrix. Observed data at time t ($N \times S$).
Ft	Numeric matrix. Covariate matrix at time t ($N \times p$).
Vt	Numeric matrix. Observation noise left-covariance ($N \times N$).
st	Numeric matrix. Smoothed state mean at time t ($p \times S$).
St	Numeric matrix. Smoothed state left-covariance at time t ($p \times p$).
nt	Numeric scalar. Filtered shape parameter at time t.
Dt	Numeric scalar. Filtered rate parameter at time t.
R	Numeric matrix. Fixed right correlation matrix ($S \times S$).

Value

Numeric scalar. Log posterior predictive density.

See Also

dMTig, lppd_IW_1t

---

lppd_IW_1t *One-step log posterior predictive density (MNIW / inverse-Wishart model)*

Description

Computes the log posterior predictive density of Y_t at a single time step under the MNIW model, evaluated at the smoothed state parameters from FFBS.

Usage

```
lppd_IW_1t(Yt, Ft, Vt, st, St, nt, Dt)
```

Arguments

Yt	Numeric matrix. Observed data at time t ($N \times S$).
Ft	Numeric matrix. Covariate matrix at time t ($N \times p$).
Vt	Numeric matrix. Observation noise left-covariance ($N \times N$).
st	Numeric matrix. Smoothed state mean at time t ($p \times S$).
St	Numeric matrix. Smoothed state left-covariance at time t ($p \times p$).
nt	Numeric scalar. Filtered degrees of freedom at time t.
Dt	Numeric matrix. Filtered scale matrix at time t ($S \times S$).

Value

Numeric scalar. Log posterior predictive density.

See Also

lppd_IG_1t, lppd_id_1t

`make_pds` *Force a matrix to be positive definite and symmetric*

Description

Symmetrises a matrix by averaging it with its transpose and, if the result is not positive definite, adds a small diagonal ridge.

Usage

```
make_pds(C, eps = 10^(-4))
```

Arguments

`C`	Numeric matrix.
`eps`	Numeric scalar. Size of the diagonal ridge added when `C` is not positive definite. Defaults to `1e-4`.

Value

The corrected positive definite symmetric matrix.

See Also

check_pds

`MNIG_sampler` *Sample from the Matrix Normal Inverse Gamma (MNIG) distribution*

Description

Draws `nsam` posterior samples of the regression coefficient matrix B and the scalar variance σ^2 under the MNIG model where the right covariance of $B|\sigma^2$ is $\sigma^2 R$:

$$\sigma^2 \sim IG(v, S), \quad B|\sigma^2 \sim MN(C, V_b, \sigma^2 R).$$

Usage

```
MNIG_sampler(nsam, X, v, S, C, Vb, R)
```

Arguments

nsam	Integer. Number of samples to draw.
X	Numeric matrix. Covariate (design) matrix ($n \times p$).
v	Numeric scalar. Shape parameter of the inverse-gamma prior on σ^2.
S	Numeric scalar. Rate parameter of the inverse-gamma prior on σ^2.
C	Numeric matrix. Prior mean of $B\|\sigma^2$ ($p \times q$).
Vb	Numeric matrix. Prior row covariance of $B\|\sigma^2$ ($p \times p$, positive definite).
R	Numeric matrix. Fixed right correlation matrix ($q \times q$, positive definite).

Value

A named list with:

sigma2 Numeric vector of length nsam with samples of σ^2.

B Array of dimension c(p, q, nsam) with samples of B.

See Also

MNIW_sampler, FFBS_sampling_sigma2R

MNIW_R *MNIW posterior update*

Description

Computes the posterior parameters of the Matrix Normal Inverse Wishart (MNIW) model using crossprod for improved numerical efficiency.

Usage

```
MNIW_R(X, Y, newX = NULL, newY = NULL, H, v, S, C, Vb)
```

Arguments

X	Numeric matrix. Covariate (design) matrix of dimension $n \times p$.
Y	Numeric matrix. Response matrix of dimension $n \times q$.
newX	Unused legacy parameter. Defaults to none.
newY	Unused legacy parameter. Defaults to none.
H	Numeric matrix. Row covariance matrix of the error term ($n \times n$).
v	Numeric scalar. Prior degrees of freedom of the inverse-Wishart on Σ.
S	Numeric matrix. Prior scale matrix of the inverse-Wishart on Σ.
C	Numeric matrix. Prior mean matrix of $B\|\Sigma$ ($p \times q$).
Vb	Numeric matrix. Prior row covariance matrix of $B\|\Sigma$ ($p \times p$).

Details

The model is $Y = XB+E$, where $E \sim MN(0, H, \Sigma)$, $\Sigma \sim IW(v, S)$, and $B|\Sigma \sim MN(C, V_b, \Sigma)$.

Value

A named list with posterior parameters:

vnew Posterior degrees of freedom.

Snew Posterior scale matrix of the inverse-Wishart.

Cnew Posterior mean matrix of $B|\Sigma$.

Vbnew Posterior row covariance matrix of $B|\Sigma$.

See Also

MNIW_R_naiive

MNIW_R_naiive *Naive MNIW posterior update*

Description

Computes the posterior parameters of the Matrix Normal Inverse Wishart (MNIW) model using explicit matrix transposes. Equivalent to MNIW_R but without crossprod optimisations. Intended for reference and testing.

Usage

```
MNIW_R_naiive(X, Y, newX = NULL, newY = NULL, H, v, S, C, Vb)
```

Arguments

X	Numeric matrix. Covariate (design) matrix of dimension $n \times p$.
Y	Numeric matrix. Response matrix of dimension $n \times q$.
newX	Unused legacy parameter. Defaults to none.
newY	Unused legacy parameter. Defaults to none.
H	Numeric matrix. Row covariance matrix of the error term ($n \times n$).
v	Numeric scalar. Prior degrees of freedom of the inverse-Wishart on Σ.
S	Numeric matrix. Prior scale matrix of the inverse-Wishart on Σ.
C	Numeric matrix. Prior mean matrix of $B|\Sigma$ ($p \times q$).
Vb	Numeric matrix. Prior row covariance matrix of $B|\Sigma$ ($p \times p$).

Details

The model is $Y = XB+E$, where $E \sim MN(0, H, \Sigma)$, $\Sigma \sim IW(v, S)$, and $B|\Sigma \sim MN(C, V_b, \Sigma)$.

Value

A named list with posterior parameters:

vnew Posterior degrees of freedom.

Snew Posterior scale matrix of the inverse-Wishart.

Cnew Posterior mean matrix of $B|\Sigma$.

Vbnew Posterior row covariance matrix of $B|\Sigma$.

See Also

MNIW_R

`MNIW_sampler` *Sample from the Matrix Normal Inverse Wishart (MNIW) distribution*

Description

Draws `nsam` posterior samples of the regression coefficient matrix B and the covariance matrix Σ under the MNIW model:

$$Y = XB + E, \quad E \sim MN(0, H, \Sigma)$$
$$\Sigma \sim IW(v, S), \quad B|\Sigma \sim MN(C, V_b, \Sigma).$$

Usage

```
MNIW_sampler(nsam, X, v, S, C, Vb)
```

Arguments

`nsam`	Integer. Number of samples to draw.
`X`	Numeric matrix. Covariate (design) matrix ($n \times p$).
`v`	Numeric scalar. Degrees of freedom of the inverse-Wishart prior on Σ.
`S`	Numeric matrix. Scale matrix of the inverse-Wishart prior ($q \times q$, positive definite).
`C`	Numeric matrix. Prior mean of $B|\Sigma$ ($p \times q$).
`Vb`	Numeric matrix. Prior row covariance of $B|\Sigma$ ($p \times p$, positive definite).

Value

A named list with:

sigma Array of dimension `c(q, q, nsam)` with samples of Σ.

B Array of dimension `c(p, q, nsam)` with samples of B.

See Also

MNIG_sampler, FFBS_sampling

`plot_panel_heatmap_9` *Plot a 3-by-3 panel of heatmaps across selected time stamps*

Description

Produces nine raster heatmaps arranged in a 3-by-3 grid, one for each time stamp in `tstamp`. Each panel shows the spatial field for a given input (row) index and time step.

Usage

```
plot_panel_heatmap_9(
  dat,
  input_num,
  tstamp,
  max_y,
  Nx,
  Ny,
  nT,
  filename = "plot_panel",
  savei = TRUE,
  col_bgr = spdbl_default_col_bgr(),
  path_fig = "."
)
```

Arguments

`dat`	List of length $\geq \max($`tstamp`$)$. Each element is a matrix of dimension `c(n_inputs, Nx * Ny)`.
`input_num`	Integer. Row index within each time-step matrix to plot.
`tstamp`	Integer vector of length 9. Time step indices to display.
`max_y`	Numeric scalar. Upper limit of the colour scale.
`Nx`	Integer. Number of grid points in the x-direction.
`Ny`	Integer. Number of grid points in the y-direction.
`nT`	Integer. Total number of time steps in `dat`.
`filename`	Character. Base name for the saved PNG (used when `savei = TRUE`). Defaults to `"plot_panel"`.
`savei`	Logical. Whether to save the panel to disk. Defaults to `TRUE`.
`col_bgr`	Character vector of colours for the heatmap gradient.
`path_fig`	Character. Directory path where the file is saved when `savei = TRUE`. Defaults to the current working directory.

Value

A list of nine `ggplot` objects (one per panel).

plot_panel_heatmap_9_cal

Plot a 3-by-3 panel of calibration heatmaps

Description

Produces nine raster heatmaps for calibration data arranged in a 3-by-3 grid, one for each time stamp in `tstamp`. Locations are specified via `loc_cal` rather than a regular grid.

Usage

```
plot_panel_heatmap_9_cal(
  dat,
  tstamp,
  max_y,
  loc_cal,
  Nx,
  Ny,
  filename = "plot_panel",
  savei = TRUE,
  col_bgr = spdbl_default_col_bgr(),
  path_fig = "."
)
```

Arguments

`dat`	Numeric matrix of dimension `c(nT, n_locations)`. Rows are time steps; columns are calibration locations.
`tstamp`	Integer vector of length 9. Time-step indices (rows of `dat`) to display.
`max_y`	Numeric scalar. Upper limit of the colour scale.
`loc_cal`	Data frame with columns `row` and `col` giving the spatial coordinates of calibration locations.
`Nx`	Integer. Number of grid points in the x-direction (used for reference only).
`Ny`	Integer. Number of grid points in the y-direction (used for reference only).
`filename`	Character. Base name for the saved PNG. Defaults to `"plot_panel"`.
`savei`	Logical. Whether to save the panel to disk. Defaults to `TRUE`.
`col_bgr`	Character vector of colours for the heatmap gradient.
`path_fig`	Character. Directory path where the file is saved when `savei = TRUE`. Defaults to the current working directory.

Value

A list of nine `ggplot` objects.

See Also

plot_panel_heatmap_9_cal_nolab

---

plot_panel_heatmap_9_cal_nolab *Plot a 3-by-3 panel of calibration heatmaps without axis labels*

Description

Like plot_panel_heatmap_9_cal but suppresses x- and y-axis labels and supports a custom lower colour-scale limit (min_y).

Usage

```
plot_panel_heatmap_9_cal_nolab(
  dat,
  tstamp,
  min_y = 0,
  max_y,
  loc_cal,
  Nx,
  Ny,
  filename = "plot_panel",
  savei = TRUE,
  col_bgr = spdbl_default_col_bgr(),
  path_fig = "."
)
```

Arguments

dat	Numeric matrix of dimension c(nT, n_locations).
tstamp	Integer vector of length 9. Time-step indices to display.
min_y	Numeric scalar. Lower limit of the colour scale. Defaults to 0.
max_y	Numeric scalar. Upper limit of the colour scale.
loc_cal	Data frame with columns row and col.
Nx	Integer. Number of grid points in the x-direction.
Ny	Integer. Number of grid points in the y-direction.
filename	Character. Base name for the saved PNG. Defaults to "plot_panel".
savei	Logical. Whether to save the panel to disk. Defaults to TRUE.
col_bgr	Character vector of colours for the heatmap gradient.
path_fig	Character. Directory path where the file is saved when savei = TRUE. Defaults to the current working directory.

Value

A list of nine ggplot objects.

See Also

plot_panel_heatmap_9_cal

`prepare_data` *Prepare PDE emulator training and testing data from CSV files*

Description

Reads a matrix of PDE input parameters and a set of PDE output CSV files, splits the inputs into training and testing sets, and reshapes the simulator outputs into time-indexed lists for emulator fitting and evaluation.

Usage

```
prepare_data(file_para, file_data, prop_train = 0.8)
```

Arguments

`file_para`	Character scalar. Path to a CSV file containing the PDE input parameter matrix. Rows correspond to simulator runs.
`file_data`	Character vector. Paths to CSV files containing PDE outputs, one file per simulator run. The length should match the number of rows in `file_para`.
`prop_train`	Numeric scalar in $(0, 1)$. Proportion of simulator runs used for training. Defaults to `0.8`.

Details

The function assumes that each file in `file_data` contains a matrix whose rows correspond to time steps and whose columns correspond to spatial locations. The number of time steps is inferred from the first PDE output file.

Value

A named list with:

pde_para_train Numeric matrix of training input parameters.

pde_para_test Numeric matrix of testing input parameters.

dt_pde_train Named list of training PDE outputs by time step.

dt_pde_test Named list of testing PDE outputs by time step.

n_train Integer. Number of training simulator runs.

n_test Integer. Number of testing simulator runs.

See Also

emulator_learn, emulator_predict

---

quick_heat *Quick raster heatmap*

Description

Creates a ggplot2 raster heatmap using the col_bgr colour palette with a squished colour scale capped at max_y.

Usage

```
quick_heat(dt, max_y, col_bgr = spdbl_default_col_bgr())
```

Arguments

dt	Data frame with columns x (horizontal position), y (vertical position), and fill (value to display).
max_y	Numeric scalar. Upper limit of the colour scale. Values above this are squished to the maximum colour.
col_bgr	Character vector of colours for the heatmap gradient.

Value

A ggplot object.

See Also

quick_save

---

quick_save *Save a ggplot to a timestamped PNG file*

Description

Saves plot to a PNG file in path_fig whose name is filename followed by the current Unix timestamp, ensuring unique file names across repeated calls.

Usage

```
quick_save(filename, path_fig, plot)
```

Arguments

filename	Character. Base name for the output file (no extension).
path_fig	Character. Directory path where the file is saved.
plot	A ggplot object to save.

Value

Invisibly returns NULL (called for its side effect).

See Also

quick_heat

---

read_big_csv_quick *Read a rectangular block from a large CSV file*

Description

Efficiently reads a contiguous rectangular block of rows and columns from a CSV file using readr::read_csv. Useful for processing big data in chunks without loading the entire file into memory.

Usage

```
read_big_csv_quick(filename, rows = c(1, Inf), cols = c(1, Inf), header = TRUE)
```

Arguments

filename	Character. Path to the CSV file.
rows	Integer vector of length 2 giving the first and last row indices to read (1-based, excluding the header row if header = TRUE). Use c(1, Inf) to read all rows. Defaults to c(1, Inf).
cols	Integer vector of length 2 giving the first and last column indices to read. Use c(1, Inf) to read all columns. Defaults to c(1, Inf).
header	Logical. Whether the file has a header row. Defaults to TRUE.

Value

A tibble (from readr) containing the requested block.

recover_from_EP_exact *Recover episode-partitioned data to original time dimension (exact)*

Description

Reassembles a flattened episode-partitioned list (as produced by the EP model) back into a list indexed by the original `nT_ori` time steps. Assumes the number of blocks per season `n_block = nT / nT_ori` is an integer.

Usage

```
recover_from_EP_exact(dat_EP, nT_ori, nT)
```

Arguments

`dat_EP`	List of length `nT`. Flattened episode-partitioned data.
`nT_ori`	Integer. Number of original (non-partitioned) time steps.
`nT`	Integer. Total number of episode time steps (`nT = nT_ori * n_block`).

Value

A named list of length `nT_ori` (`"T1"`, ..., `"T<nT_ori>"`), each element a matrix covering one original time step.

See Also

`recover_from_EP_MC`

recover_from_EP_MC *Recover episode-partitioned posterior samples to original time dimension*

Description

Reassembles a flattened episode-partitioned list of posterior sample arrays back into a list indexed by the original `nT_ori` time steps.

Usage

```
recover_from_EP_MC(dat_EP, nT_ori, nT, nsam)
```

Arguments

`dat_EP`	List of length `nT`. Each element is an array of dimension `c(p, bncol, nsam)`.
`nT_ori`	Integer. Number of original time steps.
`nT`	Integer. Total number of episode time steps.
`nsam`	Integer. Number of posterior samples.

Value

A named list of length `nT_ori`. Each element is an array of dimension `c(p, bncol * n_block, nsam)`.

See Also

`recover_from_EP_exact`

`rmn_chol` *Draw one sample from a matrix-normal distribution (Cholesky parameterisation)*

Description

Samples $\Theta \sim MN(m, U, \Sigma)$ using the upper-triangular Cholesky factors `RM` and `RSigma` of U and Σ respectively: $\Theta = m + \mathrm{RM}^\top Z\,\mathrm{RSigma}$, where Z has i.i.d. standard normal entries.

Usage

```
rmn_chol(m, RM, RSigma)
```

Arguments

`m`	Numeric matrix. Mean matrix ($p \times S$).
`RM`	Numeric matrix. Upper-triangular Cholesky factor of the row covariance U ($p \times p$).
`RSigma`	Numeric matrix. Upper-triangular Cholesky factor of the column covariance Σ ($S \times S$).

Value

A numeric matrix of dimension `c(p, S)`.

See Also

`rmn_chol_more`

rmn_chol_more — *Draw multiple samples from a matrix-normal distribution (Cholesky parameterisation)*

Description

Draws `nsam` independent samples from $MN(m, U, \Sigma)$ using the Cholesky factors of U and Σ.

Usage

```
rmn_chol_more(nsam, m, RM, RSigma)
```

Arguments

`nsam`	Integer. Number of samples to draw.
`m`	Numeric matrix. Mean matrix ($p \times S$).
`RM`	Numeric matrix. Upper-triangular Cholesky factor of the row covariance U ($p \times p$).
`RSigma`	Numeric matrix. Upper-triangular Cholesky factor of the column covariance Σ ($S \times S$).

Value

An array of dimension `c(p, S, nsam)`.

See Also

`rmn_chol`

sample_y_eta_one — *Draw predictive samples from a precomputed mean and covariance*

Description

Draws one sample from the predictive distribution $y_t \sim N(\mu_t, \Sigma)$ at each calibration time step, given precomputed mean and covariance (e.g., from `update_muSigma_eta_one`).

Usage

```
sample_y_eta_one(ind_sam, nT_cal, N_sp_train, mu_arry_train, Sigma_eta_train)
```

Arguments

`ind_sam`	Integer. Sample index (used only for loop iteration; single sample is drawn).
`nT_cal`	Integer. Number of calibration time steps.
`N_sp_train`	Integer. Number of training spatial locations.
`mu_arry_train`	Numeric array of dimension `c(nT_cal, N_sp_train)`. Predictive means.
`Sigma_eta_train`	Numeric matrix. Predictive covariance (`N_sp_train` x `N_sp_train`).

Value

Numeric matrix of dimension `c(nT_cal, N_sp_train)` with one draw per time step.

See Also

`update_muSigma_eta_one`

`scale_back_uniform` *Invert a uniform scaling transformation*

Description

Maps `x` from the unit interval back to the original range $[\text{low}, \text{high}]$: $x \cdot (\text{high} - \text{low}) + \text{low}$.

Usage

```
scale_back_uniform(x, low, high)
```

Arguments

`x`	Numeric vector or matrix. Values in $[0, 1]$.
`low`	Numeric. Lower bound of the target range.
`high`	Numeric. Upper bound of the target range.

Value

Numeric vector or matrix rescaled to $[\text{low}, \text{high}]$.

See Also

`scale_uniform`

scale_uniform *Scale a vector to the unit interval via a uniform transformation*

Description

Maps `x` to $(x - \text{low})/(\text{high} - \text{low})$. Raises an error if `x` is outside `[low, high]`. Sets `NaN` results to 0.

Usage

```
scale_uniform(x, low, high)
```

Arguments

`x`	Numeric vector or matrix. Values to scale.
`low`	Numeric. Lower bound of the original range.
`high`	Numeric. Upper bound of the original range.

Value

Numeric vector or matrix with values in $[0, 1]$.

See Also

`scale_back_uniform`

SIR *Right-hand side of the spatially extended SIR ODE*

Description

Defines the right-hand side of a spatial SIR (Susceptible-Infected-Recovered) partial differential equation suitable for use with `deSolve::ode.2D`. Diffusion is implemented via `ReacTran::tran.2D` with zero-flux boundary conditions.

Usage

```
SIR(t, y, parms)
```

Arguments

`t`	Numeric. Current time (passed by the ODE solver).
`y`	Numeric vector. Flattened state vector of length $3N_xN_y$: S values, then I values, then R values.
`parms`	Named list of parameters:
	Nx Integer. Number of grid cells in the x-direction.
	Ny Integer. Number of grid cells in the y-direction.
	N Numeric. Total population size.
	eta Numeric vector of length 5: `eta[1]` transmission rate, `eta[2]` recovery rate, `eta[3]` diffusion coefficient for S, `eta[4]` diffusion coefficient for I, `eta[5]` diffusion coefficient for R.
	Gridx 1D grid object for x (from `ReacTran::setup.grid.1D`).
	Gridy 1D grid object for y.

Value

A list with one element: a numeric vector of derivatives of the same length as `y`, as required by `deSolve`.

See Also

`gen_pde`

`update_muSigma_eta_one` *Compute posterior predictive mean and covariance without sampling (single sample)*

Description

Like `update_y_eta_one` but skips the final sampling step, returning only the predictive mean and covariance. Useful when sampling is done separately (e.g., in `sample_y_eta_one`).

Usage

```
update_muSigma_eta_one(
  eta,
  ind_sam,
  Nx,
  Ny,
  N_people,
  nT_ori,
  nT,
  nT_cal,
  N_sp,
```

```
  N_sp_train,
  AR_choice,
  bncol,
  res_ffbs,
  n_b_cal,
  ind_block_cal,
  t_VinvY_FTheta_ls,
  Vinv,
  R_cal_train,
  input,
  input_cal,
  phi_para,
  gp_sigma2,
  gp_tau2,
  ind_train_cal
)
```

Arguments

`eta`	Numeric vector. PDE parameters (length 5).
`ind_sam`	Integer. Index of the FFBS sample to use.
`Nx`	Integer. Grid cells in x-direction.
`Ny`	Integer. Grid cells in y-direction.
`N_people`	Numeric. Total population size.
`nT_ori`	Integer. Number of original time steps.
`nT`	Integer. Number of episode time steps.
`nT_cal`	Integer. Number of calibration time steps.
`N_sp`	Integer. Total number of spatial locations.
`N_sp_train`	Integer. Number of training spatial locations.
`AR_choice`	Integer. AR order (1 or 2).
`bncol`	Integer. Number of columns per block.
`res_ffbs`	List. FFBS posterior samples.
`n_b_cal`	Integer. Number of calibration blocks.
`ind_block_cal`	Data frame. Block indices for calibration.
`t_VinvY_FTheta_ls`	List. Precomputed $V^{-1}(Y - F\Theta)$ terms.
`Vinv`	Numeric matrix. Inverse of the observation covariance.
`R_cal_train`	Numeric matrix. Correlation matrix for training locations.
`input`	Numeric matrix. Observed location coordinates.
`input_cal`	Numeric matrix. Calibration location coordinates.
`phi_para`	Numeric scalar. GP range parameter.
`gp_sigma2`	Numeric scalar. GP marginal variance.
`gp_tau2`	Numeric scalar. GP nugget variance.
`ind_train_cal`	Integer vector. Indices of training locations.

Value

A named list with `mu` and `Sigma`.

See Also

`update_y_eta_one`, `sample_y_eta_one`

`update_y_eta` *Update the likelihood of observations given PDE parameters (Monte Carlo)*

Description

For a given candidate PDE parameter vector `eta`, simulates the computer model output, computes the posterior predictive mean and covariance at calibration locations using all `nsam` FFBS samples, and draws new observations from the resulting predictive distribution.

Usage

```
update_y_eta(
  eta,
  nsam,
  Nx,
  Ny,
  N_people,
  nT_ori,
  nT,
  nT_cal,
  N_sp,
  N_sp_train,
  AR_choice,
  bncol,
  res_ffbs,
  n_b_cal,
  ind_block_cal,
  t_VinvY_FTheta_ls,
  Vinv,
  R_cal_train,
  input,
  input_cal,
  phi_para,
  gp_sigma2,
  gp_tau2,
  ind_train_cal
)
```

Arguments

`eta`	Numeric vector. PDE parameters (length 5).
`nsam`	Integer. Number of posterior samples.
`Nx`	Integer. Grid cells in x-direction.
`Ny`	Integer. Grid cells in y-direction.
`N_people`	Numeric. Total population size.
`nT_ori`	Integer. Number of original time steps.
`nT`	Integer. Number of episode time steps.
`nT_cal`	Integer. Number of calibration time steps.
`N_sp`	Integer. Total number of spatial locations.
`N_sp_train`	Integer. Number of training spatial locations.
`AR_choice`	Integer. AR order (1 or 2).
`bncol`	Integer. Number of columns per block.
`res_ffbs`	List. FFBS posterior samples.
`n_b_cal`	Integer. Number of calibration blocks.
`ind_block_cal`	Data frame. Block indices for calibration.
`t_VinvY_FTheta_ls`	List. Precomputed $V^{-1}(Y - F\Theta)$ terms.
`Vinv`	Numeric matrix. Inverse of the observation covariance.
`R_cal_train`	Numeric matrix. Correlation matrix for training locations.
`input`	Numeric matrix. Observed location coordinates.
`input_cal`	Numeric matrix. Calibration location coordinates.
`phi_para`	Numeric scalar. GP range parameter.
`gp_sigma2`	Numeric scalar. GP marginal variance.
`gp_tau2`	Numeric scalar. GP nugget variance.
`ind_train_cal`	Integer vector. Indices of training locations within calibration locations.

Value

A named list with:

y Array `c(nT_cal, N_sp_train, nsam)` of predictive samples.

mu Array `c(nT_cal, N_sp_train, nsam)` of predictive means.

sigma2_cal Numeric scalar. Predictive variance constant.

Sigma Numeric matrix. Predictive covariance matrix.

See Also

`update_y_eta_one`, `gen_pde`

update_y_eta_one *Update the likelihood of observations given PDE parameters (single sample)*

Description

Like update_y_eta but uses only one FFBS sample (indexed by ind_sam) instead of averaging over all samples, making it faster for MCMC proposals.

Usage

```
update_y_eta_one(
  eta,
  ind_sam,
  Nx,
  Ny,
  N_people,
  nT_ori,
  nT,
  nT_cal,
  N_sp,
  N_sp_train,
  AR_choice,
  bncol,
  res_ffbs,
  n_b_cal,
  ind_block_cal,
  t_VinvY_FTheta_ls,
  Vinv,
  R_cal_train,
  input,
  input_cal,
  phi_para,
  gp_sigma2,
  gp_tau2,
  ind_train_cal
)
```

Arguments

eta	Numeric vector. PDE parameters (length 5).
ind_sam	Integer. Index of the FFBS sample to use.
Nx	Integer. Grid cells in x-direction.
Ny	Integer. Grid cells in y-direction.
N_people	Numeric. Total population size.

`nT_ori`	Integer. Number of original time steps.
`nT`	Integer. Number of episode time steps.
`nT_cal`	Integer. Number of calibration time steps.
`N_sp`	Integer. Total number of spatial locations.
`N_sp_train`	Integer. Number of training spatial locations.
`AR_choice`	Integer. AR order (1 or 2).
`bncol`	Integer. Number of columns per block.
`res_ffbs`	List. FFBS posterior samples.
`n_b_cal`	Integer. Number of calibration blocks.
`ind_block_cal`	Data frame. Block indices for calibration.
`t_VinvY_FTheta_ls`	List. Precomputed $V^{-1}(Y - F\Theta)$ terms.
`Vinv`	Numeric matrix. Inverse of the observation covariance.
`R_cal_train`	Numeric matrix. Correlation matrix for training locations.
`input`	Numeric matrix. Observed location coordinates.
`input_cal`	Numeric matrix. Calibration location coordinates.
`phi_para`	Numeric scalar. GP range parameter.
`gp_sigma2`	Numeric scalar. GP marginal variance.
`gp_tau2`	Numeric scalar. GP nugget variance.
`ind_train_cal`	Integer vector. Indices of training locations.

Value

A named list with `y`, `mu`, and `Sigma`.

See Also

`update_y_eta`

Index