Package: Stem (via r-universe)

November 22, 2026

Type Package

Title Spatio-Temporal Expectation-Maximization Models

Version 2.0.0

Description Estimation of hierarchical spatio-temporal models through the Expectation-Maximization (EM) algorithm within a likelihood-based framework, following Fasso and Cameletti (2010) <doi:10.1177/0037549709102150> and the heterogeneous-network model of Fasso, Cameletti and Nicolis (2007) <doi:10.1002/env.837>, with the EM machinery of McLachlan and Krishnan (2008, ISBN:9780470191613). Parameters are estimated by maximum likelihood via Kalman filtering and smoothing; standard errors and spatial prediction uncertainty are obtained by a spatio-temporal parametric bootstrap. The package also provides the spatially-clustered STEM (SC-STEM) models, an extension in which regression coefficients, variance components and latent dynamics vary across spatial regimes estimated from the data by maximizing a Potts-penalized likelihood, in the spirit of the spatially-clustered regression of Sugasawa and Murakami (2021) <doi:10.1016/j.spasta.2021.100525> and of the spatially-clustered spatial autoregressive models of Cerqueti, Maranzano and Mattera (2025) <doi:10.1007/s13253-025-00685-7>. SC-STEM comes with likelihood-based information criteria, a two-step rule based on the Bayesian information criterion for selecting the number of regimes and the spatial penalty, and a refit-with-clustering parametric bootstrap that propagates the uncertainty of the estimated partition into intervals and tests.

License GPL (>= 2)

URL https://github.com/PaoloMaranzano/Stem

BugReports https://github.com/PaoloMaranzano/Stem/issues

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports geodist, MASS, parallel, spdep, stats

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

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Michela Cameletti [aut, cph] (ORCID:
<https://orcid.org/0000-0002-6502-7779>), Francesco Caccia
[ctb, cph] (earlier contributions to the code and preparation
of the povalley data), Paolo Maranzano [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-9228-2759>)

Maintainer Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

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

Date/Publication 2026-11-21 19:30:15 UTC

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

RemoteRef HEAD

RemoteSha e257e493f6e58e4f038bce4f0a5d2d1f0a6e895b

Contents

pm10 . . . 3
povalley . . . 4
print.SCSTEM_BootInference . . . 5
print.SCSTEM_Bootstrap . . . 6
print.SCSTEM_CV . . . 7
print.SCSTEM_Estimation . . . 7
print.SCSTEM_Infocrit . . . 8
print.SCSTEM_Select . . . 8
SCSTEM_BootInference . . . 9
SCSTEM_Bootstrap . . . 11
SCSTEM_Complete . . . 14
SCSTEM_CV . . . 15
SCSTEM_Estimation . . . 17
SCSTEM_Infocrit . . . 25
SCSTEM_Select . . . 27
SCSTEM_Signal . . . 30
STEM_Bootstrap . . . 31
STEM_Complete . . . 34
STEM_control . . . 36
STEM_Estimation . . . 38
STEM_Fit . . . 44

STEM_Kriging 48
STEM_Model 52
STEM_Signal 54
STEM_Simulation 55

Index **58**

`pm10` *Realistic data which illustrate the usage of the package Stem*

Description

This simple dataset is a list of three objects and refers to 22 spatial locations and 366 time points.

Usage

```
data(pm10)
```

Format

A list with three objects containing the following components:

coords A matrix with the coordinates of the 22 spatial locations.

covariates An 8052 by 3 matrix referring to the following covariates: *intercept, emissions (g/s), and altitude (km)*. The first 366 rows refer to the first spatial location, the rows from 367 to 732 refer to the second spatial location, and so on.

z A 366 by 22 observation matrix referring to *PM10 concentration measurements* (log scale).

Author(s)

Michela Cameletti <michela.cameletti@unibg.it>

References

Fasso, A., Cameletti, M. (2007) *A general spatio-temporal model for environmental data*. Tech.rep. n.27 *Graspa* - The Italian Group of Environmental Statistics.

Examples

```
data(pm10)
names(pm10)

# Plot the coordinates
dim(pm10$coords)
plot(pm10$coords[,1], pm10$coords[,2], xlab=colnames(pm10$coords)[1],
     ylab=colnames(pm10$coords)[2])

# Plot the data
dim(pm10$z)
```

```
# Summary by station
apply(pm10$z, 2, summary)

# Plot the time series for station n.22
plot(pm10$z[,22], type="l", xlab="Days", ylab="PM10 concentrations (log)")

# Plot the station altitude
plot(pm10$covariates[,3], ylab=colnames(pm10$covariates)[3], xaxt="n", xlab="")
positions <- seq(1, 8052, 366) + 366/2
axis(1, at=positions, labels=rownames(pm10$coords), las=2)
```

`povalley` *Daily particulate matter in the Po Valley, 2019-2023*

Description

Daily concentrations of fine particulate matter measured at 36 background monitoring stations of the Po Valley (Northern Italy) over the five years 2019-2023, together with the covariates used in the reference specification. The dataset is the real-data example of the spatially-clustered STEM (SC-STEM) routines: the Po Valley is a large alluvial basin framed by the Alps and the Apennines, whose ventilation and emission regimes differ enough across sub-areas to make a single pooled regression questionable.

Usage

```
data(povalley)
```

Format

A list with five components, following the same conventions as `pm10`:

coords A 36 by 2 matrix with the longitude and latitude of the stations, in decimal degrees (EPSG:4326). Distances must therefore be computed with `distance = "geo"`.

covariates A 65736 by 3 matrix with the *intercept*, the *station altitude* (meters above sea level) and the daily *PM10* concentration (micrograms per cubic meter). Rows are stacked by station: the first 1826 rows refer to the first station, the rows from 1827 to 3652 to the second one, and so on.

z A 1826 by 36 matrix of daily *PM2.5* concentrations (micrograms per cubic meter), the response variable.

dates The 1826 dates, from 2019-01-01 to 2023-12-31.

altitude The altitude of the 36 stations, in meters.

Details

The stations were selected from the European Environment Agency air quality database, restricted to the five regions of the Po basin (Lombardia, Piemonte, Veneto, Emilia-Romagna and Friuli-Venezia Giulia), and retained only when they covered the whole 2019-2023 window with at most 366 missing days on each pollutant and an altitude not exceeding 250 meters, so that the network describes the plain rather than the surrounding reliefs. Residual gaps and outliers were interpolated series by series, and the concentrations were rounded to the nearest integer and truncated at zero. The response could equally have been left incomplete, since `STEM_Estimation` now handles missing values in `z` natively; the covariates, which the model requires complete, had to be filled in any case.

Note that the response is on the original scale, not on the logarithmic one used by `pm10`.

Author(s)

Paolo Maranzano

Source

European Environment Agency, air quality e-reporting database, https://www.eea.europa.eu/en/datahub/datahubitem-view/3b390c9c-f321-490a-b25a-ae93b2ed80c1. Data prepared by Francesco Caccia.

See Also

`pm10`, `STEM_Model` and `SCSTEM_Estimation`

Examples

```
data(povalley)

dim(povalley$z)
range(povalley$dates)

# The monitoring network
plot(povalley$coords, pch = 19, xlab = "Longitude", ylab = "Latitude",
     main = "Po Valley background stations")

# PM2.5 at the first station
plot(povalley$dates, povalley$z[, 1], type = "l",
     xlab = "", ylab = expression(PM[2.5]))
```

`print.SCSTEM_BootInference` *Print method for the SC-STEM bootstrap inference*

Description

Print method for the SC-STEM bootstrap inference

Usage

```
## S3 method for class 'SCSTEM_BootInference'
print(x, ...)
```

Arguments

x	an object of class "SCSTEM_BootInference".
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing the cluster-wise summary table.

---

print.SCSTEM_Bootstrap

Print method for the SC-STEM parametric bootstrap

Description

Print method for the SC-STEM parametric bootstrap

Usage

```
## S3 method for class 'SCSTEM_Bootstrap'
print(x, ...)
```

Arguments

x	an object of class "SCSTEM_Bootstrap".
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing a compact summary of the bootstrap run.

print.SCSTEM_CV *Print method for the blocked cross-validation of STEM and SC-STEM fits*

Description

Print method for the blocked cross-validation of STEM and SC-STEM fits

Usage

```
## S3 method for class 'SCSTEM_CV'
print(x, digits = 3, ...)
```

Arguments

x	an object of class "SCSTEM_CV".
digits	integer, number of significant digits. Default is 3.
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing the errors and the ranking of the models under each scheme.

print.SCSTEM_Estimation *Print method for SC-STEM fits*

Description

Print method for SC-STEM fits

Usage

```
## S3 method for class 'SCSTEM_Estimation'
print(x, digits = 4, ...)
```

Arguments

x	an object of class "SCSTEM_Estimation".
digits	integer, number of significant digits. Default is 4.
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing a compact summary of the fit.

print.SCSTEM_Infocrit *Print method for SC-STEM information criteria*

Description

Print method for SC-STEM information criteria

Usage

```
## S3 method for class 'SCSTEM_Infocrit'
print(x, digits = 2, ...)
```

Arguments

x	an object of class "SCSTEM_Infocrit".
digits	integer, number of significant digits. Default is 2.
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing the grid of information criteria.

print.SCSTEM_Select *Print method for the SC-STEM selection rule*

Description

Print method for the SC-STEM selection rule

Usage

```
## S3 method for class 'SCSTEM_Select'
print(x, digits = 3, ...)
```

Arguments

x	an object of class "SCSTEM_Select".
digits	integer, number of significant digits. Default is 3.
...	further arguments, currently ignored.

Value

x, invisibly. Called for its side effect of printing the two steps of the selection rule and the selected configuration.

SCSTEM_BootInference *Inference from the SC-STEM parametric bootstrap*

Description

SCSTEM_BootInference post-processes the draws produced by SCSTEM_Bootstrap. It aligns the partition of every refit onto the clusters of the original fit, and returns standard errors, several families of confidence intervals for the cluster-wise parameters, and percentile tests for the differences between clusters.

Usage

```
SCSTEM_BootInference(SCSTEMboot, level = 0.95, parameters = NULL, digits = 5)
```

Arguments

SCSTEMboot	an object of class "SCSTEM_Bootstrap" returned by SCSTEM_Bootstrap.
level	confidence level of the intervals. Default is 0.95.
parameters	optional character vector selecting the parameters to report. Defaults to all the columns of phi_hat.
digits	integer, rounding applied to the returned tables. Default is 5.

Details

Label alignment. The clusters of a refit carry arbitrary labels, so before any statistic is computed each refit cluster is mapped to the cluster of the original fit with which it shares the largest number of locations, resolving conflicts greedily by decreasing overlap (majority rule). Draws whose refit did not recover all K clusters are discarded.

Confidence intervals. From the same set of aligned draws the function computes, for every cluster-wise parameter, the intervals of Davison and Hinkley (1997):

normal $\hat{\theta} - \hat{b} \pm z_{1-\alpha/2}\, se^*$, with $\hat{b}$ the bootstrap estimate of the bias.

basic $(2\hat{\theta} - q^*_{1-\alpha/2},\ 2\hat{\theta} - q^*_{\alpha/2})$.

percentile $(q^*_{\alpha/2},\ q^*_{1-\alpha/2})$.

bc the bias-corrected percentile interval, which shifts the percentile levels by the median bias of the bootstrap distribution.

Studentized (bootstrap-t) intervals are not reported: they would require an asymptotic standard error for every cluster-wise parameter at every draw, which the EM estimation of a STEM model does not provide – the bootstrap is precisely the way standard errors are obtained here.

Tests between clusters. For every parameter and every pair of clusters, the equal-tailed percentile test on the bootstrap distribution of the difference gives a two-sided p-value $2\min\{\Pr^*(\Delta \le 0), \Pr^*(\Delta \ge 0)\}$, truncated at one. It answers the question of whether two estimated regimes really differ in a given coefficient.

Interpretation. Because the partition is re-estimated at every draw, these intervals incorporate the uncertainty of the clustering and are therefore wider – and more honest – than intervals computed conditionally on the estimated partition.

Value

An object of class "SCSTEM_BootInference", a list with

- `summary`: data frame with one row per cluster and parameter, reporting the original estimate, the bootstrap mean, bias, standard error, t statistic, normal-approximation p-value and the four families of confidence intervals.
- `tests`: data frame of the pairwise percentile tests between clusters.
- `aligned`: the aligned bootstrap draws, as a B_{valid} by K by $npar$ array, so that any further statistic can be computed by the user.
- `coclustering`: d by d matrix of the bootstrap co-clustering frequencies, that is the share of usable draws in which two locations are assigned to the same regime. It is a direct measure of the stability of the estimated partition.
- `stability`: data frame with the Adjusted Rand Index between every refit partition and the original one.
- `level`, `B_used`: settings and number of draws used.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

References

Davison, A.C., Hinkley, D.V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press.

Efron, B., Tibshirani, R.J. (1993) *An Introduction to the Bootstrap*. Chapman and Hall, New York.

See Also

`SCSTEM_Bootstrap` and `SCSTEM_Estimation`

Examples

```
# Daily PM2.5 at 36 background stations of the Po Valley. The first 180
# days are used to keep the example fast; the covariates are the intercept,
# the station altitude and the daily PM10 concentration.
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
# rows of the stacked covariate matrix belonging to the first Tn days
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66,
            sigma2omega = 1e-06,
            theta = 2e-06,
            G = matrix(0.59, 1, 1),
            Sigmaeta = matrix(4.25, 1, 1),
```

```
            m0 = as.matrix(0),
            C0 = as.matrix(1))

mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


fit <- SCSTEM_Estimation(mod, K = 2, phi_penalty = 0.05, distance = 'geo')
boot <- SCSTEM_Bootstrap(fit, B = 20, seed = 1)
inf <- SCSTEM_BootInference(boot)
inf
inf$summary
```

SCSTEM_Bootstrap *Refit-with-clustering parametric bootstrap for SC-STEM models*

Description

SCSTEM_Bootstrap implements the parametric bootstrap for spatially-clustered STEM models. Bootstrap datasets are generated cluster by cluster from the fitted model and the *entire* SC-STEM procedure – including the endogenous partitioning – is re-estimated on every bootstrap sample, so that the uncertainty of the partition is propagated to all bootstrap statistics.

Usage

```
SCSTEM_Bootstrap(
  SCSTEM,
  B = 100,
  seed = NULL,
  verbose = FALSE,
  cores = 1L,
  ...
)
```

Arguments

SCSTEM	an object of class "SCSTEM_Estimation" returned by SCSTEM_Estimation.
B	integer, the number of bootstrap replicates. Default is 100.
seed	integer or NULL. When supplied, the draws are exactly reproducible; the RNG stream is restored on exit. Default is NULL.
verbose	logical. If TRUE, progress is reported via message(). Default is FALSE.

`cores` integer, the number of R processes the draws are computed on. Default is 1: one after the other, in the current session. With `cores > 1` a socket cluster of that many processes is started with the package **parallel**, the draws are shared among them in blocks, and the cluster is stopped on exit. The processes load the installed **Stem**. Every draw has its own seed, drawn from `seed`, so the draws are the same whatever the number of processes.

`...` further arguments passed to `SCSTEM_Estimation` for the refits, overriding the settings of the original fit.

Details

The distinguishing feature of the procedure is that it re-runs the clustering at every draw, rather than conditioning on the estimated partition. The algorithm adapts the refit-with-clustering bootstrap of Maranzano, Mattera and Sugasawa (2026+) to the spatio-temporal setting:

1. Fit the SC-STEM model on the observed data at fixed K and ϕ, obtaining the partition $\hat{P}$ and the cluster-wise parameter estimates $\hat{\phi}_k$.
2. For $b = 1, \ldots, B$ and for each cluster k, simulate a spatio-temporal dataset $z^{*(b)}$ from the STEM model of cluster k evaluated at $\hat{\phi}_k$, using the observed covariates, coordinates and loading matrix of the locations in that cluster. This is `STEM_Simulation` applied cluster by cluster, so that the within-cluster spatial covariance and the latent temporal dynamics of the fitted model are both reproduced.
3. Restore the original ordering of the locations, so that the simulated data are aligned with the rows of the spatial penalty graph and every location keeps its own neighbors.
4. Re-run `SCSTEM_Estimation` on $z^{*(b)}$ with the same K, ϕ and algorithmic settings, and store the cluster-wise estimates, the refit partition and the convergence diagnostics.

Because each bootstrap sample is generated cluster by cluster from the observed covariates and coordinates, the spatial structure of the original data is preserved: the fixed-effects surface inherits the contiguity of the estimated regimes while every location retains its position in the neighborhood graph.

Failed refits. A bootstrap sample whose refit raises an error, or whose refit collapses a cluster, is recorded but excluded from the inferential summaries computed by `SCSTEM_BootInference`. The share of usable draws is reported.

Seeds and parallel draws. Every draw is computed under its own seed, drawn from `seed` before the draws start, so a draw depends on its seed only and not on the draws computed before it. The draws can therefore be shared among several processes (`cores`) without changing the result: a bootstrap with a given `seed` returns the same draws whatever the number of processes. The draws are independent, so the time of a bootstrap falls nearly in proportion to the processes, up to the cost of starting them and of sending each the fitted model.

Label alignment. The clusters of a refit carry arbitrary labels, so they are relocated onto the clusters of the original fit by the majority rule before any statistic is computed. The alignment is applied by `SCSTEM_BootInference`, and the raw draws are returned here unaligned, together with the refit partitions, so that any alternative matching can be applied afterwards.

Value

An object of class “SCSTEM_Bootstrap”, a list with

- `draws`: a B by K by $npar$ array of the cluster-wise parameter estimates of every refit, with `NA` for failed draws.
- `groups`: a d by B matrix of the refit partitions.
- `loglik`: matrix of the cluster-wise log-likelihoods of the refits.
- `info`: a data frame with one row per draw reporting whether the refit succeeded, how many clusters it recovered, its total log-likelihood, the number of iterations and the convergence message.
- `original`: the "SCSTEM_Estimation" object given in input.
- `B`, `B_valid`: requested and usable number of draws.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>, Michela Cameletti

References

Chatterjee, S., Lahiri, P., Li, H. (2008) *Parametric bootstrap approximation to the distribution of EBLUP and related prediction intervals in linear mixed models*. Annals of Statistics, 36, 1221–1245. doi:10.1214/07AOS512

Davison, A.C., Hinkley, D.V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press.

Hall, P., Maiti, T. (2006) *On parametric bootstrap methods for small area prediction*. Journal of the Royal Statistical Society, Series B, 68, 221–238. doi:10.1111/j.14679868.2006.00541.x

See Also

`SCSTEM_Estimation`, `SCSTEM_BootInference` and `STEM_Bootstrap`

Examples

```
# Daily PM2.5 at 36 background stations of the Po Valley. The first 180
# days are used to keep the example fast; the covariates are the intercept,
# the station altitude and the daily PM10 concentration.
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
# rows of the stacked covariate matrix belonging to the first Tn days
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66,
            sigma2omega = 1e-06,
            theta = 2e-06,
            G = matrix(0.59, 1, 1),
            Sigmaeta = matrix(4.25, 1, 1),
            m0 = as.matrix(0),
            C0 = as.matrix(1))
```

```
mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


fit <- SCSTEM_Estimation(mod, K = 2, phi_penalty = 0.05, distance = 'geo')
boot <- SCSTEM_Bootstrap(fit, B = 20, seed = 1)
boot
```

SCSTEM_Complete *Completed response of a fitted SC-STEM model*

Description

SCSTEM_Complete returns the conditional expectation of the response given the observed data for a spatially-clustered STEM fit, by applying STEM_Complete within each estimated regime and re-assembling the columns in their original order.

Usage

```
SCSTEM_Complete(SCSTEM)
```

Arguments

SCSTEM	an object of class "SCSTEM_Estimation" returned by SCSTEM_Estimation.

Details

Because the regimes are fitted independently on disjoint sets of locations, the prediction of a missing entry borrows only from the locations of its own regime. This is a property of the model rather than of the implementation: the measurement covariance $\Sigma_{e,k}$ is defined within a regime, and two locations in different regimes are uncorrelated by construction.

Locations belonging to a regime that could not be re-estimated are returned unchanged, that is with their missing entries still missing.

Value

A T by d numeric matrix.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

See Also

STEM_Complete, SCSTEM_Estimation

SCSTEM_CV *Blocked cross-validation of the predictive accuracy of STEM and SCSTEM fits*

Description

SCSTEM_CV measures how well one or more fitted models predict data they have not seen, under the form of spatio-temporal extrapolation an application needs, and ranks them. It is a validation tool, optional and separate from the choice of the hyperparameters, which SCSTEM_Select makes: it can be applied at any time to the selected fit, to the pooled model, or to models with different covariates, numbers of regimes or penalties, all evaluated on the same folds.

Usage

```
SCSTEM_CV(
  SCSTEM,
  scheme = "LKLO",
  folds = 5L,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

SCSTEM	an object of class "SCSTEM_Estimation", or a named list of such objects to be compared on the same folds.
scheme	character vector, one or more of "LKLO" (default), "LKTO", "LKLHTO" and "random".
folds	integer, the number of folds of each scheme. Default is 5.
seed	optional integer seed for the allocation of locations and cells to the folds. The random number stream of the session is restored on exit.
verbose	logical. If TRUE, the progress is reported via message(). Default is FALSE.
...	arguments of SCSTEM_Estimation overriding those of the original fits in every refit, for instance max_iter.

Details

Each fold removes a set of cells from the data, refits every model on what is left with its own settings, and predicts the removed cells. Four blocking schemes are available (Otto, Fasso and Maranzano, 2024):

"LKLO" leave locations out: whole locations are removed. This is prediction at unmonitored sites, and the scheme on which a partition is tested: a removed location has no label of its own and inherits the regime of its nearest retained location.

"LKTO" leave times out: contiguous blocks of time points are removed at every location, which is gap filling and reconstruction in time.

`"LKLHTO"` leave locations and times out: a band of locations and a band of time points are both removed from the fit, and only their intersection is scored, so that neither the location nor the period of a scored cell contributes anything to the fit. The strictest scheme.

`"random"` cells removed at random. Every removed cell keeps its neighbors in space and time, so this scheme measures interpolation and is the most optimistic.

A removed cell of a retained location is predicted by `SCSTEM_Complete`, the conditional mean given the observed data of its regime. A removed location is predicted by the same conditional mean, computed with the parameters of the regime it inherits and given the retained locations of that regime observed at the same time point:

$$\hat{z}_{0t} = x'_{0t}\hat{\beta}_k + A_0\hat{y}_{k,t} + \Sigma_{0O}\Sigma_{OO}^{-1}(z_{Ot} - x'_{Ot}\hat{\beta}_k - A_O\hat{y}_{k,t}),$$

which is kriging within the regime. For the pooled model, fitted by `SCSTEM_Estimation(..., K = 1)`, this is ordinary kriging.

The refits reproduce the estimation of each model, with the settings stored in its `input_args` (number of regimes, penalty, graph, initialization, penalty on the coefficients), overridable through `...`. The models must share the response, the locations and the time points; their covariates may differ. The folds depend only on `seed`, the scheme and the dimensions of the data, so separate calls with the same `seed` use the same folds.

Value

An object of class "SCSTEM_CV", a list with

- `summary`: data frame with, for every scheme and model, the root mean squared error and the mean absolute error over all the scored cells, the number of scored cells, the number of folds whose refit succeeded, and the rank of the model by root mean squared error within the scheme.
- `folds`: data frame with the same errors fold by fold, and the error message of the folds whose refit failed.
- `models`: data frame with the number of regimes and the penalty of every model.
- `scheme`, `n_folds`, `seed`: the settings used.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

References

Otto, P., Fasso, A., Maranzano, P. (2024) *A review of regularized estimation methods and cross-validation in spatiotemporal statistics*. Statistics Surveys, 18, 299–340. doi:10.1214/24SS150

See Also

`SCSTEM_Select`, `SCSTEM_Complete`, `STEM_Complete`

Examples

```
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66, sigma2omega = 1e-06, theta = 2e-06,
            G = matrix(0.59, 1, 1), Sigmaeta = matrix(4.25, 1, 1),
            m0 = as.matrix(0), C0 = as.matrix(1))

mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


pooled <- SCSTEM_Estimation(mod, K = 1, distance = 'geo')
fit2 <- SCSTEM_Estimation(mod, K = 2, phi_penalty = 0.05, distance = 'geo')
cv <- SCSTEM_CV(list(pooled = pooled, k2 = fit2), scheme = "LKLO",
                folds = 4, seed = 1)
cv
```

`SCSTEM_Estimation` *Spatially-clustered STEM estimation*

Description

`SCSTEM_Estimation` fits a spatially-clustered spatio-temporal expectation-maximization (SC-STEM) model. The d monitoring locations are partitioned into K latent spatial regimes, and a separate "STEM_Model" is estimated within each regime, so that regression coefficients, variance components and latent temporal dynamics are all cluster-specific. The partition and the parameters are estimated jointly by alternating optimization of a Potts-penalized log-likelihood.

This is one of the two estimation engines of the package. The entry point is `STEM_Fit`, which calls this function when `K > 1` and `STEM_Estimation` when `K = 1`, and which takes the same arguments and returns the same object. Call `SCSTEM_Estimation` directly only to bypass the dispatch.

Usage

```
SCSTEM_Estimation(
  StemModel,
  K = 3,
  phi_penalty = 0.05,
  phi_scale = c("auto", "per-observation", "raw"),
```

```
  knn = 5,
  distance = c("geo", "euclidean"),
  init_method = c("departures", "kmeans", "coordinates"),
  init_partition = NULL,
  label_update = c("ICM", "simultaneous"),
  precision = NULL,
  precision_full_dataset = NULL,
  regularization = 0,
  max_iter = NULL,
  abs_tol = NULL,
  rel_tol = NULL,
  min_cluster_size = NULL,
  enforce_min_size = TRUE,
  swap_pass = TRUE,
  share2conv = 0,
  seed = 123456789,
  alpha = 0,
  lambda = 0,
  penalize = NULL,
  lambda_scale = c("relative", "absolute"),
  lambda_by = c("common", "size"),
  latent = TRUE,
  spatial = TRUE,
  verbose = FALSE,
  control = NULL,
  refit_cache = NULL
)
```

Arguments

`StemModel`	an object of class "STEM_Model" given as output by the `STEM_Model` function.
`K`	integer, the number of spatial clusters. `K = 1` returns the pooled STEM fit. Default is 3.
`phi_penalty`	non-negative number, the weight of the Potts spatial penalty. `phi_penalty = 0` gives non-spatial clusterwise STEM. Default is 0.05.
`phi_scale`	character, one of `"auto"` (default), `"per-observation"` or `"raw"`, setting the scale factor of the spatial penalty. See `Details`.
`knn`	integer, the number of nearest neighbors used to build the spatial penalty graph. The graph is symmetrized, and the neighbors are measured with the metric given by `distance`, so that the penalty and the covariance see the same geometry. Default is 5.
`distance`	character, `"euclidean"` for Euclidean distance or `"geo"` for geodesic distance. It governs both the covariance of the measurement error and the nearest neighbors of the penalty graph. Use `"geo"` only when the coordinates are longitude/latitude. Default is `"geo"`.
`init_method`	character, one of `"departures"` (default), `"kmeans"` or `"coordinates"`, the starting partition of an unpenalized fit. `"departures"` fits the pooled model and

runs k-means on how each location departs from it: the mean of its residual from the pooled signal, the slopes of that residual on its covariates, and the lag-one autocorrelation and log-variance of what the slopes leave. `"kmeans"` runs k-means on the location-wise covariate means, which carry no information on the regimes when the covariates are exogenous to them; `"coordinates"` clusters the spatial coordinates, which is also the fallback for intercept-only models. In every case the features are compressed by principal components, with multiple restarts and a minimum-cluster-size filter.

`init_partition` optional integer vector of length d giving a starting partition, overriding `init_method`. Use it to supply an externally computed initialization; for instance the AMKM partition used by versions of the package before 2.0.0 can be reproduced by passing `SCDA::SC_AMKM(...)$df$cluster`. The partition is repaired if it violates `min_cluster_size`. With `phi_penalty > 0` and no `init_partition`, the fit starts from the solution of the unpenalized fit at the same K, itself started from `init_method`; a given `init_partition` is taken as that start, which is how `SCSTEM_Infocrit` passes the partition of its unpenalized fit.

`label_update` character, `"ICM"` (default) for the sequential Iterated Conditional Modes sweep, or `"simultaneous"` for the joint update of all labels.

`precision`, `precision_full_dataset`, `max_iter`, `abs_tol`, `rel_tol`
optional overrides of `control`, kept for the code written before it existed: when given, `precision` replaces `alt_em_tol_par` (the EM tolerance inside the alternation), `precision_full_dataset` replaces `em_tol_par` (the EM tolerance of the pooled fit and of the final refit), `max_iter` replaces `alt_maxit`, and `abs_tol`, `rel_tol` replace `alt_abs_tol`, `alt_rel_tol`. Default `NULL` for all.

`regularization` a non-negative number added to the diagonal of the matrices inverted by the EM algorithm; see `STEM_Estimation`. Default is 0.

`min_cluster_size`
integer or `NULL`. Minimum number of locations required to estimate a cluster-wise model. When `NULL` (default) it is set to `ncov + 2`.

`enforce_min_size`
logical. If `TRUE` (default) the assignment step never lets a cluster fall below `min_cluster_size`. See `Details`. Set to `FALSE` to reproduce the unconstrained sweep.

`swap_pass` logical. If `TRUE` (default) each sweep is followed by a size-preserving swap pass that exchanges the labels of pairs of locations whenever this increases the objective. See `Details`.

`share2conv` number in $[0, 1)$. Optional early-stopping rule kept for backward compatibility: the loop also stops when the share of locations changing cluster falls below this value. Set to 0 (default) to rely only on the objective-based criteria.

`seed` integer or `NULL`, seed used for the initialization step so that the fit is reproducible. Default is 123456789.

`alpha` the elastic-net mixing parameter for the regression coefficients, in $[0, 1]$: `0` is ridge, `1` the lasso, anything between the elastic net. The penalty acts within each regime. Ignored when `lambda` is zero. Default is 0. See `STEM_Fit`.

`lambda`	the strength of the penalty on the regression coefficients. `0`, the default, gives the unpenalized estimator, and the information criteria then count the nominal number of coefficients as before; with `lambda > 0` they count the effective number the penalty leaves.
`penalize`	which coefficients the penalty acts on: a logical vector of length r, an index vector, or `NULL` (the default) for every coefficient except the intercept.
`lambda_scale`	how `lambda` is measured, "relative" (the default) or "absolute". See `STEM_Estimation`.
`lambda_by`	how the penalty is distributed over the regimes. "common", the default, gives every regime the same `lambda`, which under the relative scale already means the same proportional shrinkage, since the reference is computed inside each regime. "size" sets $\lambda_g = \lambda\bar{n}/n_g$ with $\bar{n} = d/K$, shrinking a regime of half the average size twice as hard. Either way `lambda` stays ONE hyperparameter: genuinely cluster-specific (α_g, λ_g) is a different model and is not offered here.
`latent`	logical, passed to `STEM_Estimation` within each regime. Default is `TRUE`.
`spatial`	logical, passed to `STEM_Estimation` within each regime. Default is `TRUE`.
`verbose`	logical. If `TRUE`, progress information is emitted via `message()`. Default is `FALSE`.
`control`	the computational settings, an object returned by `STEM_control` or a list of some of its settings. Default `NULL`, the defaults of `STEM_control()`. The pooled fit and the final refit run with its `em_*` settings, the EM algorithm inside the alternation with its `alt_em_*` settings; see `Details`.
`refit_cache`	`NULL` (default) or an environment in which the final refits are stored by number of regimes and partition, so that fits of the same data share them (see `Details`). `SCSTEM_Infocrit` passes one to every fit of its grid; it is not a setting to tune, and an environment filled on other data must never be passed.

Details

Model. Conditionally on location i belonging to cluster $k \in \{1, \dots, K\}$, the SC-STEM model is the cluster-specific STEM model

$$z_{it} = x'_{it}\beta_k + A_i y_t^{(k)} + e_{it}, \qquad y_t^{(k)} = G_k y_{t-1}^{(k)} + \eta_t^{(k)},$$

with $e_t \sim N(0, \Sigma_{e,k})$, $\Sigma_{e,k} = \sigma^2_{\epsilon k} I + \sigma^2_{\omega k} C(h; \theta_k)$ and $\eta_t^{(k)} \sim N(0, \Sigma_{\eta k})$. Setting $K = 1$ recovers the pooled `STEM_Estimation` fit.

Objective. Labels $k_1, \dots, k_d$ and parameters are estimated by maximizing the penalized log-likelihood

$$Q = \sum_{i=1}^{d} \ell_i(k_i) + \phi\, c \sum_{(i,j)\in E} I(k_i = k_j),$$

where $\ell_i(k)$ is the log-likelihood contribution of location i under the parameters of cluster k, E is the edge set of the symmetrized `knn` graph with each unordered pair counted once, and $c > 0$ is the scale factor discussed below. This is the Potts-type penalty introduced for spatially-clustered regression by Sugasawa and Murakami (2021) and carried over to spatially-clustered spatial autoregressive models by Cerqueti, Maranzano and Mattera (2025); here it is applied to the spatio-temporal likelihood of a STEM model.

Scale of the penalty. In cross-sectional spatially-clustered models each unit contributes a single observation to the likelihood, so that ϕ of order one balances fit against spatial cohesion. Here each location contributes T observations, and the log-likelihood differences between clusters grow with T and with the scale of the response. The factor c restores a comparable interpretation:

- `phi_scale = "auto"` (default) sets c to the median across locations of the spread $\max_k \ell_{ik} - \min_k \ell_{ik}$, divided by the average number of neighbors. With this normalization $\phi = 1$ is the point at which full agreement with the neighborhood is worth about as much as the typical gain from picking the best-fitting cluster. The factor is computed once, at the first sweep, and is returned in `phi_multiplier`. Since a penalized fit starts from the unpenalized solution (see below), the first sweep scores the locations with the parameters of the unpenalized fit: the factor is measured on the fitted clusters, does not depend on `init_method`, and is of the order of the whole gain of the right regime over the wrong ones, so the informative range of ϕ is small: the default grid of `SCSTEM_Infocrit` is $\phi \in \{0, 0.025, 0.05, 0.1, 0.2, 0.5, 1\}$, and its last two values already override the likelihood for most locations near a border.
- `phi_scale = "per-observation"` sets $c = T$, which makes ϕ invariant to the length of the series and directly comparable with the cross-sectional literature, but leaves it dependent on the scale of the response.
- `phi_scale = "raw"` sets $c = 1$, penalizing on the untransformed likelihood scale.

The effective penalty actually applied is always reported in `phi_effective`.

Penalized fits start from the unpenalized one. With `phi_penalty > 0` the fit with `phi_penalty = 0` at the same K is run first, and its solution is the starting partition. A penalty that is strong from the first sweep freezes whatever partition it is given, so a start unrelated to the regimes would stay where it is; from the unpenalized solution the penalty only has to decide how much spatial smoothing that solution can afford. `SCSTEM_Infocrit` fits $\phi = 0$ once per K and passes its partition on, so a single fit and the corresponding member of a grid coincide.

Algorithm. The two steps are iterated until convergence:

1. *Parameter update given labels.* A STEM model is estimated by `STEM_Estimation` on each cluster. A cluster that falls below the minimum admissible size keeps its last valid parameters ("stale freeze") instead of aborting the algorithm; a cluster that never obtained a valid fit is excluded from the assignment step.
2. *Label update given parameters.* With `label_update = "ICM"` (the default) locations are visited sequentially and each label maximizes its own penalized contribution given the current labels of all the others, in the spirit of the Iterated Conditional Modes algorithm of Besag (1986). For fixed parameters this sweep cannot decrease Q, which rules out the label cycling that a simultaneous update can produce. `label_update = "simultaneous"` reproduces the joint update of all labels used in earlier versions of the package and in Sugasawa and Murakami (2021).

The loop stops when the partition is unchanged, when the improvement of Q falls below `abs_tol`/`rel_tol`, when a previously visited partition reappears, or when `max_iter` is reached.

On monotonicity. The label step is monotone at fixed parameters, but the alternation as a whole is *not* guaranteed to increase Q monotonically, and `obj_trace` may well show a decrease. The reason is structural rather than numerical. In the STEM measurement equation the spatial covariance $\Sigma_{e,k}$ couples the locations, so the exact marginal likelihood does not factorize across them and the assignment score ℓ_{ik} has to be a conditional pseudo-likelihood, whereas the parameter step

maximizes the exact within-cluster likelihood through the EM algorithm. The two objectives agree on what a good partition looks like but are not the same function, so a parameter update can lower Q while raising the exact likelihood. This is a property of the spatio-temporal specification, and it is why the best partition visited along the iterations is always the one returned.

Degeneracy and the minimum-size constraint. Within-cluster homogeneity is exactly what the assignment step seeks, so the cluster-wise variance components $\sigma^2_{\epsilon k} + \sigma^2_{\omega k}$ shrink as locations are reallocated. Left unconstrained, the cluster with the smallest residual variance then attracts every location and the partition collapses – the clusterwise counterpart of the degenerate-likelihood problem of Gaussian mixtures. With `enforce_min_size = TRUE` (the default) a location may leave its cluster only if that cluster stays at or above `min_cluster_size`, which keeps every visited configuration admissible while preserving the monotonicity of the ICM sweep within the feasible set. The constraint matters most when the number of locations is small relative to K; it can be lifted with `enforce_min_size = FALSE`.

The constraint has a cost of its own: a location sitting in a cluster that is exactly at the minimum size can never move, so with small networks a large share of the partition may stay frozen at its initial value. Each sweep is therefore followed by a *size-preserving swap pass* (`swap_pass = TRUE`, the default): the labels of two locations in different clusters are exchanged whenever the exchange strictly increases Q. A swap leaves every cluster size unchanged, so feasibility holds by construction, and it is accepted only on strict improvement, so the monotonicity of the algorithm is preserved. The number of accepted swaps is reported in `obj_trace`.

Starting values of the regimes. The first time a regime is fitted, its EM algorithm starts from the starting values of `StemModel` with the regression coefficients replaced by the least-squares fit on the locations of the regime. A penalized fit, which starts from the partition of the unpenalized fit at the same K, starts its regimes from the final refit of that partition instead, when the refit is at hand (see "Shared refits"). At every later iteration of the alternation a regime starts from its own estimates of the previous iteration, so that the EM algorithm resumes rather than restarts, and the final refit starts from the estimates of the last iteration. A regime that has no estimates yet starts from its least-squares fit.

Final refit and information criteria. On convergence the cluster-wise STEM models are re-estimated once on the final partition, with the `em_*` settings of `control`, the same as the pooled fit at $K = 1$, so that the log-likelihoods compared across K are computed to the same accuracy. All reported coefficients, variance components and information criteria come from this refit and are based on the *exact* cluster-wise log-likelihoods returned by `STEM_Estimation`, not on the pseudo-likelihood used to rank clusters during the assignment step. The number of free parameters is $k_{eff}(r + 3 + 3p)$ for a diagonal specification with r covariates and latent dimension p, where k_{eff} counts the clusters that could actually be re-estimated. Whether the EM algorithm of each regime met its stopping rule within `em_maxit` iterations is reported in `em_converged`.

Shared refits. The refit depends on the data, the partition and the settings, not on the penalty, so fits of the same data that end at the same partition share one refit: a penalized fit that ends where its unpenalized start ended reuses that refit, and `SCSTEM_Infocrit` shares the refits across its grid. Two refits of one partition from different starts stop at different points of the likelihood, usually within the tolerance of the EM algorithm, occasionally further apart when a small regime starts near the boundary of its parameter space; either way the difference is not a property of the penalty. Sharing the refit makes the fits of one partition identical, so that a comparison across penalties at one K compares partitions only. `refit_reused` reports whether the refit was shared.

Statistical features and scope. The assumptions are inherited from the STEM model and determine which datasets the family applies to. The response is *Gaussian* and *univariate*: `z` is a T by d matrix

of one variable measured at d sites, so several pollutants modeled jointly is a different specification. What is multivariate is the *latent state*, of dimension $p \geq 1$, loaded onto the locations by the known matrix A, which is not estimated and is common across regimes. The latent dynamics is a VAR(1), with G and Σ_η diagonal by default; m_0 is estimated and C_0 is held fixed. The spatial correlation function is exponential, hence isotropic and stationary *within* a regime, the partition itself being the only source of non-stationarity across the domain. Locations are point-referenced and fixed over time, and the time index is discrete and regularly spaced.

Missing values are supported in the *response*, and handled as prescribed by Durbin and Koopman (2012, 2nd ed.), Sections 2.7 and 4.10: at each time point the measurement equation is restricted to the locations actually observed, through a selection matrix whose rows are a subset of the rows of the identity, and a time point at which nothing is observed contributes no update and no likelihood term. Because the EM algorithm maximizes the expected complete-data log-likelihood, the M-step completes the sufficient statistics: a missing value enters through its conditional expectation given everything observed, and its conditional variance is added back as a correction. Note that this conditional expectation is not the signal alone, since the spatial covariance couples the locations, so the missing block of the measurement error is predicted from the observed one by the same algebra as kriging at a fixed time point. In the assignment step, each location is scored on the time points at which it was observed.

Missing values are *not* supported in the covariates or the coordinates: the design matrix enters the closed-form M-step updates directly and the coordinates enter the distance matrix, so gaps there would require a stochastic E-step. Impute them before fitting. Every location must retain at least one observation.

Value

An object of class "SCSTEM_Estimation", a list with components:

- `phi_hat`: K by $npar$ matrix of cluster-wise parameter estimates from the final refit.
- `group`: integer vector of length d with the estimated cluster label of each location.
- `df`: data frame with the coordinates and the estimated labels.
- `fit_list`: list of the "STEM_Model" objects returned by `STEM_Estimation` on the final partition.
- `idx_g`: list of the location indices of each cluster.
- `info_crit`: named vector with the total log-likelihood, the number of free parameters and AIC, BIC and KIC.
- `loglik_g`: cluster-wise exact log-likelihoods.
- `final_refit`: logical vector flagging the clusters that could be re-estimated on the final partition.
- `em_converged`: logical vector, whether the EM algorithm of each re-estimated cluster met its stopping rule within `em_maxit` iterations (`NA` for a cluster not re-estimated).
- `theta_bound`: character vector, for each re-estimated cluster whether its range parameter ended at one of the limits that the distances between its locations can identify (`"lower"`, `"upper"`) or within them (`"none"`); see the section on the range in `STEM_Estimation`.
- `refit_reused`: logical, whether the final refit was shared with an earlier fit of the same partition.

- `obj_trace`: data frame tracing the penalized objective, the number of label changes and the cluster sizes along the iterations.
- `convergence`: character describing the exit route.
- `penalized_obj`: value of the penalized objective at the exit.
- `input_args`: the arguments used for the fit, needed by `SCSTEM_Bootstrap` and `SCSTEM_Select`.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>, Michela Cameletti

References

Besag, J. (1986) *On the statistical analysis of dirty pictures*. Journal of the Royal Statistical Society, Series B, 48, 259–302.

Cerqueti, R., Maranzano, P., Mattera, R. (2025) *Spatially-clustered spatial autoregressive models with application to agricultural market concentration in Europe*. Journal of Agricultural, Biological and Environmental Statistics. doi:10.1007/s13253025006857

Fasso, A., Cameletti, M., Nicolis, O. (2007) *Air quality monitoring using heterogeneous networks*. Environmetrics, 18, 245–264. doi:10.1002/env.837

Fasso, A., Cameletti, M. (2010) *A unified statistical approach for simulation, modeling, analysis and mapping of environmental data*. Simulation, 86, 139–153. doi:10.1177/0037549709102150

Sugasawa, S., Murakami, D. (2021) *Spatially clustered regression*. Spatial Statistics, 44, 100525. doi:10.1016/j.spasta.2021.100525

See Also

`STEM_Model`, `STEM_Estimation`, `SCSTEM_Infocrit`, `SCSTEM_Select`, `SCSTEM_Bootstrap` and `pm10`

Examples

```
# Daily PM2.5 at 36 background stations of the Po Valley. The first 180
# days are used to keep the example fast; the covariates are the intercept,
# the station altitude and the daily PM10 concentration.
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
# rows of the stacked covariate matrix belonging to the first Tn days
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66,
            sigma2omega = 1e-06,
            theta = 2e-06,
            G = matrix(0.59, 1, 1),
            Sigmaeta = matrix(4.25, 1, 1),
            m0 = as.matrix(0),
            C0 = as.matrix(1))
```

```
mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


# three spatial regimes with a moderate spatial penalty
fit <- SCSTEM_Estimation(mod, K = 3, phi_penalty = 0.05, distance = 'geo')
fit

# the estimated regimes on the map
plot(povalley$coords, col = fit$group, pch = 19,
     xlab = 'Longitude', ylab = 'Latitude')
```

SCSTEM_Infocrit *Information criteria over a grid of SC-STEM configurations*

Description

SCSTEM_Infocrit fits the spatially-clustered STEM model over a grid of numbers of regimes K and spatial penalties ϕ, and returns the likelihood-based information criteria of every configuration together with the diagnostics needed to decide which of them are admissible. It is the input of the two-step tuning rule implemented in SCSTEM_Select.

Usage

```
SCSTEM_Infocrit(
  StemModel,
  K_grid = 1:3,
  phi_grid = c(0, 0.025, 0.05, 0.1, 0.2, 0.5, 1),
  verbose = FALSE,
  ...
)
```

Arguments

StemModel	an object of class "STEM_Model" given as output by the STEM_Model function.
K_grid	integer vector of candidate numbers of regimes. Default is 1:3. The pooled model $K = 1$ is always a useful reference.
phi_grid	numeric vector of candidate non-negative spatial penalties. Default is c(0, 0.025, 0.05, 0.1, 0.2, 0.5, 1): the moderate values up to 0.2 form the band in which SCSTEM_Select chooses K, and the strong values 0.5 and 1 compete only in the choice of ϕ. At every K the fit at $\phi = 0$ starts the penalized ones (see SCSTEM_Estimation), so the grid should contain 0. The fits at one K that end at the same partition share one final refit, and therefore have the same log-likelihood and criteria.

verbose	logical. If TRUE, the progress over the grid is reported via message(). Default is FALSE.
...	further arguments passed to SCSTEM_Estimation, such as distance, knn, phi_scale, label_update, precision or min_cluster_size.

Details

For each pair (K, ϕ) the function calls SCSTEM_Estimation and records the exact total log-likelihood of the final refit, the number of free parameters $\mathrm{df} = K_{eff}(r + 3 + 3p)$, and

$$AIC = -2\ell + 2\,\mathrm{df}, \qquad BIC = -2\ell + \log(N)\,\mathrm{df}, \qquad KIC = -2\ell + 3\,\mathrm{df},$$

where N is the number of observations entering the likelihood, the observed values of the response: $N = dT$, with d locations and T time points, when the panel is complete.

Admissibility. A configuration is admissible when all its K regimes could be re-estimated on the final partition, that is when K_eff == K. A configuration with collapsed regimes has fewer effective regimes than its nominal K, its likelihood is not comparable with the others, and no parametric bootstrap can be generated from it: the admissible column flags these cases and SCSTEM_Select discards them.

Caveats. Two warnings apply to any likelihood-based comparison in this setting. First, the partition is itself optimized on the data, so the maximized likelihood retains an optimism bias that a parameter count of the form $K(r + 3 + 3p)$ does not fully correct; in-sample criteria therefore tend to favor small ϕ and large K. Second, the cluster-wise variance components shrink as the assignment step pursues within-cluster homogeneity, which inflates the likelihood of the configurations with the weakest spatial regularization. Both effects are strongest at $\phi \approx 0$, which is why SCSTEM_Select chooses K within a moderate band of penalties.

Value

An object of class "SCSTEM_Infocrit": a list with

- table: a data frame with one row per configuration and columns K, phi, K_eff, admissible, loglik, df, AIC, BIC, KIC, min_size, convergence.
- fits: the list of "SCSTEM_Estimation" objects, named "K=<K>, phi=<phi>", so that the selected configuration can be used without refitting.
- groups: matrix of the estimated partitions, one column per configuration, which SCSTEM_Select compares with the selected one.
- failed: a data frame listing the configurations whose fit raised an error, with the error message.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>, Michela Cameletti

References

Cerqueti, R., Maranzano, P., Mattera, R. (2025) *Spatially-clustered spatial autoregressive models with application to agricultural market concentration in Europe*. Journal of Agricultural, Biological and Environmental Statistics. doi:10.1007/s13253025006857

Sugasawa, S., Murakami, D. (2021) *Spatially clustered regression*. Spatial Statistics, 44, 100525. doi:10.1016/j.spasta.2021.100525

See Also

SCSTEM_Estimation and SCSTEM_Select

Examples

```
# Daily PM2.5 at 36 background stations of the Po Valley. The first 180
# days are used to keep the example fast; the covariates are the intercept,
# the station altitude and the daily PM10 concentration.
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
# rows of the stacked covariate matrix belonging to the first Tn days
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66,
            sigma2omega = 1e-06,
            theta = 2e-06,
            G = matrix(0.59, 1, 1),
            Sigmaeta = matrix(4.25, 1, 1),
            m0 = as.matrix(0),
            C0 = as.matrix(1))

mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


ic <- SCSTEM_Infocrit(mod, K_grid = 1:3, phi_grid = c(0, 0.05),
                      distance = 'geo')
ic
```

SCSTEM_Select *Two-step selection of the SC-STEM hyperparameters*

Description

SCSTEM_Select applies the two-step tuning rule for the number of clusters K and the spatial penalty ϕ of a spatially-clustered STEM model, given the grid of fits produced by SCSTEM_Infocrit.

Usage

```
SCSTEM_Select(
  infocrit,
  band = c(0.025, 0.2),
  criterion = c("BIC", "AIC", "KIC")
)
```

Arguments

`infocrit`	an object of class "SCSTEM_Infocrit" returned by `SCSTEM_Infocrit`.
`band`	numeric vector of length two giving the moderate-penalty band Φ_M of step (S1). Default is `c(0.025, 0.2)`, the moderate non-zero values of the default grid of `SCSTEM_Infocrit`; its strong values 0.5 and 1 lie outside the band and enter step (S2) only. Grid values falling inside the closed interval are used. If the band contains fewer than two grid values, the whole grid is used and a warning is issued.
`criterion`	character, the information criterion used in both steps. One of `"BIC"` (default), `"AIC"` or `"KIC"`.

Details

In-sample information criteria cannot be used naively here: the partition is optimized on the data, so the maximized likelihood keeps an optimism bias that the parameter count does not fully correct, and the cluster-wise variance components shrink as the assignment step pursues within-cluster homogeneity. Both effects are strongest at $\phi \approx 0$, where in-sample criteria systematically overselect K. The rule therefore chooses K within a *moderate-penalty band* Φ_M and proceeds in two steps:

(S1) Number of clusters, by BIC within the band. For each $\phi \in \Phi_M$, record the BIC-minimizing K among the admissible configurations at that ϕ and the pooled model $K = 1$, whose criterion does not depend on ϕ; select the modal winner $\hat{K}$ across the band, resolving ties towards the smaller K. If $\hat{K} = 1$ the data do not support a partition: the pooled fit is returned and (S2) does not apply. Excluding $\phi \approx 0$ removes the region where the optimism bias is largest. Under-selection is the harmful direction, while residual over-selection is comparatively benign for prediction, because supernumerary clusters are either small or near-duplicates of existing regimes.

(S2) Spatial penalty, by the same criterion at $\hat{K}$. Among the admissible fits with $K = \hat{K}$ on the whole grid, select the one with the smallest criterion, resolving ties towards the smaller ϕ. At a given K the parameter count does not depend on ϕ (up to the effective degrees of freedom of a penalty on the coefficients), so the step compares the likelihoods reached by the partitions estimated at different penalties. Every penalized fit starts from the unpenalized one at the same K (see `SCSTEM_Estimation`): a penalty is selected only when it has moved the partition to one with a higher likelihood than the unpenalized solution, and the optimism bias, which favors $\phi = 0$, makes the step conservative. Penalties outside the band can therefore be kept on the grid: they are selected only when the likelihood supports them. This is the rule of Cerqueti, Maranzano and Mattera (2025).

Only admissible configurations, in the sense of `SCSTEM_Infocrit`, enter the rule. The pooled model enters it whenever `K_grid` contains 1, as it should: it is the answer when no partition improves on it, and it is selected when no configuration with $K > 1$ is admissible.

Value

An object of class "SCSTEM_Select", a list with

- `K_selected`, `phi_selected`: the selected hyperparameters. When $\hat{K} = 1$, `phi_selected` is the grid value at which the pooled fit is stored, which has no effect on it.
- `fit`: the corresponding fitted object, taken from `infocrit$fits` without refitting.
- `step1`: data frame with the criterion-minimizing K at each penalty in the band.
- `step2`: data frame with the criterion of every admissible fit at $K = \hat{K}$, one row per penalty; `NULL` when $\hat{K} = 1$.
- `ari_to_selected`: Adjusted Rand Index between every admissible partition on the grid and the selected one.
- `reference`: the pooled $K = 1$ row of the criteria table, when available.
- `band`, `criterion`: the settings used.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

References

Cerqueti, R., Maranzano, P., Mattera, R. (2025) *Spatially-clustered spatial autoregressive models with application to agricultural market concentration in Europe*. Journal of Agricultural, Biological and Environmental Statistics. doi:10.1007/s13253025006857

See Also

`SCSTEM_Infocrit` and `SCSTEM_Estimation`

Examples

```
# Daily PM2.5 at 36 background stations of the Po Valley. The first 180
# days are used to keep the example fast; the covariates are the intercept,
# the station altitude and the daily PM10 concentration.
data(povalley)

Tn <- 180L
Tfull <- nrow(povalley$z)
d <- ncol(povalley$z)
# rows of the stacked covariate matrix belonging to the first Tn days
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))

phi <- list(beta = matrix(c(1.25, -0.00003, 0.64), 3, 1),
            sigma2eps = 18.66,
            sigma2omega = 1e-06,
            theta = 2e-06,
            G = matrix(0.59, 1, 1),
            Sigmaeta = matrix(4.25, 1, 1),
            m0 = as.matrix(0),
            C0 = as.matrix(1))
```

```
mod <- STEM_Model(z = povalley$z[seq_len(Tn), ],
                  covariates = povalley$covariates[keep, ],
                  coordinates = povalley$coords,
                  phi = phi, A = matrix(1, d, 1))


ic <- SCSTEM_Infocrit(mod, K_grid = 1:3, distance = 'geo')
sel <- SCSTEM_Select(ic)
sel
```

SCSTEM_Signal *Fitted values of a fitted SC-STEM model*

Description

SCSTEM_Signal returns the fitted values of a spatially-clustered STEM model: STEM_Signal applied within each estimated regime and reassembled onto the original ordering of the locations. Every column is computed from the parameters and the latent path of the regime its location was assigned to.

Usage

```
SCSTEM_Signal(SCSTEM)
```

Arguments

SCSTEM	an object of class "SCSTEM_Estimation", as returned by SCSTEM_Estimation or by STEM_Fit with K > 1.

Details

See STEM_Signal for what the quantity is and for why it is not SCSTEM_Complete. Regimes that the final refit could not estimate contribute columns of NA rather than a value borrowed from elsewhere.

Value

A T by d matrix of fitted values.

See Also

STEM_Signal, SCSTEM_Complete, SCSTEM_Estimation

STEM_Bootstrap *Parametric bootstrap*

Description

This functions performs the spatio-temporal parametric bootstrap for computing the parameter standard errors

Usage

```
STEM_Bootstrap(
  StemModel,
  B,
  distance = "euclidean",
  precision = NULL,
  regularization = 0,
  verbose = FALSE,
  control = NULL
)
```

Arguments

StemModel	an object of class "STEM_Model" given as output by the STEM_Model function.
B	number of bootstrap iterations.
distance	character, indicating the type of distance. 'euclidean' compute euclidean distance while 'geo' compute the geodesic distance. use 'geo' only if the coordinates format is Longitude, Latitude. Default is 'euclidean'.
precision	optional; when given, it replaces em_tol_par of control in every refit (see STEM_Estimation). Default NULL.
regularization	a non-negative number added to the diagonal of the matrices inverted by the EM algorithm of every refit; see STEM_Estimation. Default is 0.
verbose	logical. If TRUE, the progress of each bootstrap iteration is reported through message(). Default is FALSE.
control	the computational settings of the refits, an object returned by STEM_control or a list of some of its settings. Default NULL, the defaults.

Details

The spatio-temporal bootstrap is used for parameter uncertainty assessment. The resampling scheme is based on the estimated model: each bootstrap sample is drawn directly from the Gaussian distributions which define the model with the parameter vector replaced by the corresponding ML estimates. For each of the B bootstrap samples, the ML estimates are computed (using the procedure of STEM_Estimation function). Then the B bootstrap replications are returned in a list.

Value

The function returns a list of elements called "boot.output". **Each** element of the list is an object of class "STEM_Model" and so it is composed by the following elements:

- skeleton a list with components `phi`, `p`, `A` where the `phi` vector is given by the `StemModel$estimates$phi.hat` vector.
- data a list with components `z`, `coordinates`, `covariates`, `r`, `n` and `d` where the `z` matrix is given by the simulated data matrix.
- estimates A list of four objects: `phi.hat`, `y.smoothed`, `loglik`, `convergence.par` as the output of the `STEM_Estimation` function.

Author(s)

Michela Cameletti < michela.cameletti@unibg.it >

References

Amisigo, B.A., Van De Giesen, N.C. (2005) *Using a spatio-temporal dynamic state-space model with the EM algorithm to patch gaps in daily riverflow series*. Hydrology and Earth System Sciences 9, 209–224. Fasso', A., Cameletti, M., Nicolis, O. (2007) *Air quality monitoring using heterogeneous networks*. Environmetrics 18, 245–264. Fasso', A., Cameletti, M. (2007) *A general spatio-temporal model for environmental data*. Tech.rep. n.27 *Graspa* - The Italian Group of Environmental Statistics. Fasso, A. and M. Cameletti (2010). *A Unified Statistical Approach for Simulation, Modeling, Analysis and Mapping of Environmental Data*. SIMULATION 86(3): 139-153. <doi: 10.1177/0037549709102150> Xu, K., Wikle, C.K. (2007) *Estimation of parameterized spatio-temporal dynamic models*. Journal of Statistical Inference and Planning 137, 567–588. Mc Lachlan, G.J., Krishnan, T. (1997) *The EM Algorithm and Extensions*. Wiley, New York. Shumway, R.H., Stoffer, D.S. (2006) *Time Series Analysis and Its Applications: with R Examples*. Springer, New York.

See Also

See Also `pm10`, `STEM_Model` and `STEM_Estimation`

Examples

```
#load the data
data(pm10)

#extract the data
coordinates <- pm10$coords
covariates <- pm10$covariates
z <- pm10$z

#build the parameter list
#(the phi list is used for the algorithm starting values)
phi <- list(beta=matrix(c(3.65,0.046,-0.904),3,1),
            sigma2eps=0.1,
            sigma2omega=0.2,
            theta=0.01,
```

```
            G=matrix(0.77,1,1),
            Sigmaeta=matrix(0.3,1,1),
            m0=as.matrix(0),
            C0=as.matrix(1))

A <- matrix(1,ncol(z),1)

mod1 <- STEM_Model(z=z,covariates=covariates,
                   coordinates=coordinates,phi=phi,A=A)
class(mod1)


#mod1 is given as output by the STEM_Model function
mod1.est <- STEM_Estimation(mod1)

#it is computer intensive
mod1.boot <- STEM_Bootstrap(StemModel=mod1.est, B=3)
names(mod1.boot)
#the first element of the output list
names(mod1.boot$boot.output[[1]])


#check if there is no convergence for some bootstrap iteration
B <- length(mod1.boot$boot.output)
n.null <- sum (unlist (lapply(mod1.boot$boot.output,
                              function(x) length(x$StemModel)==1)) )
pos.null <- which((unlist (lapply(mod1.boot$boot.output,
                                  function(x) length(x$StemModel)==1)) ))
message("------------B non null =", B - n.null)
if(length(pos.null)>0) boot1.mod = mod1.boot[-pos.null]

#put the bootstrap output in a matrix
npar <- length(unlist((mod1.boot$boot.output[[1]]$estimates$phi.hat)))-1
boot.estimates <- matrix(NA, nrow = (B - n.null), ncol = npar)

for(b in 1:(B - n.null)) {
  phi.estimated <- mod1.boot$boot.output[[b]]$estimates$phi.hat
  boot.estimates[b,] <-  c(phi.estimated$beta,
                           phi.estimated$sigma2eps,
                           phi.estimated$sigma2omega,
                          phi.estimated$theta,
                          phi.estimated$G,
                          phi.estimated$Sigmaeta,
                          phi.estimated$m0)
}

#compute the parameter standard errors
se <- sqrt(diag(var(na.omit(boot.estimates))))

#create a summary table with Estimates, Standard Errors (SE) and T-statistics.
phi.hat <- mod1.est$estimates$phi.hat
MLE <- c(phi.hat$beta, phi.hat$sigma2eps, phi.hat$sigma2omega,
         phi.hat$theta, phi.hat$G, phi.hat$Sigmaeta,phi.hat$m0)
```

```
output1 <- cbind(MLE, se, MLE/se)
colnames(output1)<- c("Estimate", "SE", "T-stat.")
output1

#compute the 95\% confidence intervals
IC <- matrix(NA,nrow=npar,ncol=2)
for(i in 1 : npar) {
  IC[i,] <- c(quantile(boot.estimates[,i],0.025),
              quantile(boot.estimates[,i],0.975))
}


#create a summary table with Estimates, Standard Errors (SE)
#and T-statistics and confidence intervals.
output2 <- cbind(output1,IC)
colnames(output2) <- c("Estimate", "SE", "T-stat.", "IC_inf", "IC_sup")
output2
```

`STEM_Complete` *Completed response of a fitted STEM model*

Description

`STEM_Complete` returns the conditional expectation of the response given the observed data, $\mathrm{E}[z \mid \mathcal{Z}]$, for a fitted STEM model. It equals the observed value wherever the response was observed and the model-based prediction wherever it was not, so it is at once the vector of fitted values and the interpolator of the missing entries.

Usage

```
STEM_Complete(StemModel, distance = "euclidean", cov.spat = Sigmastar.exp)
```

Arguments

`StemModel`	an object returned by `STEM_Estimation`.
`distance`	character, `"euclidean"` or `"geo"`, the distance used when the model was estimated. Default is `"euclidean"`.
`cov.spat`	the spatial covariance function used when the model was estimated. Default is the exponential one.

Details

The prediction at a missing entry is *not* the signal $x'_{it}\hat{\beta} + A_i\hat{y}_t$ alone. That would be exact only if the measurement covariance were diagonal. Under the STEM specification $\Sigma_e = \sigma^2_\epsilon I + \sigma^2_\omega C_\theta$

couples the locations, so the missing block of the measurement error is predicted from the block observed at the same time point by Gaussian conditioning:

$$\hat{z}_{M,t} = x'_{M,t}\hat{\beta} + A_M \hat{y}_t + P_t \left(z_{O,t} - x'_{O,t}\hat{\beta} - A_O \hat{y}_t \right), \qquad P_t = [\Sigma_e]_{MO} \left([\Sigma_e]_{OO} \right)^{-1},$$

which is the same algebra as kriging, applied at a fixed time point. This is the quantity the EM algorithm uses to complete its sufficient statistics; see Durbin and Koopman (2012, Sections 2.7 and 4.10) for the state-space treatment of missing observations.

At a time point where nothing was observed the correction vanishes and the prediction is the signal, with the smoothed state carrying the information of the neighboring time points.

The function is the natural building block of a cross-validation exercise: blanking a set of cells, refitting, and reading this matrix at those cells gives out-of-sample predictions that never see the blanked values.

Value

A T by d numeric matrix.

Author(s)

Paolo Maranzano <pmaranzano.ricercastatistica@gmail.com>

References

Durbin, J., Koopman, S.J. (2012) *Time Series Analysis by State Space Methods*, 2nd edition. Oxford University Press, Oxford.

See Also

STEM_Estimation, SCSTEM_Complete

Examples

```
data(pm10)
phi <- list(beta = matrix(c(3.65, 0.046, -0.904), 3, 1),
            sigma2eps = 0.1, sigma2omega = 0.2, theta = 0.01,
            G = matrix(0.77, 1, 1), Sigmaeta = matrix(0.3, 1, 1),
            m0 = as.matrix(0), C0 = as.matrix(1))
z <- pm10$z
z[5, 2] <- NA                     # blank one cell
mod <- STEM_Model(z = z, covariates = pm10$covariates,
                  coordinates = pm10$coords, phi = phi,
                  A = matrix(1, ncol(z), 1))

fit <- STEM_Estimation(mod, precision = 0.05, max.iter = 5)
zhat <- STEM_Complete(fit)
zhat[5, 2]                        # the model-based prediction of the blank
```

STEM_control *Computational settings of the STEM and SC-STEM estimators*

Description

STEM_control collects the settings that govern the numerical side of the estimation – which algorithm maximizes the likelihood, when it stops, how many iterations it may take, how long the alternation of SC-STEM runs and how hard the Newton-Raphson step of the spatial parameters tries – in one object, in the spirit of the control argument of optim. The defaults are meant for an analysis; an expert user may change them, for instance to trade accuracy for speed in a simulation study.

Usage

```
STEM_control(
  algorithm = c("SQUAREM", "EM", "ECME", "SQUAREM-ECME"),
  em_tol_par = 0.001,
  em_tol_loglik = 0.001,
  em_maxit = 500L,
  em_stop = c("any", "all"),
  alt_em_tol_par = 0.01,
  alt_em_tol_loglik = 1,
  alt_em_maxit = 50L,
  alt_maxit = 10L,
  alt_abs_tol = 1e-05,
  alt_rel_tol = 1e-06,
  nr_maxit = 50L
)
```

Arguments

algorithm	the algorithm that maximizes the likelihood: "SQUAREM" (default), "EM", "ECME" or "SQUAREM-ECME". It applies to the pooled fit and to the final refits of SC-STEM; inside the alternation, whose fits stop after a few iterations, the iterations of EM or ECME are not accelerated.
em_tol_par	tolerance of the relative change of every free parameter, for the pooled fit and the final refit. Default 1e-3.
em_tol_loglik	tolerance of the absolute change of the log-likelihood, for the pooled fit and the final refit. Default 1e-3.
em_maxit	maximum number of EM iterations of the pooled fit and of every regime in the final refit. Default 500.
em_stop	how the two criteria combine, for every EM run: "any" (default), stop as soon as either is met; "all", stop only when both are met.
alt_em_tol_par, alt_em_tol_loglik, alt_em_maxit	the same three settings for the EM algorithm run inside the alternation of SC-STEM. Defaults 1e-2, 1 and 50.

`alt_maxit` maximum number of iterations of the alternation of SC-STEM. Default `10`. Zero refits the regimes of a given starting partition without moving it.

`alt_abs_tol`, `alt_rel_tol` absolute and relative tolerance on the penalized objective, below which the alternation stops. Defaults `1e-5` and `1e-6`.

`nr_maxit` maximum number of Newton-Raphson iterations of the spatial parameters within one EM iteration. Default `50`.

Details

The algorithm. `"EM"`, `"ECME"`, `"SQUAREM"` (the default) or `"SQUAREM-ECME"`; see the section "Algorithms" of `STEM_Estimation`. All four maximize the same likelihood. The EM algorithm converges linearly and slowly on the regimes of SC-STEM, where the latent process and the spatial field share most of their variation; ECME updates the regression coefficients and m_0 on the observed likelihood, which makes their estimates accurate at the stop; SQUAREM extrapolates the iterations and cuts their number.

When the EM algorithm stops. Two criteria; with `em_stop = "any"` (the default) the algorithm stops as soon as either is met, with `em_stop = "all"` only when both are met at the same iteration, and in either case after `em_maxit` iterations:

$$\max_l \frac{|\psi_l^{(i)} - \psi_l^{(i-1)}|}{\max(|\psi_l^{(i-1)}|, 10^{-3})} < \texttt{tol_par}, \qquad |\ell(\psi^{(i)}) - \ell(\psi^{(i-1)})| < \texttt{tol_loglik}.$$

The first is the relative change of every free parameter taken one at a time, so that a small parameter still moving – the persistence of the latent process, for instance – is not hidden by the large ones, as it is in the relative change of the whole parameter vector. The floor 10^{-3} keeps the criterion finite for parameters near zero. The loading matrix and the initial variance C_0, which the EM algorithm does not estimate, are excluded. The second criterion is the absolute change of the log-likelihood: a relative change of the log-likelihood depends on its scale, which grows with the size of the data, while the differences that matter – between fits, in the information criteria – are absolute. In a simulation study of SC-STEM, tolerances of 10^{-3} on both criteria left the estimates within 1 to 2% of those at 10^{-4} and the selected models unchanged, while 10^{-2} moved the range and the variances by tens of percent.

Session defaults. Every estimation function takes `control = NULL` by default, which means the defaults of `STEM_control()`, or, when the option is set, `options(Stem.control = list(...))`: a list of some of the settings, which then apply to every fit of the session that does not pass its own `control`.

Two sets of EM settings. The pooled model and the final refit of SC-STEM – the fits whose estimates and log-likelihoods are reported and enter the information criteria – use `em_tol_par`, `em_tol_loglik` and `em_maxit`. Inside the alternation of SC-STEM the EM algorithm only has to rank the regimes, and every regime starts from its estimates of the previous iteration, so looser settings suffice there: `alt_em_tol_par`, `alt_em_tol_loglik` and `alt_em_maxit`.

Value

An object of class “STEM_control”: a named list of the settings.

See Also

STEM_Estimation, SCSTEM_Estimation

Examples

```
STEM_control()
## the plain EM algorithm, with a tighter stopping rule
STEM_control(algorithm = "EM", em_tol_par = 1e-4, em_tol_loglik = 1e-4)
```

STEM_Estimation *ML Estimation*

Description

The function STEM_Estimation computes the maximum likelihood estimates of the unknown parameters of a hierarchical spatio-temporal model of class "STEM_Model". The estimates are obtained using Kalman filtering and the EM algorithm, by default accelerated by SQUAREM (see algorithm in STEM_control).

This is one of the two estimation engines of the package. The entry point is STEM_Fit, which calls this function when K = 1 and SCSTEM_Estimation when K > 1, and which takes the same arguments and returns the same object. Call STEM_Estimation directly only to bypass the dispatch; it is the historical interface of the package and is kept unchanged.

Usage

```
STEM_Estimation(
  StemModel,
  precision = NULL,
  max.iter = NULL,
  flag.Gdiag = TRUE,
  flag.Sigmaetadiag = TRUE,
  cov.spat = Sigmastar.exp,
  distance = "euclidean",
  regularization = 0,
  verbose = FALSE,
  alpha = 0,
  lambda = 0,
  penalize = NULL,
  lambda_scale = c("relative", "absolute"),
  latent = TRUE,
  spatial = TRUE,
  control = NULL
)
```

Arguments

StemModel	an object of class "STEM_Model" given as output by the STEM_Model function.
precision	optional; when given, it replaces em_tol_par of control, the tolerance of the relative change of every free parameter. Kept for the code written before control existed.
max.iter	optional; when given, it replaces em_maxit of control, the maximum number of EM iterations.
flag.Gdiag	logical, indicating whether the transition matrix G is diagonal.
flag.Sigmaetadiag	logical, indicating whether the variance-covariance matrix of the state equation Σ_η is diagonal.
cov.spat	type of spatial covariance function. For the moment only the *exponential* function is implemented.
distance	character, indicating the type of distance. 'euclidean' compute euclidean distance while 'geo' compute the geodesic distance. use 'geo' only if the coordinates format is Longitude, Latitude. Default is 'euclidean'.
regularization	a non-negative number added to the diagonal of the matrices the EM algorithm inverts (the smoother gains at the first and last time point, Σ^* in the update of σ^2_ω, Σ_e in the update of β, the blocks of the missing values and the Hessian of the Newton step). Default 0. With a positive value the iterations no longer converge to the maximum of the likelihood the filter computes; versions before 2.0.0 used 0.01, which biased the variance parameters by several percent.
verbose	logical. If TRUE, the progress of the EM and Newton-Raphson iterations is reported through message(). Default is FALSE.
alpha	the elastic-net mixing parameter for the regression coefficients, in $[0, 1]$: 0 is ridge, 1 is the lasso, anything in between is the elastic net. Ignored when lambda is zero. Default is 0.
lambda	the strength of the penalty on the regression coefficients. 0, the default, gives the ordinary maximum likelihood estimator. See STEM_Fit for the interface that selects it.
penalize	which regression coefficients the penalty acts on: a logical vector of length r, an index vector, or NULL (the default) for every coefficient except the intercept.
lambda_scale	how lambda is measured. "relative", the default, puts the L1 part of the penalty on the scale of the largest partial gradient, so that lambda is free of the units of the response and lambda * alpha >= 1 zeroes every penalized coefficient; "absolute" applies it to the scaled normal equations directly. The L2 part is unaffected: it multiplies a curvature whose diagonal is already one, and is scale free either way.
latent	logical. FALSE switches the latent process off, by setting the loading matrix to zero and holding G, Σ_η and m_0 at their input values, which they are unidentified without it. Default is TRUE.
spatial	logical. FALSE replaces the exponential correlation by the identity, so that Σ_e is a single variance and the Newton-Raphson step is skipped. Only the sum $\sigma^2_\varepsilon + \sigma^2_\omega$ is then identified, and the range has no meaning. Default is TRUE.

`control` the computational settings, an object returned by `STEM_control` or a list of some of its settings. Default `NULL`, the defaults of `STEM_control()`. This function uses `algorithm`, `em_tol_par`, `em_tol_loglik`, `em_maxit`, `em_stop` and `nr_maxit`.

Details

This function estimates the vector parameter `phi` of the hierarchical spatio-temporal model of class "STEM_Model" using Kalman filtering and EM algorithm.

Value

The function returns an object of class "STEM_Model" which is a list given by:

- skeleton As the `skeleton` component of the `StemModel` object given in input.
- data As the `data` component of the `StemModel` object given in input.
- estimatesA list of four objects: `phi.hat`, `y.smoothed`, `loglik`, `convergence.par` here described. `phi.hat` is a list with the parameter ML estimates (`sigma2omega`, `beta`, `G`, `Sigmaeta`, `m0`, `C0`, `theta`, `sigma2eps`). `y.smoothed` is a `ts` object (n by p) with the smoothed latent states at the estimates. `loglik` is the log-likelihood at the estimates. `convergence.par` is a list with some information about the convergence of the algorithm: `conv.log` and `conv.par` are logical values for the two convergence criteria described below, and `converged` says whether the stopping rule was met (either criterion with `em_stop = "any"`, both with `em_stop = "all"`) rather than the limit on the iterations reached; `max.rel.par` and `delta.loglik` are the values of the two criteria at the last check; `iterEM` is the number of iterations (EM or ECME steps, those of the extrapolations of SQUAREM included), `iterNR` the number of Newton-Raphson iterations within each, `algorithm` the algorithm used, `theta.limits` and `theta.bound` the limits of the range and whether the estimate sits at one of them (see the section on the range), and `control` the settings.

Computational form of the filter

The forward pass does not build the $d \times d$ matrix $Q_t = ZR_tZ' + \Sigma_e$ that the prediction step nominally requires, and does not invert it. Two facts make that avoidable. The measurement covariance Σ_e does not depend on t: it is rebuilt once per EM iteration. And ZR_tZ' has rank p, the dimension of the latent state, which is usually one. The Woodbury identity therefore gives

$$Q_t^{-1} = \Sigma_e^{-1} - U(R_t^{-1} + Z'\Sigma_e^{-1}Z)^{-1}U', \qquad U = \Sigma_e^{-1}Z,$$

and the matrix determinant lemma gives

$$\log|Q_t| = \log|\Sigma_e| + \log|R_t| + \log|R_t^{-1} + Z'\Sigma_e^{-1}Z|.$$

Here U, $Z'\Sigma_e^{-1}Z$, the Cholesky factor of Σ_e and $\log|\Sigma_e|$ are constants of the pass and are computed once. Every quantity the recursion needs – the gain applied to the innovation, the updated state variance, the quadratic form of the log-likelihood – then reduces to p by p algebra plus one triangular solve, so the cost per time point falls from $O(d^3)$ to $O(d^2)$ and the cost of a pass from $O(Td^3)$ to $O(d^3 + Td^2)$. The two forms are algebraically identical and agree to floating point. The identity holds for any p, and needs only that Σ_e and R_t be invertible, which positive definiteness already

guarantees; the gain is largest when p is small relative to d, which is the regime the model is written for.

With missing values the constants depend on which rows are observed, so they are computed once per distinct missingness pattern and cached.

The log-density is evaluated directly rather than as the logarithm of the density. On d observations the Gaussian density is of order e^{-d}, so on a network of a few hundred locations it falls below the smallest representable double and taking its logarithm afterwards returns `-Inf`.

Computational form of the M-step

The sums the M-step accumulates over time are written as matrix products rather than as loops over t. Because the loading matrix does not depend on time, the second moment of the measurement error collapses to

$$\sum_t \{ZC_t^s Z' + r_t r_t'\} = Z(\sum_t C_t^s)Z' + R'R,$$

with R the T by d matrix of residuals, so one cross-product replaces T outer products and the T matrices of size d by d that used to be held at once. The three sums of outer products of the smoothed states are cross-products of the T by p matrix of those states, and the two accumulations entering the update of β are cross-products of the design blocks stacked by period. Finally, every trace of a matrix product is evaluated as $tr(AB) = \sum_{ij} A_{ij}B_{ji}$, which costs $O(d^2)$ instead of forming the product. The algorithm details and formulas are given in Fasso' and Cameletti (2007, 2009). Note that some parameters (`beta`, `sigma2omega`, `G`, `Sigmaeta` and `m0`) are updated using closed form solutions while `theta` and `sigma2epsilon` using the Newton-Raphson algorithm.

Algorithms

Four algorithms are available, through `algorithm` in `STEM_control`. They reach the same maximum of the likelihood and differ in the number of iterations they take and in how close to the maximum a given stopping rule leaves them.

`"EM"` the EM algorithm: the E-step is the Kalman smoother, the M-step the sequence of conditional updates above.

`"ECME"` the ECME algorithm (Liu and Rubin 1994): the same, but β and m_0 maximize the observed log-likelihood given the other parameters, a generalized least-squares step computed by the Kalman filter run on the columns of the covariates. It removes the slow direction in which the intercept and m_0 move together, at about 25% more time per iteration.

`"SQUAREM"` (the default) the EM iterations accelerated by SQUAREM (Varadhan and Roland 2008): two EM steps are extrapolated along the direction of slowest convergence, and the extrapolation is kept only if it does not lower the log-likelihood. The steplength follows the R package SQUAREM (Du and Varadhan 2020).

`"SQUAREM-ECME"` the ECME iterations accelerated by SQUAREM.

With a ridge on β (`lambda > 0`, `alpha = 0`) the safeguard of SQUAREM compares the penalized log-likelihood, with the scale of the penalty measured at each point: the scale moves with Σ_e, so there is no fixed objective, but the accelerated iterations reach the fixed point of the plain ones. With the lasso or the elastic net (`alpha > 0`) the fit runs the plain iterations of EM or ECME.

The range parameter

The distances between the locations identify the range only within limits. When the correlation between the two closest locations falls below 0.05, the spatial field is white within the set of locations and cannot be told from the nugget; when the correlation between the two farthest exceeds 0.95, the field is constant over the locations at every time and cannot be told from an effect common to them. Beyond either limit the likelihood is flat in θ, and on a small set of locations – a regime of SC-STEM, for instance – the iterations, left free, ran along it towards 0 or infinity. θ is therefore kept within

$$-\log(0.95)/h_{\max} \leq \theta \leq -\log(0.05)/h_{\min},$$

with $h_{\min}$ and $h_{\max}$ the smallest and largest distance between two locations. A fit whose range ends at a limit reports it in `convergence.par$theta.bound` (`"lower"` or `"upper"`; it is `"none"` otherwise), with the limits in `convergence.par$theta.limits`. The Newton-Raphson step of the spatial parameters accepts a step only if it does not worsen the objective, halving it otherwise; when the Hessian is not positive definite the point moves once to the best of a grid, and a second such Hessian ends the iterations.

Starting values and stopping rule

For initializing the algorithm the values contained in `StemModel$skeleton$phi` are used as initial values. The algorithm stops when either of the following two criteria (named in the output as `conv.par` and `conv.log` respectively) is met, or, with `em_stop = "all"` in `control`, when both are

$$\max_l \frac{|\psi_l^{(i)} - \psi_l^{(i-1)}|}{\max(|\psi_l^{(i-1)}|, 10^{-3})} < \texttt{em_tol_par}$$

$$|\log L(\psi^{(i)}) - \log L(\psi^{(i-1)})| < \texttt{em_tol_loglik}$$

or after `em_maxit` iterations, with the settings taken from `control` (see STEM_control). The first criterion is the relative change of every free parameter taken one at a time, on the scale on which the algorithm updates it (the range and the ratio of the two variances on the log scale); the loading matrix and C_0, which are not estimated, are excluded. The second is the absolute change of the log-likelihood, which, unlike a relative change, does not loosen as the log-likelihood grows with the size of the data. Earlier versions of the package stopped when the relative change of the whole parameter vector, fixed loadings included, and the relative change of the log-likelihood were both below `precision`; with large log-likelihoods that rule stopped while several units could still be gained. With SQUAREM the criteria are checked at the start of every cycle, on the EM step taken from there. The estimates returned are the last iterate, and the log-likelihood and the smoothed states are computed at them.

Author(s)

Michela Cameletti < michela.cameletti@unibg.it>

References

The model estimated by this function, its EM algorithm and the parametric bootstrap are those of the following three companion works.

Fasso, A., Cameletti, M., Nicolis, O. (2007) *Air quality monitoring using heterogeneous networks*. Environmetrics, 18, 245–264. doi:10.1002/env.837

Fasso, A., Cameletti, M. (2007) *A general spatio-temporal model for environmental data*. GRASPA Technical Report n. 27, The Italian Group of Environmental Statistics. The reference that introduces this package.

Fasso, A., Cameletti, M. (2010) *A unified statistical approach for simulation, modeling, analysis and mapping of environmental data*. Simulation, 86, 139–153. doi:10.1177/0037549709102150

The algorithms:

Du, Y., Varadhan, R. (2020) *SQUAREM: An R package for off-the-shelf acceleration of EM, MM and other EM-like monotone algorithms*. Journal of Statistical Software, 92(7), 1–41. doi:10.18637/jss.v092.i07

Liu, C., Rubin, D.B. (1994) *The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence*. Biometrika, 81, 633–648. doi:10.1093/biomet/81.4.633

Varadhan, R., Roland, C. (2008) *Simple and globally convergent methods for accelerating the convergence of any EM algorithm*. Scandinavian Journal of Statistics, 35, 335–353. doi:10.1111/j.14679469.2007.00585.x

Further background:

Amisigo, B.A., Van De Giesen, N.C. (2005) *Using a spatio-temporal dynamic state-space model with the EM algorithm to patch gaps in daily riverflow series*. Hydrology and Earth System Sciences, 9, 209–224.

McLachlan, G.J., Krishnan, T. (2008) *The EM Algorithm and Extensions*, 2nd edition. Wiley, New York.

Shumway, R.H., Stoffer, D.S. (2006) *Time Series Analysis and Its Applications: with R Examples*. Springer, New York.

Xu, K., Wikle, C.K. (2007) *Estimation of parameterized spatio-temporal dynamic models*. Journal of Statistical Inference and Planning, 137, 567–588.

See Also

See Also `STEM_Model` and `pm10`

Examples

```
#load the data
data(pm10)

#extract the data
coordinates <- pm10$coords
covariates <- pm10$covariates
z <- pm10$z

#build the parameter list
#(the phi list is used for the algorithm starting values)
phi <- list(beta=matrix(c(3.65,0.046,-0.904),3,1),
            sigma2eps=0.1,
            sigma2omega=0.2,
            theta=0.01,
```

```
            G=matrix(0.77,1,1),
            Sigmaeta=matrix(0.3,1,1),
            m0=as.matrix(0),
            C0=as.matrix(1))

A <-matrix(1,ncol(z),1)

mod1 <- STEM_Model(z=z,covariates=covariates,
                   coordinates=coordinates,phi=phi,A=A)
class(mod1)

#mod1 is given as output by the STEM_Model function
mod1.est <- STEM_Estimation(mod1)
phi.estimates <- unlist(mod1.est$estimates$phi.hat)
```

`STEM_Fit` *Fit a STEM or SC-STEM model, with or without a penalty on the coefficients*

Description

One entry point for the four estimators the package provides. Which one runs is decided by two arguments and nothing else:

K	lambda	what is fitted
1	0	the pooled STEM model, `STEM_Estimation`
> 1	0	the spatially-clustered model, `SCSTEM_Estimation`
1	> 0	the pooled model with an elastic net on β
> 1	> 0	the clustered model with an elastic net on β within each regime

The defaults `K = 1`, `alpha = 0`, `lambda = 0` therefore reproduce the classical estimator exactly, to floating point.

Usage

```
STEM_Fit(
  StemModel,
  K = 1,
  alpha = 0,
  lambda = 0,
  penalize = NULL,
  lambda_scale = c("relative", "absolute"),
  lambda_by = c("common", "size"),
  latent = TRUE,
  spatial = TRUE,
  phi_penalty = 0.05,
```

```
  distance = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

StemModel	an object of class "STEM_Model", from STEM_Model.
K	number of spatial regimes. 1, the default, fits the pooled model; K > 1 fits the spatially-clustered model.
alpha	the elastic-net mixing parameter, in $[0, 1]$. 0 is ridge, 1 is the lasso, in between is the elastic net. Ignored when lambda is zero. Default is 0.
lambda	the strength of the penalty on the regression coefficients. 0, the default, gives the unpenalized maximum likelihood estimator.
penalize	which coefficients the penalty acts on: a logical vector of length r, an index vector, or NULL (the default) for every coefficient except the intercept.
lambda_scale	how lambda is measured. "relative", the default, puts the L1 part of the penalty on the scale of the largest partial gradient, so lambda is free of the units of the response and lambda * alpha >= 1 zeroes every penalized coefficient, as in glmnet. "absolute" applies it to the scaled normal equations directly. The L2 part is unaffected either way: it multiplies a curvature whose diagonal the internal scaling has already set to one, so a ridge is scale free by construction and rescaling it would only break that.
lambda_by	how the penalty is spread over the regimes when K > 1: "common" (the default) or "size". See SCSTEM_Estimation. Both keep lambda a single hyperparameter.
latent	logical. FALSE switches the latent process off.
spatial	logical. FALSE replaces the exponential correlation by the identity. Together with latent = FALSE, and with regularization at its default of 0, this reduces the model exactly to penalized linear regression – see the section below.
phi_penalty	the strength of the Potts penalty on the partition, passed to SCSTEM_Estimation. Ignored when K = 1.
distance	"geo" or "euclidean". Default is "euclidean" for the pooled model and "geo" for the clustered one, which are the defaults of the two functions being called.
verbose	logical, passed on.
...	further arguments passed to STEM_Estimation when K = 1 and to SCSTEM_Estimation when K > 1.

Details

The penalty. In the parameterization of glmnet, the objective subtracts from the expected complete-data log-likelihood

$$\lambda \left\{ \alpha \|D\beta\|_1 + \frac{1-\alpha}{2} \beta' D \beta \right\},$$

with D the diagonal indicator of the penalized coordinates. So alpha = 0 is ridge, alpha = 1 is the lasso, and anything in between is the elastic net. Only the regression coefficients are penalized: the

variance components, the range, the transition matrix and the initial state are estimated by maximum likelihood as before.

Only the M-step changes. The penalty is a function of β alone and does not involve the latent states, so the E-step is untouched and the usual argument still gives an EM algorithm that increases the *penalized* observed-data log-likelihood at every iteration. For ridge the M-step keeps its closed form, $\hat{\beta} = (M + \lambda(1-\alpha)D)^{-1}v$; for the lasso and the elastic net it is solved by cyclic coordinate descent, which converges to the exact maximizer because the objective is convex and the penalty separable.

The intercept is not penalized unless `penalize` says otherwise, and not merely by convention: the model already carries a latent process whose initial mean absorbs the level, so the intercept and m_0 are only weakly identified apart.

Scale. The accumulation $M = \sum_t X_t' \Sigma_e^{-1} X_t$ is a generalized least squares cross-product, so the penalty acts in that metric and not in the Euclidean one. The columns of the design are therefore scaled internally to put the diagonal of M at one, which is what makes a given `lambda` mean the same thing for every covariate and, in the clustered model, for every regime. Coefficients are returned on the original scale.

Choosing lambda and alpha. They are hyperparameters like `K` and `phi_penalty` and are chosen the same way: over a grid, by an information criterion computed with the *effective* number of coefficients the penalty leaves, or by spatio-temporal cross-validation. `SCSTEM_Infocrit` accepts `alpha` and `lambda` through its `...` and returns criteria already corrected for the effective degrees of freedom. Inference after selection is not valid naively: if `lambda` is chosen from the data, the parametric bootstrap has to repeat the selection on every draw, exactly as `SCSTEM_Bootstrap` repeats the clustering.

The derivation – the penalized EM and its monotonicity, the M-step in both forms, the generalized least squares metric, the effective degrees of freedom and what is still open – is kept as a standalone document in the repository, at `dev/regularization/stem-elastic-net.tex`, since the material is a study of its own rather than documentation of the software.

Value

The object the underlying function returns: of class "STEM_Model" when `K = 1`, of class "SCSTEM_Estimation" when `K > 1`. In both cases the penalty in force and the effective number of coefficients are recorded, under `estimates$penalty` and inside each regime's fit respectively.

Penalized linear regression as a special case

Two switches take the model down to an ordinary regression. `latent = FALSE` sets the loading matrix to zero, so the state contributes nothing and G, Σ_η and m_0, which are then unidentified, are held where they started. `spatial = FALSE` replaces the exponential correlation by the identity, so $\Sigma_e = \sigma^2 I$ and the Newton-Raphson step, which would be estimating a range that has nothing to estimate, is skipped. What is left is

$$z_{ti} = x_{ti}'\beta + e_{ti}, \qquad e \sim N(0, \sigma^2 I),$$

estimated by penalized least squares, provided `regularization` is left at its default of 0: a ridge added for conditioning would be the only thing separating the two, and on a collinear design it is not negligible.

```
fit <- STEM_Fit(mod, alpha = 0.5, lambda = 0.3,
                latent = FALSE, spatial = FALSE)
```

On a design with a deliberately collinear pair this agrees with the elastic net computed directly on $X'X$ and $X'y$ to between 10^{-14} and 10^{-11}, for the ridge, the lasso, the elastic net and the unpenalized case alike. With `K > 1` the same switches give clusterwise penalized regression, the partition still estimated.

How the penalty and the clustering interact

With `K > 1` the objective carries two penalties doing different things: `phi_penalty` penalizes disagreement between the labels of neighboring locations, `lambda` penalizes the size of the coefficients within a regime. They are not applied one after the other. The algorithm alternates

1. a *parameter step*, in which the penalized EM runs inside each regime, so that $\hat{\Psi}_g$ is the penalized estimate; and
2. a *label step*, an ICM sweep in which each location takes the label maximizing its own contribution plus the Potts term.

The penalty $\sum_g \mathrm{pen}(\beta_g)$ is constant with respect to the labels at fixed parameters, so it drops out of the arg-max of the label step: the form of that step is exactly what it was without a penalty. The penalty reaches the clustering only *through the parameters the labels are scored against* – which is enough to move the partition, since shrinking the coefficients makes regimes look more alike.

Three consequences are worth knowing before choosing the four hyperparameters.

They are not separable. The `lambda` minimizing predictive risk grows with `K`, because a regime of d/K locations carries less information than the pooled network; and the `K` a criterion selects grows with `lambda`, because shrinkage lowers the effective number of coefficients and so makes an extra regime cheaper. Fixing `lambda` once on the pooled fit and only then selecting `K` under-shrinks precisely where shrinkage is needed.

The automatic scale of the Potts penalty moves with lambda. With `phi_scale = "auto"` the multiplier is calibrated from the spread of the per-location log-likelihood contributions across regimes, and that spread narrows as `lambda` grows. A given `phi_penalty` therefore does not mean the same thing at two different values of `lambda`; use `phi_scale = "raw"` when a fixed meaning is needed across a grid.

The criteria use the unpenalized log-likelihood. `SCSTEM_Infocrit` reports the exact log-likelihood evaluated at the penalized estimate, charged with the *effective* number of coefficients. Subtracting the penalty from the log-likelihood as well would charge for it twice, once through the objective and once through the degrees of freedom.

References

Zou, H., Hastie, T. (2005) *Regularization and variable selection via the elastic net*. Journal of the Royal Statistical Society B, 67, 301–320. doi:10.1111/j.14679868.2005.00503.x

Zou, H., Hastie, T., Tibshirani, R. (2007) *On the degrees of freedom of the lasso*. The Annals of Statistics, 35, 2173–2192. doi:10.1214/009053607000000127

Otto, P., Fasso, A., Maranzano, P. (2024) *A review of regularized estimation methods and cross-validation in spatiotemporal statistics*. Statistics Surveys, 18, 299–340. doi:10.1214/24SS150

See Also

STEM_Estimation, SCSTEM_Estimation, SCSTEM_Infocrit, SCSTEM_Select

Examples

```
data(povalley)
Tn <- 120L
Tfull <- nrow(povalley$z)
d  <- nrow(povalley$coords)
z  <- povalley$z[seq_len(Tn), ]
# rows of the stacked covariate matrix belonging to the first Tn days:
# the intercept, the station altitude and the daily PM10 concentration
keep <- as.vector(outer(seq_len(Tn), (seq_len(d) - 1L) * Tfull, '+'))
X  <- povalley$covariates[keep, ]
ok  <- !is.na(as.vector(z))
ols <- stats::lm.fit(x = X[ok, ], y = as.vector(z)[ok])
s2  <- stats::var(ols$residuals)
mod <- STEM_Model(z = z, covariates = X, coordinates = povalley$coords,
  phi = list(beta = matrix(ols$coefficients, ncol = 1),
             sigma2eps = 0.7 * s2, sigma2omega = 0.3 * s2,
             theta = 1 / 100000, G = matrix(0.8, 1, 1),
             Sigmaeta = matrix(0.2 * s2, 1, 1),
             m0 = as.matrix(0), C0 = as.matrix(1)),
  A = matrix(1, d, 1))

## the classical estimator
fit0 <- STEM_Fit(mod, distance = "geo", max.iter = 5)

## the same model with a ridge on the coefficients
fitr <- STEM_Fit(mod, alpha = 0, lambda = 1, distance = "geo", max.iter = 5)
```

STEM_Kriging *Dynamical spatial mapping*

Description

This functions performs spatial prediction in a set of new S spatial locations for a fixed time point.

Usage

```
STEM_Kriging(
  StemModel,
  coord.newlocations,
  covariates.newlocations,
  A.newlocations,
  time.point,
  cov.spat = Sigmastar.exp,
```

```
  distance = "euclidean"
)
```

Arguments

StemModel	an object of class "STEM_Model" given as output by the STEM_Model function.
coord.newlocations	a matrix or a data frame of dimension S by 2.
covariates.newlocations	a matrix of dimension S by r, where r is the number of covariates as in the StemModel object. It has the same structure of the StemModel$data$covariates object in the sense that each station data set is stacked under the others; see the DETAILS of the STEM_Model function.
A.newlocations	a loading matrix of dimension S by p.
time.point	the time point between 1 and n for which the spatial prediction is performed.
cov.spat	type of spatial covariance function. For the moment only the *exponential* function is implemented.
distance	character, indicating the type of distance. 'euclidean' compute euclidean distance while 'geo' compute the geodesic distance. use 'geo' only if the coordinates format is Longitude, Latitude. Default is 'euclidean'.

Details

Given the observation matrix and using the multivariate Normal distribution standard theory, the predictor in the new generic spatial location s_0 at time t is an univariate Gaussian distribution with mean $z(s_0, t)$ and variance $\sigma^2(s_0)$ given by:

$$z(s_0, t) = X(s_0, t)\beta + A(s_0)y_t + \Omega'\Sigma_e^{-1}(z_t - X_t\beta - Ay_t)$$

$$\sigma^2(s_0) = \sigma^2\omega - \Omega'\Sigma_e^{-1}\Omega$$

where Ω is the $d \times 1$ constant in time covariance vector, whose i-th generic element ($i = 1, ..., d$) is $Cov(z(s_i, t), z(s_0, t))$. Moreover, $X(s_0, t)$ is the $1 \times r$ vector of covariates for the new site s_0 and $A(s_0)$ is the $1 \times p$ loading vector. Note that all the parameters in the previous formula are ML estimates and the latent process y_t is the output of the Kalman filtering procedures for each time point t.

The predictor is the conditional mean of the response at s_0 given the data observed at time t, with the latent state at its smoothed value. The standard error se.pred is $\sigma(s_0)$ above, the formula of Fasso and Cameletti (2010): it is the standard error of the spatially structured component at s_0, and it leaves out the nugget σ_ϵ^2 and the uncertainty of the smoothed state, so it understates the standard error of a new observation at s_0.

Value

The function returns a list which is given by:

- data.newlocations a list of five objects related to the new spatial locations: the coordinates (`coordinates`), the covariates (`covariates`), the loading matrix `A`, the predictions `z` and the prediction standard errors (`se.pred`).
- time.pointthe time point for which the spatial prediction is performed.

Author(s)

Michela Cameletti < michela.cameletti@unibg.it >

References

Amisigo, B.A., Van De Giesen, N.C. (2005) *Using a spatio-temporal dynamic state-space model with the EM algorithm to patch gaps in daily riverflow series*. Hydrology and Earth System Sciences 9, 209–224. Fasso, A., Cameletti, M., Nicolis, O. (2007) *Air quality monitoring using heterogeneous networks*. Environmetrics 18, 245–264. Fasso', A., Cameletti, M. (2007) *A general spatio-temporal model for environmental data*. Tech.rep. n.27 *Graspa* - The Italian Group of Environmental Statistics. Fasso, A. and M. Cameletti (2010). *A Unified Statistical Approach for Simulation, Modeling, Analysis and Mapping of Environmental Data*. SIMULATION 86(3): 139-153. <doi: 10.1177/0037549709102150> Mc Lachlan, G.J., Krishnan, T. (1997) *The EM Algorithm and Extensions*. Wiley, New York. Shumway, R.H., Stoffer, D.S. (2006) *Time Series Analysis and Its Applications: with R Examples*. Springer, New York. Xu, K., Wikle, C.K. (2007) *Estimation of parameterized spatio-temporal dynamic models*. Journal of Statistical Inference and Planning 137, 567–588.

See Also

See Also STEM_Model, pm10 and STEM_Estimation

Examples

```
#load the data
data(pm10)

#extract the data
coordinates <- pm10$coords
covariates <- pm10$covariates
z <- pm10$z

#build the parameter list
#(the phi list is used for the algorithm starting values)
phi <- list(beta=matrix(c(3.65,0.046,-0.904),3,1),
            sigma2eps=0.1,
            sigma2omega=0.2,
            theta=0.01,
            G=matrix(0.77,1,1),
            Sigmaeta=matrix(0.3,1,1),
            m0=as.matrix(0),
            C0=as.matrix(1))

A<-matrix(1,ncol(z),1)
```

```
mod1 <- STEM_Model(z=z,covariates=covariates,coordinates=coordinates,phi=phi,A=A)
class(mod1)

#mod1 is given as output by the STEM_Model function

mod1.est <- STEM_Estimation(mod1)

#coordinates of the 25 new points displaced in a regular grid (S=25)
xxx  <- seq(400,470,length=5)
yyy <- seq(5000,5070,length=5)
coord.new <- expand.grid(x=xxx,y=yyy)

#plot of the spatial locations
plot(pm10$coords[,1],pm10$coords[,2],xlab=colnames(pm10$coords)[1],
     ylab=colnames(pm10$coords)[2])
points(coord.new[,1],coord.new[,2],col=2,pch=19)
legend("topleft",col=c(1,2),lty=c(0,0), pch=c(21,19),
       legend=c("Original spatial locations","New spatial locations"))

#the covariates matrix for the new 25 spatial locations for the 10th time point
covariates.new <- cbind(rep(1,25),
                        c(37.98348, 18.14824, 15.32287, 11.00458, 6.67696,
                          29.120820, 10.487590, 2.401088, 26.112971, 1.683525,
                          19.211907, 31.363448, 3.629172, 10.352472, 48.289624,
                          7.199692, 3.524810, 25.546621, 19.598600, 10.521586,
                       0.004736363, 0.365510044, 0.975484255, 25.523642458, 4.671496566),
                        c(0.227688, 0.173037, 0.139985, 0.116392, 0.102476,
                          0.278325, 0.256422, 0.168136, 0.129460, 0.121040,
                          0.722656, 0.238780, 0.202586, 0.166547, 0.154638,
                          0.733208, 1.467990, 0.380001, 0.251896, 0.240350,
                          2.292299 ,2.275844 ,1.382322, 0.300729, 0.208798))

A.new<-matrix(1,25,1)

#dynamical spatial prediction (10th day)
mod1.pred <-STEM_Kriging(StemModel=mod1.est,coord.newlocations=coord.new,
                         covariates.newlocations=covariates.new,
                         A.newlocations<-A.new,time.point=10)

#post-processing: build an image map
image(x=xxx,y=yyy,z=matrix(mod1.pred$data.newlocations$z,
                           length(xxx),length(yyy)),
      xlab=colnames(pm10$coords)[1],ylab=colnames(pm10$coords)[2],
      xlim=range(mod1.est$data$coordinates[,1])+5,
      ylim=range(mod1.est$data$coordinates[,2]+5))
points(pm10$coords[,1],pm10$coords[,2])
points(coord.new[,1],coord.new[,2],col=2,pch=19)

byline <- min((range(xxx)[2]-range(xxx)[1])/4,(range(yyy)[2]-range(yyy)[1])/4)
abline(v=seq(range(xxx)[1],range(xxx)[2],by=byline),col="grey",lty=2)
abline(h=seq(range(yyy)[1],range(yyy)[2],by=byline),col="grey",lty=2)
```

STEM_Model *Create a Stem Model*

Description

The function STEM_Model is used to create an object of class "STEM_Model".

Usage

```
STEM_Model(...)
```

Arguments

...	A list with named elements: phi, A, z, coordinates, covariates, and optionally, p (default is 1). See model details below.

Details

The hierarchical spatio-temporal model is defined as:

$$z_t = X_t\beta + Ay_t + e_t, e_t \sim N(0, \Sigma_e)$$

$$y_t = Gy_{t-1} + \eta_t, \eta_t \sim N(0, \Sigma_\eta)$$

for $t = 1, ..., n$. The initialization follows: $y_0 \sim N(m0, C0)$.

The observation vector z_t has dimension $d \times 1$, where d is the number of spatial locations. The latent process y_t has dimension $p \times 1$. The matrix X_t represents known covariates and has dimension $d \times r$, where r is the number of covariates.

The spatial covariance matrix Σ_e is defined as:

$$\Sigma_e = \sigma_\epsilon^2 I + \sigma_\omega^2 C(h, \theta)$$

where $C(h, \theta)$ is an exponential covariance function:

$$C(h, \theta) = \exp(-\theta h).$$

The parameter vector ϕ consists of β, σ_ϵ^2, σ_ω^2, θ, G, Σ_η, and $m0$ (assuming $C0$ is fixed).

The input list must have:

phi A list containing:

- beta ($r \times 1$)
- sigma2eps (scalar)
- sigma2omega (scalar)
- theta (scalar)
- G ($p \times p$)
- Sigmaeta ($p \times p$)
- m0 ($p \times 1$)

`C0` ($p \times p$)

`A` Loading matrix ($d \times p$).

`z` Observation matrix ($n \times d$).

`coordinates` Matrix ($d \times 2$) with spatial coordinates.

`covariates` Matrix ($(n \times d) \times r$) stacking data by station.

Value

The function returns a list containing:

- `skeleton` A list with components `phi`, `p`, and `A` as provided in the input.
- `data` A list with components `z`, `coordinates`, `covariates` from the input, as well as `r`, `n`, and `d`.

Note

Missing values are allowed in the response `z`, and are handled by the EM algorithm as prescribed by Durbin and Koopman (2012, 2nd ed.), Sections 2.7 and 4.10: at each time point the measurement equation is restricted to the locations actually observed, and a time point at which nothing is observed contributes no update and no likelihood term. The panel may therefore be unbalanced. Every location must retain at least one observation.

No missing values are allowed in `covariates` or `coordinates`: the design matrix enters the closed-form M-step updates directly and the coordinates enter the distance matrix, so gaps there would require a stochastic E-step. Impute them before fitting.

Author(s)

Michela Cameletti <michela.cameletti@unibg.it>

References

- Fasso', A., Cameletti, M. (2007). *A general spatio-temporal model for environmental data*. Graspa Technical Report 27.
- McLachlan, G.J., Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley.
- Shumway, R.H., Stoffer, D.S. (2006). *Time Series Analysis and Its Applications: with R Examples*. Springer.

See Also

`pm10`

Examples

```
# Load data
data(pm10)
names(pm10)

# Extract data
```

```
coordinates <- pm10$coords
covariates <- pm10$covariates
z <- pm10$z

# Define model parameters
phi <- list(beta = matrix(c(3.65, 0.046, -0.904), 3, 1),
            sigma2eps = 0.1,
            sigma2omega = 0.2,
            theta = 0.01,
            G = matrix(0.77, 1, 1),
            Sigmaeta = matrix(0.3, 1, 1),
            m0 = as.matrix(0),
            C0 = as.matrix(1))

A <- matrix(1, ncol(z), 1)

# Create model
mod1 <- STEM_Model(z = z, covariates = covariates,
                   coordinates = coordinates, phi = phi, A = A)

class(mod1)
```

STEM_Signal *Fitted values of a STEM model*

Description

STEM_Signal returns the fitted values of a STEM model in the sense the word usually carries: the conditional mean of the response given the latent path,

$$\hat{\mu}_{ti} = x'_{ti}\hat{\beta} + A_i\hat{y}_t,$$

with $\hat{y}_t$ the smoothed state. It is the systematic part of the measurement equation — the regression surface plus the latent process — and it excludes the measurement error, which is what makes it an estimate of something rather than a copy of the data.

Usage

```
STEM_Signal(StemModel)
```

Arguments

StemModel	a fitted model of class "STEM_Model", as returned by STEM_Estimation or STEM_Fit.

Details

This is not `STEM_Complete`**, and the difference matters.** `STEM_Complete()` returns $E[z \mid z_{obs}]$: it fills the gaps of an incomplete record and, wherever the response *was* observed, it returns the observation itself, because the conditional expectation of something you have seen is that thing. On a complete record `STEM_Complete()` is therefore the data, and scoring it against anything measures nothing. Use it to impute; use `STEM_Signal()` to fit.

In state-space language $A_i \hat{y}_t$ is the signal, and the function is named for it; the regression term is added because in this model the systematic part is the sum of the two.

What to score it against. Against the response z the residual contains the measurement error, which no model can remove, so an in-sample comparison between two specifications is compressed by a constant that belongs to neither. In a simulation the honest target is the true conditional mean; on real data it is the response at held-out cells, with a blocking scheme that respects both dependencies — see Otto, Fasso and Maranzano (2024).

Value

A T by d matrix of fitted values, with the dimnames of the response.

References

Durbin, J., Koopman, S. J. (2012) *Time Series Analysis by State Space Methods*, 2nd edition. Oxford University Press.

Otto, P., Fasso, A., Maranzano, P. (2024) *A review of regularized estimation methods and cross-validation in spatiotemporal statistics*. Statistics Surveys, 18, 299–340. doi:10.1214/24SS150

See Also

`STEM_Complete` for the imputation of missing responses, `STEM_Kriging` for prediction at unobserved locations, `SCSTEM_Signal` for the clustered model.

`STEM_Simulation` *Simulation of spatio-temporal data*

Description

The function `STEM_Simulation` simulates spatio-temporal data. of a hierarchical spatio-temporal model of class "STEM_Model". The estimates are obtained using Kalman filtering and EM algorithm.

Usage

```
STEM_Simulation(StemModel, distance = "euclidean")
```

Arguments

StemModel	an object of class "STEM_Model" given as output by the STEM_Model function.
distance	character, indicating the type of distance. 'euclidean' compute euclidean distance while 'geo' compute the geodesic distance. use 'geo' only if the coordinates format is Longitude, Latitude. Default is 'euclidean'.

Details

Note that the values contained in StemModel$skeleton$phi are used as the true values of the parameters.

Value

The functions return a $n \times d$ matrix of data.

Author(s)

Michela Cameletti < michela.cameletti@unibg.it >

References

Amisigo, B.A., Van De Giesen, N.C. (2005) *Using a spatio-temporal dynamic state-space model with the EM algorithm to patch gaps in daily riverflow series*. Hydrology and Earth System Sciences 9, 209–224. Fasso, A., Cameletti, M., Nicolis, O. (2007) *Air quality monitoring using heterogeneous networks*. Environmetrics 18, 245–264. Fasso', A., Cameletti, M. (2007) *A general spatio-temporal model for environmental data*. Tech.rep. n.27 *Graspa* - The Italian Group of Environmental Statistics. Fasso, A. and M. Cameletti (2010). *A Unified Statistical Approach for Simulation, Modeling, Analysis and Mapping of Environmental Data*. SIMULATION 86(3): 139-153. <doi: 10.1177/0037549709102150> Mc Lachlan, G.J., Krishnan, T. (1997) *The EM Algorithm and Extensions*. Wiley, New York. Shumway, R.H., Stoffer, D.S. (2006) *Time Series Analysis and Its Applications: with R Examples*. Springer, New York. Xu, K., Wikle, C.K. (2007) *Estimation of parameterized spatio-temporal dynamic models*. Journal of Statistical Inference and Planning 137, 567–588.

See Also

See Also STEM_Model and pm10

Examples

```
data(pm10)
names(pm10)

#extract the data
coordinates <- pm10$coords
covariates <- pm10$covariates
z <- pm10$z

#build the parameter list
```

```
phi <- list(beta=matrix(c(3.65,0.046,-0.904),3,1),
            sigma2eps=0.1,
            sigma2omega=0.2,
            theta=0.01,
            G=matrix(0.77,1,1),
            Sigmaeta=matrix(0.3,1,1),
            m0=as.matrix(0),
            C0=as.matrix(1))

A <-matrix(1,ncol(z),1)

mod1 <- STEM_Model(z=z,covariates=covariates,
                   coordinates=coordinates,phi=phi,A=A)

class(mod1)

simulateddata = STEM_Simulation(mod1)
```

Index

∗ **datasets**
  pm10, 3
  povalley, 4
∗ **models**
  SCSTEM_BootInference, 9
  SCSTEM_Bootstrap, 11
  SCSTEM_Complete, 14
  SCSTEM_CV, 15
  SCSTEM_Estimation, 17
  SCSTEM_Infocrit, 25
  SCSTEM_Select, 27
  STEM_Bootstrap, 31
  STEM_Complete, 34
  STEM_Estimation, 38
  STEM_Kriging, 48
  STEM_Model, 52
  STEM_Simulation, 55
∗ **spatial**
  SCSTEM_BootInference, 9
  SCSTEM_Bootstrap, 11
  SCSTEM_Complete, 14
  SCSTEM_CV, 15
  SCSTEM_Estimation, 17
  SCSTEM_Infocrit, 25
  SCSTEM_Select, 27
  STEM_Bootstrap, 31
  STEM_Complete, 34
  STEM_Estimation, 38
  STEM_Kriging, 48
  STEM_Model, 52
  STEM_Simulation, 55

optim, *36*

pm10, 3, *4, 5*, *24*, *32*, *43*, *50*, *53*, *56*
povalley, 4
print.SCSTEM_BootInference, 5
print.SCSTEM_Bootstrap, 6
print.SCSTEM_CV, 7
print.SCSTEM_Estimation, 7
print.SCSTEM_Infocrit, 8
print.SCSTEM_Select, 8

SCSTEM_BootInference, 9, *12, 13*
SCSTEM_Bootstrap, *9, 10*, 11, *24*, *46*
SCSTEM_Complete, 14, *16*, *30*, *35*
SCSTEM_CV, 15
SCSTEM_Estimation, *5*, *10–15*, 17, *25–30*, *38*, *44, 45*, *48*
SCSTEM_Infocrit, *19–22*, *24*, 25, *27–29*, *46–48*
SCSTEM_Select, *15, 16*, *24–26*, 27, *27*, *48*
SCSTEM_Signal, 30, *55*
STEM_Bootstrap, *13*, 31
STEM_Complete, *14*, *16*, 34, *55*
STEM_control, *20*, *31*, 36, *38*, *40–42*
STEM_Estimation, *5*, *17*, *19–24*, *31, 32*, *34, 35*, *37*, 38, *38*, *44, 45*, *48*, *50*, *54*
STEM_Fit, *17*, *19*, *30*, *38, 39*, 44, *54*
STEM_Kriging, 48, *55*
STEM_Model, *5*, *18*, *24, 25*, *31, 32*, *39*, *43*, *45*, *49, 50*, 52, *56*
STEM_Signal, *30*, 54
STEM_Simulation, *12*, 55