Package 'glmSTARMA'

January 26, 2026

Type Package

Title (Double) Generalized Linear Models for Spatio-Temporal Data

Version 1.0.0

Date 2026-01-14

Author Steffen Maletz [aut, cre] (ORCID:
<https://orcid.org/0009-0004-4851-0947>),
Konstantinos Fokianos [aut] (ORCID:
<https://orcid.org/0000-0002-0051-711X>),
Roland Fried [aut] (ORCID: <https://orcid.org/0000-0002-9830-9713>),
Valerie Weismann [ctb]

Maintainer Steffen Maletz <maletz@statistik.tu-dortmund.de>

Description Fit spatio-temporal models within a (double) generalized linear modelling framework. The package includes functions for estimation, simulation and inference.

URL https://github.com/stmaletz/glmSTARMA

BugReports https://github.com/stmaletz/glmSTARMA/issues

Imports Rcpp (>= 1.0.10), copula, nloptr (>= 1.2.0)

LinkingTo Rcpp, RcppArmadillo, roptim, nloptr (>= 1.2.0)

Suggests Matrix, testthat, spdep

Depends R (>= 4.5.0)

SystemRequirements C++17

License GPL (>= 3)

ByteCompile true

NeedsCompilation yes

Encoding UTF-8

RoxygenNote 7.3.3

Repository CRAN

Date/Publication 2026-01-26 16:30:07 UTC

Contents

`glmSTARMA-package` *glmSTARMA: (Double) Generalized Linear Models for Spatio-Temporal Data*

Description

Fit spatio-temporal models within a (double) generalized linear modelling framework. The package includes functions for estimation, simulation and inference.

Details

The implemented models are based on spatio-temporal autoregressive moving average (STARMA) models. They incorporate spatial and temporal dependencies by spatial lagging, via spatial weight matrices, and temporal lagging via past observations and past values of the linear predictor.

The main functions for fitting such models are `glmstarma` and `dglmstarma`. The main difference between the two functions is that `glmstarma` fits a model for the (conditional) mean of the spatio-temporal process and `dglmstarma` fits two models, one for the (conditional) mean and another one for the (conditional) dispersion. The mean model in both functions generalizes the structure of

spatio-temporal Poisson autoregressions, and allows for various distributions from the exponential dispersion family. The dispersion model can be seen as an generalization of an spatio-temporal GARCH or log-GARCH model. Data can be simulated with `glmstarma.sim` and `dglmstarma.sim`.

For more details on the models see the documentation of the fitting functions `glmstarma` and `dglmstarma`.

Author(s)

Maintainer: Steffen Maletz <maletz@statistik.tu-dortmund.de> (ORCID)

Authors:

- Konstantinos Fokianos <fokianos@ucy.ac.cy> (ORCID)
- Roland Fried <fried@statistik.tu-dortmund.de> (ORCID)

Other contributors:

- Valerie Weismann [contributor]

References

- Armillotta, M., Tsagris, M., & Fokianos, K. (2024). Inference for Network Count Time Series with the R Package PNAR. The R Journal, 15(4), 255–269. doi:10.32614/RJ2023094
- Barreto-Souza, W., Piancastelli, L. S., Fokianos, K., & Ombao, H. (2025). Time-Varying Dispersion Integer-Valued GARCH Models. Journal of Time Series Analysis. doi:10.1111/jtsa.12838
- Cliff, A. D., & Ord, J. K. (1975). Space-Time Modelling with an Application to Regional Forecasting. Transactions of the Institute of British Geographers, 64, 119–128. doi:10.2307/621469
- Jahn, M., Weiß, C.H., Kim, H.Y. (2023), Approximately linear INGARCH models for spatio-temporal counts, Journal of the Royal Statistical Society Series C: Applied Statistics, 72(2), 476-497, doi:10.1093/jrsssc/qlad018
- Jørgensen, B. (1987), Exponential Dispersion Models. Journal of the Royal Statistical Society: Series B (Methodological), 49(2), 127-145. doi:10.1111/j.25176161.1987.tb01685.x
- Knight, M., Leeming, K., Nason, G., & Nunes, M. (2020). Generalized Network Autoregressive Processes and the GNAR Package. Journal of Statistical Software, 96(5), 1–36. doi:10.18637/jss.v096.i05
- Maletz, S., Fokianos, K., & Fried, R. (2024). Spatio-Temporal Count Autoregression. Data Science in Science, 3(1). doi:10.1080/26941899.2024.2425171
- Meyer, S., Held, L., & Höhle, M. (2017). Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance. Journal of Statistical Software, 77(11), 1–55. doi:10.18637/jss.v077.i11
- Otto, P. (2024). A multivariate spatial and spatiotemporal ARCH Model. Spatial Statistics, 60. doi:10.1016/j.spasta.2024.100823
- Pfeifer, P. E., & Deutsch, S. J. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling Phillip. Technometrics, 22(1), 35–47. doi:10.2307/1268381
- Smyth, G.K. (1989), Generalized Linear Models with Varying Dispersion. Journal of the Royal Statistical Society: Series B (Methodological), 51(1), 47-60. doi:10.1111/j.25176161.1989.tb01747.x

See Also

Useful links:

- https://github.com/stmaletz/glmSTARMA
- Report bugs at https://github.com/stmaletz/glmSTARMA/issues

---

`chickenpox` *Chickenpox Infections in Hungary*

Description

Multivariate count time series consisting of weekly chickenpox infections in the districts of Hungary.

Format

chickenpox A matrix with counts of chickenpox infections (rows = districts, columns = time points).

W_hungary A list of matrices containing spatial weight matrices:

1. Identity matrix.
2. Row-normalized adjacency matrix of the districts.

population_hungary A numeric matrix containing the population per 10000 inhabitants of each district over time.

Details

This dataset contains chickenpox counts in the 20 districts (NUTS 3) of Hungary over a time period of 522 weeks (from 2005 to 2014).

The row-normalized adjacency matrix indicates which districts share a common border.

The population data is only availabyle on a yearly basis and has been linearly interpolated by us to obtain weekly estimates.

The dataset is not included directly in the package. Use `load_data("chickenpox")` to download it.

Source

The data originate from the UCI Machine Learning Repository and the Hungarian Central Statistical Office and are licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license:

- https://archive.ics.uci.edu/dataset/580/hungarian+chickenpox+cases
- https://www.ksh.hu/stadat_files/nep/en/nep0034.html

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

covariates <- list(population = population_hungary,
                   season_cos = SpatialConstant(cos(2 * pi / 52 * 1:522)),
                   season_sin = SpatialConstant(sin(2 * pi / 52 * 1:522)))
glmstarma(chickenpox, list(past_obs = 1), wlist = W_hungary,
          covariates = covariates, family = vpoisson("log"))
glmstarma(chickenpox, list(past_obs = 1), wlist = W_hungary,
          covariates = covariates, family = vnegative.binomial("log"))

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
           dispersion_link = "log", wlist = W_hungary, mean_covariates = covariates)
```

---

coef.glmstarma *Extract Coefficients of glmstarma and dglmstarma Models*

Description

Extracts model coefficients from objects of class `glmstarma` and `dglmstarma`.

Usage

```
## S3 method for class 'glmstarma'
coef(object, asList = FALSE)

## S3 method for class 'dglmstarma'
coef(object, asList = FALSE)
```

Arguments

`object`	An object of class `glmstarma` or `dglmstarma`
`asList`	Logical; if `TRUE`, returns coefficients as a list, or otherwise as a numeric vector. Default is `FALSE`.

Value

A numeric vector, or a list, of model coefficients.

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
coef(fit)
coef(fit, asList = TRUE)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
coef(fit2)
coef(fit2, asList = TRUE)
```

---

delete_glmSTARMA_data *Delete cached example datasets*

Description

Delete one or more cached example datasets downloaded via `load_data()`.

Usage

```
delete_glmSTARMA_data(name = NULL)
```

Arguments

`name`	Name(s) of the dataset(s) to delete. One or more of `"rota"`, `"chickenpox"`, or `"sst"`. If `NULL` (default), no action is taken.

Details

This function deletes datasets that were previously downloaded and cached using `load_data()` from the user-specific data directory. If no datasets are found in the cache, a message is printed and no action is taken.

Value

Invisibly returns `TRUE` if all specified datasets were deleted, `FALSE` otherwise.

See Also

load_data, rota, chickenpox, sst

Examples

```
delete_glmSTARMA_data("chickenpox")  # Only gives a message if dataset is not cached
```

---

dglmstarma *Fit STARMA Models based on double generalized linear models*

Description

The function dglmstarma estimates a multivariate time series model based on double generalized linear models (DGLM) introduced by Smyth (1989). The primary application is for spatio-temporal data, but different applications, such as network data, are also feasible. Conditionally on the past, each component of the multivariate time series is assumed to follow a distribution from the exponential dispersion family, see Jørgensen (1987). In contrast to standard generalized linear models, the dispersion parameter of the distribution is allowed to vary. The model framework links the mean of the time series conditional on the past, to a linear predictor. This linear predictor allows regression on past observations, past values of the linear predictor and covariates, as described in the details. Additionally, the dispersion parameter of the distribution is modeled with an additional linear predictor, which can also include spatial and temporal dependencies as well as covariates. Various distributions with several link-functions are available.

Usage

```
dglmstarma(
  ts,
  mean_model = list(),
  dispersion_model = list(),
  mean_family = NULL,
  dispersion_link = c("log", "identity", "inverse"),
  wlist = NULL,
  mean_covariates = list(),
  dispersion_covariates = list(),
  pseudo_observations = c("deviance", "pearson"),
  wlist_past_mean = NULL,
  wlist_covariates = NULL,
  wlist_pseudo_obs = NULL,
  wlist_past_dispersion = NULL,
  wlist_covariates_dispersion = NULL,
  control = list()
)
```

Arguments

`ts` Multivariate time series. Rows indicate the locations and columns the time.

`mean_model` a named list specifying the model orders of the linear predictor, which can be of the following elements:

- `intercept` : (Optional) character
  - `'homogenous'` (default) for a homogenous model, i.e. the same intercept for all components
  - `'inhomogenous'` for inhomogenous models, i.e. fitting an individual intercept for each component
- `past_obs` : (Optional)
  - Integer vector with the maximal spatial orders for the time lags in `past_obs_time_lags`.
  - Alternatively: a binary matrix, with the entry in row i and column j indicating whether the $(i-1)$-spatial lag for the j-th time lag is included in the model.
- `past_obs_time_lags` : (Optional) integer vector
  - indicates the time lags for `past_obs`. Defaults to `seq(length(past_obs))` (for vectors) and `seq(ncol(past_obs))` (for a matrix)
- `past_mean` : (Optional)
  - Spatial orders for the regression on past values of (latent) linear process values.
  - Values can be entered in the same format as in `past_obs`. If not specified, no regression to the feedback process is performed.
- `past_mean_time_lags` : (Optional) integer vector
  - Time lags for the regression on the (latent) linear process. Values can be entered in the same format as in `past_obs_time_lags`.
- `covariates` : (Optional)
  - spatial orders for the covariate processes passed in the argument `covariates`. The values can be passed as in `past_obs` and `past_means`, where the j-th entry or column represents the j-th covariate.
  - Default is spatial order 0 for all covariates, which corresponds to the first matrix in argument `wlist_covariates`.

`dispersion_model` a named list specifying the model orders of the dispersion linear predictor, which can have the same elements as the `mean_model` argument.

`mean_family` A list generated by one of the family functions of this package, see `stfamily`. This argument specifies the marginal conditional distributions of the observations and the type of model fitted for the mean linear predictor.

`dispersion_link` Link function that is used for the dispersion model. Available options are "log" (default), "identity" and "inverse".

`wlist` A list of quadratic matrices, with the same dimension as the time series has rows, which describe the spatial dependencies. Row-normalized matrices are recommended. See Details.

`mean_covariates`
: List of covariates for the mean linear predictor, containing matrices of same dimension as `ts` or returns of the covariate functions of this package (see also `TimeConstant`, `SpatialConstant`).

`dispersion_covariates`
: List of covariates for the dispersion linear predictor, containing matrices of same dimension as `ts` or returns of the covariate functions of this package (see also `TimeConstant`, `SpatialConstant`).

`pseudo_observations`
: (character vector) Defines how pseudo observations for the past dispersion values are calculated. Options are "deviance" (default) and "pearson". See Details.

`wlist_past_mean`
: (Optional) List of matrices, which describes spatial dependencies for the values of the linear predictor. If this is `NULL`, the matrices from `wlist` are used.

`wlist_covariates`
: (Optional) List of matrices, which describes spatial dependencies for the covariates. If this is `NULL`, the matrices from `wlist` are used.

`wlist_pseudo_obs`
: (Optional) List of matrices, which describes spatial dependencies for past values of the pseudo observations. If this is `NULL`, the matrices from `wlist` are used.

`wlist_past_dispersion`
: (Optional) List of matrices, which describes spatial dependencies for the past dispersion values (latent process). If this is `NULL`, the matrices from `wlist` are used.

`wlist_covariates_dispersion`
: (Optional) List of matrices, which describes spatial dependencies for the covariates in the dispersion model. If this is `NULL`, the matrices from `wlist` are used.

`control`
: A list of parameters for controlling the fitting process. This list is passed to `dglmstarma.control`.

Details

For a multivariate time series $\{Y_t = (Y_{1,t}, \ldots, Y_{p,t})'\}$, we assume that the (marginal) conditional components $Y_{i,t} \mid \mathcal{F}_{t-1}$, on the past, follow a distribution that is a member of the exponential dispersion family. The joint multivariate distribution of $Y_t \mid \mathcal{F}_{t-1}$ is assumed to be generated by a process involving copulas. The distributional assumptions imply that the conditional mean $\mu_t := \mathbb{E}(Y_t \mid \mathcal{F}_{t-1})$ and the conditional variance $\mathrm{Var}(Y_{i,t} \mid \mathcal{F}_{t-1}) = \phi_{i,t} V(\mu_{i,t})$, where $V(\cdot)$ is the variance function of the chosen distribution and $\phi_{i,t}$ is the dispersion parameter for location i at time t. The conditional mean is linked to a linear process by the link-function, i.e. $g(\mu_t) = \psi_t$, which is applied elementwise. A second linear process ζ_t is linked to the dispersion parameters of the distributions via a second link-function g_d, i.e. $g_d(\phi_t) = \zeta_t$. The linear predictor for the mean process is defined by regression on past observations, past values of the linear predictor and covariates. It has the following structure:

$$\psi_t = \delta + \sum_{i=1}^{q} \sum_{\ell=0}^{a_i} \alpha_{i,\ell} W_{\alpha}^{(\ell)} h(\psi_{t-i}) + \sum_{j=1}^{r} \sum_{\ell=0}^{b_j} \beta_{j,\ell} W_{\beta}^{(\ell)} \tilde{h}(\mathbf{Y}_{t-j}) + \sum_{k=1}^{m} \sum_{\ell=0}^{c_k} \gamma_{k,\ell} W_{\gamma}^{(\ell)} \mathbf{X}_{k,t},$$

where the matrices $W_{\alpha}^{(\ell)}$, $W_{\beta}^{(\ell)}$, and $W_{\gamma}^{(\ell)}$ are taken from the lists `wlist_past_mean`, `wlist`, and `wlist_covariates`, respectively, and ℓ denotes the spatial order. If $\delta = \delta_0 \mathbf{1}$ with a scalar δ_0, the model is called homogenous with respect to the intercept; otherwise, it is inhomogenous. Spatial orders, intercept structure and time lags for the mean model are specified in the argument `mean_model`. If `past_mean` is specified, it is also required that `past_mean` is specified for identifiability.

The linear process of the dispersion model is defined similarly, but instead of direct observations it includes pseudo observations $\mathbf{d}_t$, which are either defined based on deviance or Pearson residuals. The linear process of the dispersion model has the following structure:

$$\zeta_t = \tilde{\delta} + \sum_{i=1}^{\tilde{q}} \sum_{\ell=0}^{\tilde{a}_i} \tilde{\alpha}_{i,\ell} W_{\alpha,\phi}^{(\ell)} \zeta_{t-i} + \sum_{j=1}^{\tilde{r}} \sum_{\ell=0}^{\tilde{b}_j} \tilde{\beta}_{j,\ell} W_{\beta,\phi}^{(\ell)} \tilde{h}_{\phi}(\mathbf{d}_{t-j}) + \sum_{k=1}^{\tilde{m}} \sum_{\ell=0}^{\tilde{c}_k} \tilde{\gamma}_{k,\ell} W_{\gamma,\phi}^{(\ell)} \tilde{\mathbf{X}}_{k,t},$$

The model orders, neighborhood structures are specified analogously to the mean model via the argument `dispersion_model` and the `wlist_` arguments.

The unknown parameters of the model are estimated with an iterative procedure, which alternates between estimating the mean and dispersion model until convergence. Within each step, a quasi-maximum likelihood approach is used, where for the mean model the quasi-log-likelihood of the observations resulting from the `mean_family` argument is maximized. For the dispersion model, the quasi-likelihood resulting from a Gamma-Density with fixed dispersion parameter of 2 is maximized.

In case of a negative binomial family, the pseudo-observations are always calculated based on Pearson residuals using the Poisson variance function. The dispersion model is defined on these pseudo-observations, which have expectation $1 + \phi_{i,t}\mu_{i,t}$.

Value

The function returns an object of class `dglmstarma`, which includes

- `mean` A list containing information about the mean model, see `glmstarma` for details. Additionally, it contains:
  - `param_history` The sequence of parameters estimates of the mean model during the fitting process.
  - `log_likelihood_history` The sequence of log-likelihood evaluations of the mean model during the fitting process.
- `dispersion` Information about the dispersion model, see `glmstarma` for details. In `ts` it stores the final pseudo-observations. Additionally, it contains:
  - `pseudo_type` Type of pseudo observations used ("deviance" or "pearson").
  - `param_history` The sequence of parameters estimates of the dispersion model during the fitting process.
  - `log_likelihood_history` The sequence of log-likelihood evaluations of the dispersion model during the fitting process.
- `target_dim` Number of parameters in the model.
- `algorithm_info` Information about the fitting algorithm for each iteration of the inner fitting loop.
- `convergence_info` Information about the convergence of the inner fitting loop.

- `total_log_likelihood_history` Evolution of the log-likelihood during the fitting process.
- `total_log_likelihood` The final log-likelihood of the fitted model.
- `aic` AIC of the (full) model based on the log-likelihood, see information_criteria.
- `bic` BIC of the (full) model based on the log-likelihood, see information_criteria.
- `qic` QIC of the (full) model based on the log-likelihood, see QIC.
- `call` The function call.
- `control` The control parameters used for fitting the model.

References

- Jørgensen, B. (1987), Exponential Dispersion Models. Journal of the Royal Statistical Society: Series B (Methodological), 49: 127-145. doi:10.1111/j.25176161.1987.tb01685.x
- Smyth, G.K. (1989), Generalized Linear Models with Varying Dispersion. Journal of the Royal Statistical Society: Series B (Methodological), 51: 47-60. doi:10.1111/j.25176161.1989.tb01747.x

See Also

`stfamily`, `glmstarma.control`, `dglmstarma`, `TimeConstant`, `SpatialConstant`

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
dglmstarma(chickenpox, model_autoregressive, dispersion_model = list(past_obs = 1),
           mean_covariates = list(population = population_hungary),
           wlist = W_hungary, mean_family = vquasipoisson("log"))
```

---

dglmstarma.control *Control Parameters for* dglmstarma *Fitting*

Description

List of control parameters to be passed as an an argument to `dglmstarma`.

Usage

```
dglmstarma.control(
  parameter_init = "zero",
  parameter_init_dispersion = "zero",
  use_sparsity = TRUE,
  sparsity_threshold = 2/3,
```

```
  init_link = "first_obs",
  init_dispersion = "first_obs",
  use_backtracking = TRUE,
  alpha_shrink = 0.5,
  alpha_start = 1,
  min_alpha = 0.05,
  print_progress = TRUE,
  print_warnings = FALSE,
  convergence_threshold = 1e-06,
  max_fits = 50L,
  use_fast_if_const_dispersion = FALSE,
  lower_dispersion = 1e-07,
  upper_dispersion = 1e+06,
  drop_max_mean_lag = TRUE,
  previous_param_as_start = FALSE,
  method = "nloptr",
  constrained_mean = TRUE,
  constrained_dispersion = TRUE,
  constraint_tol = 1e-08,
  constrain_method_mean = "sum_of_absolutes",
  constrain_method_dispersion = "sum_of_absolutes",
  gradtol = sqrt(.Machine$double.eps),
  changetol = sqrt(.Machine$double.eps),
  trace = 0L,
  fnscale = 1,
  maxit = 10000L,
  abstol = -Inf,
  reltol = sqrt(.Machine$double.eps),
  lmm = 5,
  factr = 1e+07,
  pgtol = 0
)
```

Arguments

`parameter_init` Character or list. Start values for parameter estimation. See details.

`parameter_init_dispersion`
: Character or list. Start values for dispersion parameter estimation. See details.

`use_sparsity` Logical; whether to use sparse matrices for the neighborhood matrices.

`sparsity_threshold`
: Numeric in $[0, 1]$. Threshold for proportion of non-zero elements for considering neighborhood matrices as sparse (default: `2/3`).

`init_link` Character or matrix, specifing how to initialize the linear process of the mean model, if regression on the feedback process is included.

- `"first_obs"`: Use the first (transformed) observed values at each location.
- `"mean"`: Use the mean of the (transformed) observed values at each location.

- "transformed_mean": Calculates the mean of the obsverved values at each location and transforms it by the link function.
- "zero": Use zero as initial value.
- (numeric matrix) specifying starting values (rows = location, columns = time, must match maximum temporal order of model)

init_dispersion
: Character or matrix, specifing how to initialize the linear process of the dispersion model, if feedback mechanism is included in the dispersion model.
  - "first_obs": Use the first (transformed) values at each location.
  - "mean": Use the mean of the (transformed) values at each location.
  - "transformed_mean": Calculates the mean of the values at each location and transforms it by the link function.
  - "zero": Use zero as initial value.
  - (numeric matrix) specifying starting values (rows = location, columns = time, must match maximum temporal order of the dispersion model)

use_backtracking
: Logical; whether to use backtracking line search when updating parameters in the fitting procedure. Default is TRUE. See details.

alpha_shrink
: Numeric; shrinkage factor for backtracking line search. Default is 0.5.

alpha_start
: Numeric; initial step size for backtracking line search. Default is 1.0.

min_alpha
: Numeric; minimum step size for backtracking line search. Default is 0.05.

print_progress
: Logical; whether to print progress information during fitting.

print_warnings
: Logical; whether to print warnings if convergence was not achieved (only applicable if print_progress is TRUE).

convergence_threshold
: Numeric. Convergence threshold for fitting procedure. See details.

max_fits
: Integer. Maximum number of iterations between fitting mean and dispersion model. See details.

use_fast_if_const_dispersion
: Logical; whether to use a faster fitting method if the dispersion model is constant, i.e. only an intercept model. See details.

lower_dispersion
: Numeric. Lower bound for pseudo observations. See details.

upper_dispersion
: Numeric. Upper bound for pseudo observations. See details.

drop_max_mean_lag
: Logical; whether to drop the first max_time_lag observations of the mean model when fitting the dispersion model. Default is TRUE (recommended).

previous_param_as_start
: Logical; whether to use the parameter estimates of the previous fitting step as starting values for the next fitting step when iterating between fitting mean and dispersion model. If FALSE, the initial parameter values specified via parameter_init and parameter_init_dispersion are used for each fitting step. Default is FALSE.

`method` Character. Optimization method to be used. Options are:

- `"nloptr"` (requires **nloptr**, default),
- `"optim"` (base R `optim`)

`constrained_mean`
Logical; whether to use parameter constraints ensuring a stable solution. Only works with `method = "nloptr"`.

`constrained_dispersion`
Logical; whether to use parameter constraints ensuring a stable solution for the dispersion model. Only works with `method = "nloptr"`.

`constraint_tol` Numeric. Tolerance for fulfilling constraint.

`constrain_method_mean`
Character. Method for applying parameter constraints.

- `"sum_of_absolutes"`: Sum of absolute values of parameters is constrained
- `"absolute_sum"`: Absolute sum of parameters is constrained. (only intended for univariate models)
- `"soft"`: Constraints for `"softplus"` and `"softclipping"` link functions (not available for different link functions).

`constrain_method_dispersion`
Character. Method for applying parameter constraints for the dispersion model.

- `"sum_of_absolutes"`: Sum of absolute values of parameters is constrained
- `"absolute_sum"`: Absolute sum of parameters is constrained. (only intended for univariate models)

`gradtol` Numeric. Tolerance for gradient convergence. See details.

`changetol` Numeric. Tolerance for parameter change convergence. See details.

`trace` Integer. Level of tracing output. See details.

`fnscale` Numeric. Scaling factor for the objective function. See details.

`maxit` Integer. Maximum number of iterations. See details.

`abstol` Numeric. Absolute convergence tolerance. See details.

`reltol` Numeric. Relative convergence tolerance. See details.

`lmm` Integer. Limited-memory BFGS parameter. See details.

`factr` Numeric. Factor for controlling the convergence tolerance. See details.

`pgtol` Numeric. Tolerance for projected gradient convergence. See details.

Details

This function is called internally in `dglmstarma` to validate control parameters in the `control` argument.

The arguments `constraint_tol`, `gradtol`, `changetol`, `trace`, `fnscale`, `maxit`, `abstol`, `reltol`, `lmm`, `factr`, and `pgtol` are passed to the optimization routines and control the convergence behavior and output. Some of these arguments are not used by all optimization methods.

Iteration between fitting the mean and dispersion model stops when relative change in log-likelihood or absolute change in parameters is below `convergence_threshold` or when `max_fits` is reached.

The optim method uses the L-BFGS-B algorithm when non-negative parameters are required, otherwise the BFGS algorithm is used. Stability constraints cannot be applied when using optim. Only if method = "nloptr" stability constraints are supported, and the specified constrain_method is applied. For optimization we use the SLSQP routine. The constraints implied by constrain_method are given by:

- "sum_of_absolutes":
$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}|\alpha_{i\ell}| + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}|\beta_{j\ell}| \leq 1$$
- "absolute_sum":
$$\left|\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}\alpha_{i\ell} + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}\beta_{j\ell}\right| \leq 1$$
- "soft":
$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}\max\{0,\alpha_{i\ell}\} + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}\max\{0,\beta_{j\ell}\} \leq 1$$
and
$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}|\alpha_{i\ell}| < 1$$

To avoid numerical issues when fitting the dispersion model, the pseudo observations are clamped in between lower_dispersion and upper_dispersion.

If the dispersion model is constant (i.e., only an intercept), setting use_fast_if_const_dispersion = TRUE the dispersion parameters are estimated using means or colMeans of the Pearson or deviance residuals instead of optimizing the dispersion model. Note that this sets the dispersion_link to identity during fitting.

If use_backtracking = TRUE, the fitting procedure aims to increase the total log-likelihood of the model after each fit by applying a backtracking line search.

Start values for the optimization can be provided as a named list via parameter_init or as a character. If a named list is provided, these must match the model orders, see glmstarma.sim. Otherwise, parameter_init must be one of the following:

- "zero": All parameters initialized to (near) zero. If parameters must be non-negative a small value within the feasible region is used.
- "random": All parameters initialized to random values in the stationary region of the model.

Value

A named list of control parameters

See Also

dglmstarma, nloptr, optim

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary
mean_model <- list(past_obs = 1)
dispersion_model <- list(past_obs = 1)
dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
          dispersion_link = "log", W_hungary,
          control = list(parameter_init = "random", print_progress = FALSE))
```

`dglmstarma.sim` *Simulate spatial time-series based on double generalized linear models*

Description

Generates a simulated multivariate time series based on a GLM-like model (see `dglmstarma` for details)

Usage

```
dglmstarma.sim(
  ntime,
  parameters_mean,
  parameters_dispersion,
  model_mean,
  model_dispersion,
  mean_family = NULL,
  dispersion_link = c("log", "identity", "inverse"),
  wlist = NULL,
  mean_covariates = list(),
  dispersion_covariates = list(),
  pseudo_observations = c("deviance", "pearson"),
  wlist_past_mean = NULL,
  wlist_covariates = NULL,
  wlist_pseudo_obs = NULL,
  wlist_past_dispersion = NULL,
  wlist_covariates_dispersion = NULL,
  n_start = 100L,
  control = list()
)
```

Arguments

`ntime`	Number of observation times to be simulated

`parameters_mean`
: a named list specifying the parameters of the model to be simulated:
  - `intercept` (numeric): Intercept parameter. If an inhomogeneous model is simulated, a value must be specified for each component of the time series.
  - `past_obs` (numeric matrix): Parameter values for the past observations.
  - `past_mean` (numeric matrix): Parameter values for the past means.
  - `covariates` (numeric matrix): Parameter values for the covariates.

`parameters_dispersion`
: a named list specifying the parameters of the dispersion model to be simulated, with the same possible elements as in `parameters_mean`.

`model_mean`
: a named list specifying the model for the linear predictor, which can be of the following elements:
  - `intercept` (character): 'homogenous' (default) for a homogenous model, i.e. the same intercept for all components, 'inhomogenous' for inhomogenous models, i.e. an individual intercept for each component.
  - `past_obs` (integer vector/binary matrix): Maximal spatial orders for the time lags in `past_obs_time_lags`. A binary matrix can be passed as an alternative, with the entry in row i and column j indicating whether the $(i-1)$-spatial lag for the j-th time lag is included in the model. If not specified, no regression on past observations is performed.
  - `past_obs_time_lags` (optional integer vector) indicates the time lags for regression on past observations. Defaults are `seq(length(past_obs))` (for vectors) and `seq(ncol(past_obs))` (for a matrix)
  - `past_mean` (integer vector/binary matrix): Spatial orders for the regression on the (latent) feedback process. Values can be entered in the same format as in `past_obs`. If not specified, no regression to the feedback process is performed.
  - `past_mean_time_lags` (optional integer vector) indicates the time lags for regression on past values of the feedback process. Defaults are `seq(length(past_mean))` (for vectors) and `seq(ncol(past_mean))` (for a matrix)
  - `covariates` (integer vector/binary matrix) spatial orders for the covariate processes passed in the argument `covariates`. The values can be passed as in `past_obs` and `past_means`, where the j-th entry or column represents the j-th covariable. If no values are specified but covariates are included, the spatial order 0 is used by default, which corresponds to the first matrix in argument `wlist_covariates`.

`model_dispersion`
: a named list specifying the model for the dispersion linear predictor, with the same possible elements as in `model_mean`. Orders supplied in `past_obs` are applied to the pseudo-observations.

`mean_family`
: An object of class `stfamily` that specifies the marginal distributions of the observations and the link-function for the mean model.

`dispersion_link`
: Link function for the dispersion model. Possible values are `"log"` (default), `"identity"`, and `"inverse"`.

`wlist` A list of quadratic matrices, with the same dimension as the time series, which describe the spatial dependencies. Row-normalized matrices are recommended.

`mean_covariates` List of covariates included in the mean model, containing matrices or returns of the covariate functions of this package (see also `TimeConstant`, `SpatialConstant`).

`dispersion_covariates` List of covariates included in the dispersion model.

`pseudo_observations` Method to generate the pseudo-observations for the dispersion model. Possible values are `"deviance"` (default) and `"pearson"`.

`wlist_past_mean` (Optional) List of matrices, which describes spatial dependencies for the past mean. If this is `NULL`, the matrices from `wlist` are used.

`wlist_covariates` (Optional) List of matrices, which describes spatial dependencies for the covariates. If this is `NULL`, the matrices from `wlist` are used.

`wlist_pseudo_obs` (Optional) List of matrices, which describes spatial dependencies for the pseudo-observations in the dispersion model. If this is `NULL`, the matrices from `wlist` are used.

`wlist_past_dispersion` (Optional) List of matrices, which describes spatial dependencies for the past dispersion in the dispersion model. If this is `NULL`, the matrices from `wlist` are used.

`wlist_covariates_dispersion` (Optional) List of matrices, which describes spatial dependencies for the covariates in the dispersion model. If this is `NULL`, the matrices from `wlist` are used.

`n_start` Number of observations to be used for the burn-in period

`control` A list of parameters for controlling the fitting process. This list is passed to `dglmstarma.control`.

Value

a named list with the following elements:

- `observations` (numeric matrix): The simulated time series
- `link_values` (numeric matrix): The underlying linear predictor resulting from the model and simulation
- `pseudo_observations` (numeric matrix): The pseudo-observations generated for the dispersion model
- `dispersion_values` (numeric matrix): The dispersion values resulting from the dispersion model
- `mean_model` (list): The mean model used for the simulation
- `dispersion_model` (list): The dispersion model used for the simulation
- `parameters_mean` (list): The true parameters used for the mean model
- `parameters_dispersion` (list): The true parameters used for the dispersion model

Examples

```
set.seed(42)
n_obs <- 200L
W <- generateW("rectangle", 100, 2, 10)
model_orders_mean <- list(intercept = "homogeneous",
                          past_obs = 2, past_mean = 1,
                          covariates = c(0, 0))
model_orders_dispersion <- list(intercept = "homogeneous",
                                past_obs = 1,
                                covariates = c(0, 0))

covariates_mean <- list(season = SpatialConstant(sin(2 * pi / 12 * seq(n_obs))),
                  location = TimeConstant(rnorm(100, sd = 0.81)))

covariates_dispersion <- list(season = SpatialConstant(sin(2 * pi / 24 * seq(n_obs))),
                        location = TimeConstant(runif(100)))

params_mean <- list(intercept = 0.6,
                    past_mean = matrix(c(0.2, 0.1), nrow = 2),
                    past_obs = matrix(c(0.2, 0.1, 0.05), nrow = 3),
                    covariates = matrix(c(0.9, 0.2), ncol = 2))
params_dispersion <- list(intercept = 0.5,
                    past_obs = matrix(c(0.5, 0.2), nrow = 2),
                    covariates = matrix(c(0.1, 0.75), ncol = 2))
family <- vnormal(copula = "frank", copula_param = 2)
dglmstarma.sim(n_obs, params_mean, params_dispersion, model_orders_mean,
               model_orders_dispersion, mean_family = family,
               wlist = W, pseudo_observations = "deviance",
               mean_covariates = covariates_mean,
               dispersion_covariates = covariates_dispersion)
```

`fitted.glmstarma` *Fitted values for glmstarma Models*

Description

Compute fitted values for `glmstarma` and `dglmstarma` models.

Usage

```
## S3 method for class 'glmstarma'
fitted(object, drop_init = TRUE)

## S3 method for class 'dglmstarma'
fitted(object, return_value = c("mean", "dispersion"), drop_init = TRUE)
```

Arguments

object	A glmstarma or dglmstarma object.
drop_init	Logical; if TRUE, initial first max_time_lag columns of fitted values are dropped.
return_value	Character; return fitted values of the mean model ("mean") or the dispersion model ("dispersion").

Value

A matrix of fitted values.

See Also

fitted, glmstarma, dglmstarma

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
fitted.values(fit)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
fitted.values(fit2)
fitted.values(fit2, return_value = "dispersion")
```

---

generateW *Generate spatial weight matrices for simulation*

Description

This function generates row-normalized spatial weight matrices for different types of neighborhood structures.

Usage

```
generateW(
  method = c("rectangle", "line", "circle", "full", "independent"),
  dim,
  maxOrder = NULL,
  width = NULL,
  ...
)
```

Arguments

`method`	(character scalar) Defines type of neighborhood structure. Options are "rectangle", "line", "circle", "full", and "independent". Default is "rectangle".
`dim`	(integer scalar) Number of locations, i.e. dimension of the time series.
`maxOrder`	(integer scalar) Maximum spatial order up to which the spatial weight matrices are generated. Ignored if ‘method‘ is "full" or "independent".
`width`	(integer scalar) Width of the rectangular grid. Must be a divisor of `dim`. Ignored if `method` is not "rectangle".
`...`	Additional arguments passed to specific methods.

Details

The function generates spatial weight matrices for different types of neighborhood structures. The options are:

- "rectangle" - A regular rectangular grid (2 dimensional) with `width` columns and `dim / width` rows. The spatial order is defined by the Euclidean distances between locations.
- "line" - Locations are placed on a line (1 dimensional). The spatial order is defined by the Euclidean distances between locations.
- "circle" - Locations are placed on a circle. The spatial order is defined by the Euclidean distances between locations. In contrast to the "line" neighborhood, there are no boundary locations.
- "full" - Generates a list with `dim^2` matrices. Allows simulation/fitting of a full time series model without any restrictions in dependencies between the locations. Not recommended if `dim` is large.
- "independent" - Generates a list with `dim` matrices. Each matrix is a spatial weight matrix with a single 1 in the diagonal. Allows simultaneously simulation/fitting of `dim` univariate time series models without spatial dependencies.

Value

A list of (row normalized) spatial weight matrices.

References

For more advanced spatial weight matrices, consider using the spdep package.

- Bivand R, Pebesma E, Gómez-Rubio V (2013). *Applied spatial data analysis with R*, Second edition. Springer, NY. https://asdar-book.org/.
- Pebesma E, Bivand R (2023). *Spatial Data Science With Applications in R*. Chapman & Hall. https://r-spatial.org/book/.

Examples

```
generateW(method = "rectangle", dim = 100, maxOrder = 2, width = 5)
generateW(method = "full", dim = 4)
```

`glmstarma` *Fit STARMA Models based on generalized linear models*

Description

The function `glmstarma` estimates a multivariate time series model based on generalised linear models (GLM). The primary application is for spatio-temporal data, but different applications, like time-varying network data can be ttacked by this methodology. The model framework links the mean of the time series conditional on the past, to a linear predictor. This linear predictor allows regression on past observations, past values of the linear predictor and covariates, as described in the details. Various distributions with several link-functions are available.

Usage

```
glmstarma(
  ts,
  model = list(),
  wlist,
  family = NULL,
  covariates = NULL,
  wlist_past_mean = NULL,
  wlist_covariates = NULL,
  control = list()
)
```

Arguments

`ts`	Multivariate time-series. Rows indicate the locations and columns the time.
`model`	a named list specifying the model orders of the linear predictor, which can be of the following elements: • `intercept` : (Optional) character – `'homogenous'` (default) for a homogenous model, i.e. the same intercept for all components

  - `'inhomogenous'` for inhomogenous models, i.e. fitting an individual intercept for each component
- `past_obs` :
  - Integer vector with the maximal spatial orders for the time lags in `past_obs_time_lags`.
  - Alternatively: a binary matrix, with the entry in row i and column j indicating whether the $(i-1)$-spatial lag for the j-th time lag is included in the model.
- `past_obs_time_lags` : (Optional) integer vector
  - indicates the time lags for `past_obs`. Defaults to `seq(length(past_obs))` (for vectors) and `seq(ncol(past_obs))` (for a matrix)
- `past_mean` : (Optional)
  - Spatial orders for the regression on past values of (latent) linear process values.
  - Values can be entered in the same format as in `past_obs`. If not specified, no regression to the feedback process is performed.
- `past_mean_time_lags` : (Optional) integer vector
  - Time lags for the regression on the (latent) linear process. Values can be entered in the same format as in `past_obs_time_lags`.
- `covariates` : (Optional)
  - spatial orders for the covariate processes passed in the argument `covariates`. The values can be passed as in `past_obs` and `past_means`, where the j-th entry or column represents the j-th covariate.
  - Default is spatial order 0 for all covariates, which corresponds to the first matrix in argument `wlist_covariates`.

`wlist` A list of quadratic matrices, with the same dimension as the time series has rows, which describe the spatial dependencies. Row-normalized matrices are recommended. See Details.

`family` A list generated by one of the family functions of this package `stfamily`. This argument specifies the marginal distributions and the type of model fitted.

`covariates` (Potentially named) list of covariates, containing matrices of same dimension as `ts` or returns of the covariate functions of this package (see `TimeConstant`, `SpatialConstant`).

`wlist_past_mean`
(Optional) List of matrices, which describes spatial dependencies for the values of the linear predictor. If this is `NULL`, the matrices from `wlist` are used.

`wlist_covariates`
(Optional) List of matrices, which describes spatial dependencies for the covariates. If this is `NULL`, the matrices from `wlist` are used.

`control` A list of parameters for controlling the fitting process. This list is passed to `glmstarma.control`.

Details

For the time series $\{Y_t = (Y_{1,t}, \ldots, Y_{p,t})'\}$, we assume that the (marginal) conditional components $Y_{i,t} \mid \mathcal{F}_{t-1}$ follow a distribution that is a member of the exponential family. The term $\mathcal{F}_{t-1}$ denotes

the history of the process up to time $t-1$. The multivariate distribution of $Y_t \mid \mathcal{F}_{t-1}$ is not necessarily identifiable. The conditional expected value $\mu_t := \mathbb{E}(Y_t \mid \mathcal{F}_{t-1})$ is connected to the linear process by the link-function, i.e. $g(\mu_t) = \psi_t$, which is applied elementwise. The linear process has the following structure:

$$\psi_t = \delta + \sum_{i=1}^{q} \sum_{\ell=0}^{a_i} \alpha_{i,\ell} W_{\alpha}^{(\ell)} h(\psi_{t-i}) + \sum_{j=1}^{r} \sum_{\ell=0}^{b_j} \beta_{j,\ell} W_{\beta}^{(\ell)} \tilde{h}(\mathbf{Y}_{t-j}) + \sum_{k=1}^{m} \sum_{\ell=0}^{c_k} \gamma_{k,\ell} W_{\gamma}^{(\ell)} \mathbf{X}_{k,t},$$

where the matrices $W_{\alpha}^{(\ell)}$, $W_{\beta}^{(\ell)}$, and $W_{\gamma}^{(\ell)}$ are taken from the lists `wlist_past_mean`, `wlist`, and `wlist_covariates`, respectively, and ℓ denotes the spatial order. If $\delta = \delta_0 \mathbf{1}$ with a scalar δ_0, the model is called homogenous with respect to the intercept; otherwise, it is inhomogenous. Spatial orders, intercept structure and deviating time lags are specified in the argument `model`. If `past_mean` is specified, it is also required that `past_mean` is specified for identifiability.

The functions h and $\tilde{h}$ are set internally with the `family` argument. In nearly all cases, h corresponds to the identity function, i.e. $h(\psi_{t-i}) = \psi_{t-i}$, and $\tilde{h}$ is similar to the link-function. Using count data as an example, for `family = vpoisson("identity")` and `family = vpoisson("log")` result in the linear and log-linear Poisson STARMA models from Maletz et al. (2024), where `vpoisson("softplus")` results in the approximately linear model by Jahn et al. (2023).

The unknown parameters δ, $\alpha_{i,\ell}$, $\beta_{j,\ell}$, and $\gamma_{k,\ell}$ are estimated by quasi-maximum likelihood estimation, assuming conditional independence of the components given the past for calculating the quasi-likelihood. Parameter estimation is by default performed under stability constraints to ensure stability of the model. These constraints can be modified or deactivated via the `control` argument. See `glmstarma.control` for details.

Value

The function returns an object of class `glmstarma`, which contains beside the (maybe revised) input to the function:

- `target_dim` Number of locations. Corresponds to the number of rows in `ts`.
- `n_obs_effective` Effective number of observation times. Corresponds to the number of columns in `ts` minus the maximum time lag of the model.
- `max_time_lag` Maximum time lag in the model.
- `log_likelihood` The (quasi)-log-likelihood of the fitted model, which is based on `n_obs_effective` observation times.
- `score` The (quasi)-score vector at the quasi maximum likelihood estimation.
- `information` The (quasi)-information matrix at the quasi maximum likelihood estimation.
- `variance_estimation` The variance estimation of the parameter estimates. Calculated based on a sandwich estimator.
- `aic` AIC of the model based on the quasi log-likelihood, see information_criteria.
- `bic` BIC of the model based on the quasi log-likelihood, see information_criteria.
- `qic` QIC of the model based on the quasi log-likelihood, see QIC.
- `design_matrix` The final design matrix of the model
- `derivatives` The derivatives of the linear predictor with respect to the parameters at each time point.

- `fitted.values` The fitted values of the model, which can be extracted by the `fitted` method.
- `link_values` Fitted values of the linear process, i.e. ψ_t.
- `algorithm` Information about the fitting method.
- `convergence` A named list with information about the convergence of the optimization:
  - `start` The values used for the coefficients at the start of the estimation.
  - `fncount` Number of calls of the quasi-loglikelihood during optimization.
  - `grcount` Number of calls of the quasi-score during optimization.
  - `hecount` Number of calls of the quasi-information during optimization. In algorithms not using the information, this is 0 or the number how often constrains are evaluated.
  - `fitting_time` The time in milliseconds it took to estimate the model.
  - `convergence` Logical value indicating the convergence of the algorithm.
  - `message` An optional message by the optimization algorithm.
- `call` The function call.

References

- Cliff, A. D., & Ord, J. K. (1975). Space-Time Modelling with an Application to Regional Forecasting. Transactions of the Institute of British Geographers, 64, 119–128. doi:10.2307/621469
- Jahn, M., Weiß, C.H., & Kim, H., (2023), Approximately linear INGARCH models for spatio-temporal counts, *Journal of the Royal Statistical Society Series C: Applied Statistics*, 72(2), 476–497, doi:10.1093/jrsssc/qlad018
- Maletz, S., Fokianos, K., & Fried, R. (2024). Spatio-Temporal Count Autoregression. *Data Science in Science*, 3(1), doi:10.1080/26941899.2024.2425171
- Pfeifer, P. E., & Deutsch, S. J. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling Phillip. Technometrics, 22(1), 35–47. doi:10.2307/1268381

See Also

`stfamily`, `glmstarma.control`, `dglmstarma`, `TimeConstant`, `SpatialConstant`

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(intercept = "homogeneous", past_obs = rep(1, 7))
glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
          covariates = list(population = population_hungary),
          control = list(parameter_init = "zero"))
```

`glmstarma.control` *Control Parameters for* `glmstarma` *Fitting*

Description

List of control parameters to be passed as an an argument to `glmstarma`.

Usage

```
glmstarma.control(
  parameter_init = "zero",
  init_link = "first_obs",
  dispersion_est_type = "deviance",
  use_sparsity = TRUE,
  sparsity_threshold = 2/3,
  method = "nloptr",
  constrained = TRUE,
  constraint_tol = 1e-08,
  constrain_method = "sum_of_absolutes",
  gradtol = sqrt(.Machine$double.eps),
  changetol = sqrt(.Machine$double.eps),
  trace = 0L,
  fnscale = 1,
  maxit = 10000L,
  abstol = -Inf,
  reltol = sqrt(.Machine$double.eps),
  lmm = 5,
  factr = 1e+07,
  pgtol = 0
)
```

Arguments

`parameter_init`	Character or list. Start values for parameter estimation. See details.
`init_link`	Character or matrix, specifing how to initialize the linear process of the mean model, if regression on the feedback process is included. • `"first_obs"`: Use the first (transformed) observed values at each location. • `"mean"`: Use the mean of the (transformed) observed values at each location. • `"transformed_mean"`: Calculates the mean of the obsverved values at each location and transforms it by the link function. • `"zero"`: Use zero as initial value. • (numeric matrix) specifying starting values (rows = location, columns = time, must match maximum temporal order of model)

`dispersion_est_type`
: Character. Estimation of global dispersion parameter either based on deviance (`"deviance"`) or pearson residuals (`"pearson"`), if applicable.

`use_sparsity`
: Logical; whether to use sparse matrices for the neighborhood matrices.

`sparsity_threshold`
: Numeric in $[0, 1]$. Threshold for proportion of non-zero elements for considering neighborhood matrices as sparse (default: `2/3`).

`method`
: Character. Optimization method to be used. Options are:
  - `"nloptr"` (requires **nloptr**, default),
  - `"optim"` (base R `optim`)

`constrained`
: Logical; whether to use parameter constraints ensuring a stationary solution. Only works with `method = "nloptr"`.

`constraint_tol`
: Numeric. Tolerance for constraint satisfaction.

`constrain_method`
: Character. Method for applying parameter constraints.
  - `"sum_of_absolutes"`: Sum of absolute values of parameters is constrained
  - `"absolute_sum"`: Absolute sum of parameters is constrained. (only intended for univariate models)
  - `"soft"`: Constraints for `"softplus"` and `"softclipping"` link functions (not available for different link functions).

`gradtol`
: Numeric. Tolerance for gradient convergence. See details.

`changetol`
: Numeric. Tolerance for parameter change convergence. See details.

`trace`
: Integer. Level of tracing output. See details.

`fnscale`
: Numeric. Scaling factor for the objective function. See details.

`maxit`
: Integer. Maximum number of iterations. See details.

`abstol`
: Numeric. Absolute convergence tolerance. See details.

`reltol`
: Numeric. Relative convergence tolerance. See details.

`lmm`
: Integer. Limited-memory BFGS parameter. See details.

`factr`
: Numeric. Factor for controlling the convergence tolerance. See details.

`pgtol`
: Numeric. Tolerance for projected gradient convergence. See details.

Details

This function is called internally in `glmstarma` to validate control parameters in the `control` argument.

The arguments `constraint_tol`, `gradtol`, `changetol`, `trace`, `fnscale`, `maxit`, `abstol`, `reltol`, `lmm`, `factr`, and `pgtol` are passed to the optimization routines and control the convergence behavior and output. Some of these arguments are not used by all optimization methods.

The `optim` method uses the L-BFGS-B algorithm when non-negative parameters are required, otherwise the BFGS algorithm is used. Stationarity constraints cannot be applied when using `optim`. Only if `method = "nloptr"` stationarity constraints are supported, and the specified `constrain_method` is applied. For optimization we use the SLSQP routine. The constraints implied by `constrain_method` are given by:

- `"sum_of_absolutes"`:

$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}|\alpha_{i\ell}| + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}|\beta_{j\ell}| \leq 1$$

- `"absolute_sum"`:

$$\left|\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}\alpha_{i\ell} + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}\beta_{j\ell}\right| \leq 1$$

- `"soft"`:

$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}\max\{0,\alpha_{i\ell}\} + \sum_{j=1}^{r}\sum_{\ell=0}^{b_j}\max\{0,\beta_{j\ell}\} \leq 1$$

  and

$$\sum_{i=1}^{q}\sum_{\ell=0}^{a_i}|\alpha_{i\ell}| < 1$$

Start values for the optimization can be provided as a named list via `parameter_init` or as a character. If a named list is provided, these must match the model orders, see `glmstarma.sim`. Otherwise, `parameter_init` must be one of the following:

- `"zero"`: All parameters initialized to (near) zero. If parameters must be non-negative a small value within the feasible region is used.
- `"random"`: All parameters initialized to random values in the stationary region of the model.

In case of a negative binomial family, the global dispersion parameter is always estimated using dispersion_est_type = "pearson". It corresponds to the shape parameter of the negative binomial distribution.

Value

A named list of control parameters

See Also

`glmstarma`, `nloptr`, `optim`

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
          control = list(parameter_init = "random", init_link = "mean"))
```

`glmstarma.sim` *Simulate spatial time-series based on generalized linear models*

Description

Generates a simulated multivariate time series based on a GLM-like model (see `glmstarma` for details)

Usage

```
glmstarma.sim(
  ntime,
  parameters,
  model,
  family = NULL,
  wlist,
  covariates = list(),
  wlist_past_mean = NULL,
  wlist_covariates = NULL,
  n_start = 100L,
  control = list()
)
```

Arguments

`ntime`	Number of observation times to be simulated
`parameters`	a named list specifying the parameters of the model to be simulated, which has the following elements: • `intercept` (numeric): Intercept parameter. If an inhomogeneous model is simulated, a value must be specified for each component of the time series. • `past_obs` (numeric matrix): Parameter values for the past observations. • `past_mean` (numeric matrix): Parameter values for the past means. • `covariates` (numeric matrix): Parameter values for the covariates.
`model`	a named list specifying the model for the linear predictor, which can be of the following elements: • `intercept` (character): 'homogenous' (default) for a homogenous model, i.e. the same intercept for all components, 'inhomogenous' for inhomogenous models, i.e. an individual intercept for each component. • `past_obs` (integer vector/binary matrix): Maximal spatial orders for the time lags in `past_obs_time_lags`. A binary matrix can be passed as an alternative, with the entry in row i and column j indicating whether the $(i-1)$-spatial lag for the j-th time lag is included in the model. If not specified, no regression on past observations is performed. • `past_obs_time_lags` (optional integer vector) indicates the time lags for regression on past observations. Defaults are `seq(length(past_obs))` (for vectors) and `seq(ncol(past_obs))` (for a matrix)

- past_mean (integer vector/binary matrix): Spatial orders for the regression on the (latent) feedback process. Values can be entered in the same format as in past_obs. If not specified, no regression to the feedback process is performed.
- past_mean_time_lags (optional integer vector) indicates the time lags for regression on past values of the feedback process. Defaults are seq(length(past_mean)) (for vectors) and seq(ncol(past_mean)) (for a matrix)
- covariates (integer vector/binary matrix) spatial orders for the covariate processes passed in the argument covariates. The values can be passed as in past_obs and past_means, where the jth entry or column represents the jth covariable. If no values are specified but covariates are included, the spatial order 0 is used by default, which corresponds to the first matrix in argument wlist_covariates.

family — An object of class stfamily that specifies the marginal distributions and the type of model fitted.

wlist — A list of quadratic matrices, with the same dimension as the time series, which describe the spatial dependencies. Row-normalized matrices are recommended.

covariates — List of covariates, containing matrices or returns of the covariate functions of this package (see also TimeConstant, SpatialConstant). The matrices must have the same dimension as ts

wlist_past_mean — (Optional) List of matrices, which describes spatial dependencies for the past mean. If this is NULL, the matrices from wlist are used.

wlist_covariates — (Optional) List of matrices, which describes spatial dependencies for the covariates. If this is NULL, the matrices from wlist are used.

n_start — Number of observations to be used for the burn-in period

control — A list of parameters for controlling the fitting process. This list is passed to glmstarma.control.

Value

a named list with the following elements:

- observations (numeric matrix): The simulated time series
- link_values (numeric matrix): The underlying linear predictor resulting from the model and simulation
- model (list): The model used for the simulation
- parameters (list): The true parameters used for the simulation

Examples

```
set.seed(42)
n_obs <- 200L
W <- generateW("rectangle", 100, 2, 10)
model_orders <- list(intercept = "homogeneous", past_obs = 2, past_mean = 1)
parameter <- list(intercept = 0.5, past_obs = matrix(c(0.3, 0.2, 0.05), nrow = 3),
```

```
                  past_mean = matrix(c(0.1, 0.05), nrow = 2),
                  covariates = c(0.75, 0.5))
covariates <- list(season = SpatialConstant(sin(2* pi / 12 * seq(n_obs))),
                   location = TimeConstant(rnorm(100, sd = 0.81)))
# Simulation using marginal poisson distribution
glmstarma.sim(n_obs, parameter, model_orders, W, covariates, family = vpoisson("log"))
# Simulation using negative binomial marginals
glmstarma.sim(n_obs, parameter, model_orders, W, covariates,
              family = vnegative.binomial(dispersion = 3))
```

`glmstarma_sim.control` *Control Parameters for Simulation of* `glmstarma` *Models*

Description

List of control parameters to be passed to the `glmstarma.sim` function.

Usage

```
glmstarma_sim.control(
  return_burn_in = FALSE,
  init_link = "parameter",
  use_sparsity = TRUE,
  sparsity_threshold = 2/3
)
```

Arguments

`return_burn_in`	Logical; if `TRUE`, include the burn-in period in the returned simulated data. Default is `FALSE`.
`init_link`	Character or matrix. Method to initialize first link values in the burn-in period. See details.
`use_sparsity`	Logical; whether to use sparse matrices for the neighborhood matrices.
`sparsity_threshold`	Numeric in $[0, 1]$. Threshold for proportion of non-zero elements for considering neighborhood matrices as sparse (default: `2/3`).

Details

This function validates control arguments for `glmstarma.sim`. By default, the initial link values for the burn-in period are generated by calculating the unconditional mean of the process based on the model parameters, ignoring covariates. Different initial link values can submitted as a numeric matrix, with p rows (number of locations) and `max_time_lag` columns (maximum time lag of the model).

Value

A named list of control parameters

See Also

glmstarma.sim, glmstarma.control

---

information_criteria *Information Criteria for glmstarma and dglmstarma objects*

Description

Compute AIC, BIC, and QIC and (Quasi-)log-likelihood for glmstarma and dglmstarma objects.

Usage

```
## S3 method for class 'glmstarma'
AIC(object, k = 2, adjust = TRUE)

## S3 method for class 'dglmstarma'
AIC(object, k = 2, adjust = TRUE)

## S3 method for class 'glmstarma'
BIC(object, adjust = TRUE)

## S3 method for class 'dglmstarma'
BIC(object, adjust = TRUE)

## S3 method for class 'glmstarma'
logLik(object, adjust = TRUE)

## S3 method for class 'dglmstarma'
logLik(object, adjust = TRUE)
```

Arguments

object	An object of class glmstarma or dglmstarma
k	Numeric; penalty per parameter to be used. Default is 2 (standard AIC).
adjust	Logical; if TRUE (default), the (quasi-)log-likelihood is adjusted for the effective sample size. See Details.

Details

During model fitting, the (quasi-)log-likelihood is computed only on the last n_eff time-points, where n_eff = n - max_time_lag_mean - max_time_lag_dispersion. Here n is the total number of time-points, max_time_lag_mean the maximum temporal lag in the mean model, and max_time_lag_dispersion the maximum temporal lag in the dispersion model (for dglmstarma objects). If no dispersion model is present (class glmstarma), max_time_lag_dispersion is zero.

To be more specific the (quasi-)log-likelihood calculated during model estimation is given by

$$\ell(\theta) = \sum_{t=\tau}^{n} \sum_{i=1}^{p} \ell_{i,t}(\theta),$$

where $\ell_{i,t}(\theta)$ denotes the (quasi-)log-likelihood of the observation at location i at time t, and $\tau = n - n_{\text{eff}}$.

This calculation of the (quasi-)log-likelihood introduces bias when comparing models of different temporal orders. If `adjust = TRUE`, the (quasi-)log-likelihood is rescaled to `n` observations by multiplying with n/n_{eff}, before calculating the AIC, BIC, QIC or (quasi)-log-likelihood.

Value

A numeric value for the (possibly adjusted) AIC, BIC, QIC or (quasi-)log-likelihood.

See Also

AIC, BIC, logLik, QIC

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
AIC(fit)
BIC(fit)
logLik(fit)
QIC(fit)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
AIC(fit2)
BIC(fit2)
logLik(fit2)
QIC(fit2)
```

load_data *Load example datasets*

Description

Download and return datasets from the glmSTARMA GitHub repository

Usage

```
load_data(
  name = NULL,
  refresh = FALSE,
  directory = tools::R_user_dir("glmSTARMA", which = "data")
)
```

Arguments

name	Name of the dataset to load. One of "rota", "chickenpox", or "sst".
refresh	Logical; re-download the dataset if it already exists locally.
directory	Directory where the dataset should be cached. Defaults to a user-specific data directory of the glmSTARMA package.

Details

This function downloads example datasets from the glmSTARMA GitHub repository and caches them in a directory specified by the user. The default directory is a user-specific data directory of the glmSTARMA package. If the dataset has already been downloaded to the specified directory, it is loaded from the local cache unless refresh = TRUE is specified.

Value

A named list of objects

See Also

delete_glmSTARMA_data, rota, chickenpox, sst

Examples

```
# Load the 'chickenpox' dataset
chickenpox_data <- load_data("chickenpox", directory = tempdir())
str(chickenpox_data)
```

QIC *Quasi Information Criterion (QIC) for glmstarma and dglmstarma objects*

Description

Generic function to compute the QIC (Pan, 2001), a model selection criterion commonly used for Generalized Estimating Equations (GEE) and related models.

Usage

```
QIC(object, ...)

## S3 method for class 'glmstarma'
QIC(object, adjust = TRUE, ...)

## S3 method for class 'dglmstarma'
QIC(object, adjust = TRUE, ...)
```

Arguments

object	Object of class glmstarma or dglmstarma.
...	Additional arguments passed to specific methods.
adjust	Logical; if TRUE (default), an adjustment for the temporal orders of the model is applied to the likelihood. See Details.

Details

The quasi information criterion (QIC) has been proposed by Pan (2001) as alternative to Akaike's information criterion (AIC) which is properly adjusted for regression analysis based on the generalized estimating equations (GEE). It is defined as

$$QIC = -2 \cdot \ell + 2 \cdot \left(\text{trace}(G_\mu^{-1} H_\mu) + \text{trace}(G_\phi^{-1} H_\phi) \right),$$

where ℓ is the (quasi-)log-likelihood of the estimated model, G_μ is the expected information matrix of the regression parameters of the mean model, and H_μ the empirical covariance matrix of the regression parameters of the mean model. Similarly, G_ϕ and H_ϕ denote the corresponding matrices for the dispersion model (only for dglmstarma objects). For glmstarma objects, the second term reduces to $\text{trace}(G_\mu^{-1} H_\mu)$.

For more details on the calculation of G and H, see sandwich_variance

During model estimation, the (quasi-)log-likelihood is computed only on the last n_eff time-points, where n_eff = n - max_time_lag_mean - max_time_lag_dispersion. Here n is the total number of time-points, max_time_lag_mean the maximum temporal lag in the mean model, and max_time_lag_dispersion the maximum temporal lag in the dispersion model (for dglmstarma objects). If no dispersion model is present (class glmstarma), max_time_lag_dispersion is zero.

To be more specific the (quasi-)log-likelihood calculated during model estimation is given by

$$\ell(\theta) = \sum_{t=\tau}^{n} \sum_{i=1}^{p} \ell_{i,t}(\theta),$$

where $\ell_{i,t}(\theta)$ denotes the (quasi-)log-likelihood of the observation at location i at time t, and $\tau = n - n_{\text{eff}}$.

This calculation of the (quasi-)log-likelihood introduces bias when comparing models of different temporal orders. If `adjust = TRUE`, the (quasi-)log-likelihood is rescaled to `n` observations by multiplying with n/n_{eff}, before calculating the QIC.

Value

A numeric value for the QIC.

References

Pan, W. (2001). Akaike's Information Criterion in Generalized Estimating Equations. *Biometrics*, 57(1), 120–125. doi:10.1111/j.0006341X.2001.00120.x

See Also

`AIC`, `BIC`, `logLik`, `information_criteria`

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
QIC(fit)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
QIC(fit2)
```

`residuals.glmstarma` *Residuals for glmstarma and dglmstarma Models*

Description

Compute residuals for fitted `glmstarma` and `dglmstarma` models.

Usage

```
## S3 method for class 'glmstarma'
residuals(
  object,
  type = c("response", "pearson", "deviance"),
  drop_init = TRUE,
  ignore_dispersion = TRUE
)

## S3 method for class 'dglmstarma'
residuals(
  object,
  type = c("response", "pearson", "deviance"),
  drop_init = TRUE,
  ignore_dispersion = TRUE
)
```

Arguments

`object`	A fitted `glmstarma` or `dglmstarma` object.
`type`	Type of residuals to compute. Options are `"response"` (raw residuals), `"pearson"`. See details.
`drop_init`	Logical; if `TRUE`, initial first `max_time_lag` columns of residuals are dropped.
`ignore_dispersion`	Logical; if `TRUE`, values are not scaled by the dispersion parameter

Details

The `type` argument specifies the type of residuals to compute:

- `"response"`: Raw residuals, computed as the difference between observed and fitted values.
- `"pearson"`: Pearson residuals, defined as

$$r_i = \frac{y_i - \mu_i}{\sqrt{V(\mu_i)}}$$

, where $V(\mu_i)$ is the variance function of the specified family.

- `"deviance"`: Deviance residuals, defined as

$$r_i = 2 \cdot (\ell(y_i; y_i) - \ell(y_i; \mu_i)),$$

i.e. the log-likelihood difference of a saturated model and the fitted model. If `ignore_dispersion` is set to `FALSE`, pearson and deviance residuals are scaled by the dispersion parameter(s).

Value

A matrix of residuals.

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
residuals(fit)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
residuals(fit2)
```

---

rota *Rota Virus Infections in Germany*

Description

Multivariate count time series with rota virus infections in the counties of Germany.

Format

rota A matrix with counts of rota virus infections (rows = counties, columns = time points).

gdr_feature A numeric vector indicating whether a county was part of the former German Democratic Republic (1) or not (0). Counties in the Capital Berlin have the value 0.5.

population_germany A numeric matrix containing the population in 10000 inhabitants of each district over time.

W_germany A list of matrices containing spatial weight matrices:

1. Identity matrix.
2. Row-normalized adjacency matrix of the districts. See details.
3. Row-normalized adjacency matrix of order 2. See details.

Details

This dataset contains weekly rota virus counts in 411 counties of Germany over a time period of 1252 weeks (from 2001 to 2024). Estimates of the population size of each county are only available on a yearly basis and have been linearly interpolated to obtain weekly estimates.

Throughout the observation period, there were several territorial reforms in which certain regions were merged or reorganized. The data source (RKI) for infection counts takes into account the regional divisions that were in existence at the time the data was queried from the website. The data on population size contains the data for the divisions valid at the time of data collection. For the purpose of standardisation, the data was aggregated in order to reflect the territorial reforms as accurately as possible and to align it with the infection figures. For Berlin, annual population estimates are only available for Berlin as a whole. We use data from the 2022 census to divide the population proportionally among the 12 districts.

The row-normalized adjacency matrix of first order indicates which districts share a common border. The weight matrix of order 2 indicates which districts can be reached in two steps (i.e., via a common neighbor).

The dataset is not included directly in the package. Use `load_data("rota")` to download it.

Source

The data originate from the Robert Koch Institute (RKI) and the Federal Statistical Office of Germany (Destatis).

Infection data: Robert Koch Institute: SurvStat@RKI 2.0, https://survstat.rki.de, retrieved on 2025-02-03.

Population data: Federal Statistical Office of Germany (Destatis), Data licence Germany – attribution – version 2.0, https://www-genesis.destatis.de/datenbank/online/statistic/12411/table/12411-0015/table-toolbar, retrieved on 2025-12-08.

Census data (Berlin): Federal Statistical Office of Germany (Destatis), Data licence Germany – attribution – version 2.0, https://ergebnisse.zensus2022.de/datenbank/online/statistic/1000A/table/1000A-0000, retrieved on 2025-12-08.

Examples

```
# Note: Complete examples take around 2,5 minutes to run #
dat <- load_data("rota", directory = tempdir())
rota <- dat$rota
gdr_feature <- dat$gdr_feature
population_germany <- dat$population_germany
W_germany <- dat$W_germany

covariates <- list(population = population_germany,
                   gdr = TimeConstant(gdr_feature),
                   season_cos = SpatialConstant(cos(2 * pi / 52 * 1:1252)),
                   season_sin = SpatialConstant(sin(2 * pi / 52 * 1:1252)),
                   vaccine_west = (gdr_feature == 0) %*% t(seq(ncol(rota)) >= 654),
                   vaccine_east = (gdr_feature > 0) %*% t(seq(ncol(rota)) >= 654))
fit <- glmstarma(rota, list(past_obs = rep(2, 4)), wlist = W_germany,
                 covariates = covariates, family = vpoisson("log"))
```

```
mean_model <- list(past_obs = rep(2, 4))
dispersion_model <- list(past_obs = 2)
fit2 <- dglmstarma(rota, mean_model, dispersion_model, mean_covariates = covariates,
                   dispersion_covariates = covariates,
                   mean_family = vquasipoisson("log"),
                    dispersion_link = "log", W_germany)
```

`SpatialConstant` *Creates a spatial constant covariate*

Description

This function assigns a `const` attribute set to `"space"` to a numeric vector.

Usage

```
SpatialConstant(x)
```

Arguments

x	(numeric vector) covariate values for each time-point. Values are used for each location.

Details

A spatial-constant covariate has the form

$$\mathbf{x}_t = x_t \mathbf{1}_p,$$

i.e., it has the same value x_t for all locations at time-point t.

Value

The input numeric vector with an additional attribute `const` set to `"space"`.

See Also

TimeConstant

`sst` *Sea Surface Temperature Anomalies in the Pacific*

Description

Multivariate time series containing monthly sea surface temperature anomalies in the Pacific

Format

SST A matrix with monthly sea surface temperature anomalies (rows = spatial locations, columns = time points).

W_directed A list of (sparse) matrices containing spatial weight matrices. See details

1. Identity matrix.
2. `north`: weight matrix for northward direction
3. `east`: weight matrix for eastward direction
4. `south`: weight matrix for southward direction
5. `west`: weight matrix for westward direction

locations data.frame containing the latitude and longitude of each spatial location.

Details

This dataset contains monthly sea surface temperature anomalies (in degree Celsius) at 25 spatial locations in the Pacific Ocean over a time period of 33 years (396 months) from January 1970 to December 2002.

The neighborhood matrices in the list `W_directed` for orders 2, 3, 4, and 5 correspond to neighbors with the location directly north, east, south, and west. Coefficients estimated using these matrices then reflect directed dependencies. Not all neighbors always exist at boundary locations of the observed area. In these cases, the corresponding rows of the weight matrices contain only zeros. These matrices are stored as objects of class 'dgCMatrix' from the 'Matrix' R package.

The dataset is not included directly in the package. Use `load_data("sst")` to download it.

Source

The data is a transformed subset of the `SST_df`-Dataset from the (archived) `STRbook` R package. It is still available on GitHub (https://github.com/andrewzm/STRbook)

References

- Wikle, C.K., Zammit-Mangion, A., and Cressie, N. (2019). *Spatio-Temporal Statistics with R*. Chapman & Hall/CRC, Boca Raton, FL.
- Cressie, N, and Wikle, C.K. (2011). *Statistics for Spatio-Temporal Data*. John Wiley & Sons, Incorporated.

Examples

```
# Note: Complete examples take around 4 minutes to run #
# Requires the 'Matrix' package
if(requireNamespace("Matrix")){
 dat <- load_data("sst", directory = tempdir())
 SST <- dat$SST
 W_directed <- dat$W_directed
 locations <- dat$locations

 times <- seq(from = as.Date("1970-01-01"), to = as.Date("2002-12-01"), by = "m")
 times <- format(times, "%b %Y")
 covariates <- list(trend = SpatialConstant(seq(times) / length(times)),
                    longitude = TimeConstant(locations$lon / 360),
                    season_cos = SpatialConstant(cos(2 * pi / 12 * seq(times))),
                    season_sin = SpatialConstant(sin(2 * pi / 12 * seq(times))),
                    abs_lat_inc = TimeConstant(pmin(abs(locations$lat), 6) / 90),
                    abs_lat_dec = TimeConstant(pmax(abs(locations$lat) - 6, 0) / 90))

 fit <- glmstarma(SST, model = list(past_obs = 4, past_mean = 4),
                  wlist = W_directed, wlist_past_mean = W_directed,
                  covariates = covariates, family = vnormal())

 fit2 <- dglmstarma(SST, mean_model = list(past_obs = 4, past_mean = 4),
                    dispersion_model = list(past_obs = 4),
                    wlist = W_directed, mean_covariates = covariates,
                    dispersion_covariates = covariates, mean_family = vnormal())
}
```

`stfamily` *Families for spatio-temporal GLMs*

Description

These functions create family objects for various distributions used in spatio-temporal generalized linear models (STGLMs). Each function returns an object of class `"stfamily"` describing a (conditional) marginal distribution, a link function, and optional dispersion values. The output is intended only for use within this package and is processed internally in the functions of this package.

Usage

```
vpoisson(
  link = c("log", "identity", "sqrt", "softplus"),
  const = 1,
  copula = NULL,
  copula_param = NULL,
  sampling_method = c("inversion", "poisson_process")
)
```

```
vquasipoisson(
  link = c("log", "identity", "sqrt", "softplus"),
  dispersion = NULL,
  const = 1,
  copula = NULL,
  copula_param = NULL,
  sampling_method = c("build_up", "chop_down", "branching", "negbin")
)

vnegative.binomial(
  link = c("log", "identity", "sqrt", "softplus"),
  dispersion = NULL,
  const = 1,
  copula = NULL,
  copula_param = NULL
)

vbinomial(
  link = c("softclipping", "identity", "logit", "probit"),
  size = 1,
  const = 1,
  copula = NULL,
  copula_param = NULL
)

vquasibinomial(
  link = c("softclipping", "identity", "logit", "probit"),
  size = 1,
  dispersion = NULL,
  const = 1,
  copula = NULL,
  copula_param = NULL
)

vgamma(
  link = c("inverse", "log", "identity"),
  dispersion = NULL,
  copula = NULL,
  copula_param = NULL
)

vinverse.gaussian(
  link = c("1/mu^2", "inverse", "identity", "log"),
  dispersion = NULL,
  copula = NULL,
  copula_param = NULL
)
```

```
vnormal(
  link = c("identity", "log", "inverse"),
  dispersion = NULL,
  copula = NULL,
  copula_param = NULL
)
```

Arguments

link	Character string specifying the link function. Options depend on the distribution (see Details).
const	Optional numeric constant used in some link functions.
copula	Optional copula family to model dependence between responses. Has no effect on parameter estimation, but only on situations in which data is generated with this family.
copula_param	Parameter for the copula. (Numeric scalar of length 1)
sampling_method	Sampling algorithm for Poisson and quasi-Poisson.
dispersion	Optional dispersion parameter(s). Can be either a numerical scalar describing a global, time-invariant dispersion parameter, a vector with (temporal) constant dispersion parameters for each location, or a matrix with dispersion parameters for each location and time. (Rows are locations, columns are time points) If NULL, dispersion will be estimated as a scalar where applicable.
size	Number of trials for binomial-type families.

Details

The link argument specifies the link function to be used for the family. The available link functions depend on the distribution:

- Poisson, Quasi-Poisson, Negative-Binomial: "log", "identity", "sqrt", "softplus"
- Binomial, Quasi-Binomial: "softclipping", "identity", "logit", "probit"
- Gamma: "inverse", "log", "identity"
- Inverse Gaussian: "1/mu^2", "inverse", "identity", "log"
- Gaussian/Normal: "identity", "log", "inverse"

The following families are available:

- vpoisson() – Poisson distribution
- vquasipoisson() – Quasi-Poisson, i.e. Poisson like, but dispersion can differ from 1
- vnegative.binomial() – Negative binomial distribution
- vbinomial() – Binomial distribution
- vquasibinomial() – Quasi-Binomial, i.e. Binomial like, but dispersion can differ from 1
- vgamma() – Gamma distribution
- vinverse.gaussian() – Inverse Gaussian distribution

- `vnormal()` – Gaussian distribution
- `vgarch()` – GARCH distribution

The following copulas are available:

- "normal" – Gaussian copula
- "t" – t copula
- "clayton" – Clayton copula
- "frank" – Frank copula
- "gumbel" – Gumbel copula
- "joe" – Joe copula

The data generating processes of each distribution rely on (sequences of) uniform marginals, which are transformed to obtain the observed data. The copula specifies the dependence structure between these uniform marginals. If no copula (`copula = NULL`) is specified, the uniform marginals are generated independent.

For most distributions, the data generation is based on inversion of the cumulative distribution function (CDF). For the Poisson distribution, two different sampling methods are implemented: `sampling_method = "inversion"` results in the inversion method using `qpois` and `sampling_method = "poisson_process"` implements the Poisson process method described in Fokianos et al. (2020). For a quasi-Poisson model, four different sampling methods are implemented: `sampling_method = "build_up"`, `sampling_method = "chop_down"`, `sampling_method = "branching"` and `sampling_method = "negbin"`. The first three methods generate data from a generalized Poisson distribution, see Consul and Jain (1973) and are described in Demirtas (2017). The `sampling_method = "negbin"` uses the Inversion method on properly parameterized negative binomial distribution to generate the data. If the "branching" or "negbin" method is used, only overdispersion can be generated. Dispersion values resulting in underdispersion will be set to 1, i.e. a standard Poisson case. For Quasi-Binomial models, in case of overdispersion, the data is generated from a sequence of positive correlated Bernoulli trials, see Ahn and Chen (1995) for a discussion. In case of underdispersion, data is generated using a normal approximation. In case of the inverse Gaussian distribution, data is generated using the Michael-Schucany-Haas method, see Michael et al. (1976).

Note that for the negative binomial family, the dispersion parameter corresponds to the shape parameter of the negative binomial distribution.

Value

An object of class `"stfamily"` containing elements such as `link`, `distribution`, `variance`, `dev.resids`.

References

- Ahn, H., & Chen, J. J. (1995). Generation of Over-Dispersed and Under-Dispersed Binomial Variates. Journal of Computational and Graphical Statistics, 4(1), 55–64. doi:10.1080/10618600.1995.10474665
- Consul, P. C., & Jain, G. C. (1973). A Generalization of the Poisson Distribution. Technometrics, 15(4), 791–799. doi:10.1080/00401706.1973.10489112

- Demirtas, H. (2017). On accurate and precise generation of generalized Poisson variates. Communications in Statistics - Simulation and Computation, 46(1), 489–499. doi:10.1080/03610918.2014.968725
- Fokianos, K., Støve, B., Tjøstheim, D., & Doukhan, P. (2020). Multivariate count autoregression. Bernoulli, 26(1), 471–499. doi:10.3150/19BEJ1132
- Michael, J. R., Schucany, W. R., & Haas, R. W. (1976). Generating Random Variates Using Transformations with Multiple Roots. The American Statistician, 30(2), 88–90. doi:10.1080/00031305.1976.10479147

See Also

family

Examples

```
fam <- vpoisson(link = "log")
print(fam)

fam2 <- vbinomial(link = "logit", size = 10)
print(fam2)
```

---

summary.dglmstarma *Summarize a dglmstarma Model*

Description

This functions summarizes the model fit of a dglmstarma model.

Usage

```
## S3 method for class 'dglmstarma'
summary(object, phi = 1, alternative = c("two.sided", "less", "greater"), ...)
```

Arguments

object	An object of class dglmstarma
phi	Numeric value indicating the null hypothesis value for the dispersion parameter test. Default is 1.
alternative	Character string specifying the alternative hypothesis for the dispersion parameter test. Must be one of "two.sided" (default), "less" or "greater".
...	Additional arguments passed to specific methods.

Details

Standard errors, z-values and p-values are computed assuming asymptotic normality of the parameter estimation. The variance estimation is based on the sandwich estimator to adjust for quasi-maximum-likelihood estimation. If the model requires non-negative parameters, the p-values are adjusted accordingly. Note that this adjustment is only valid for testing single parameters against the null hypothesis of being zero. If multiple parameters are tested simultaneously, or a linear combination of them, a different adjustment is necessary.

If the dispersion model is constant, i.e. it is only an intercept model, a test is performed to test whether the estimated dispersion parameter is significantly different from the null hypothesis value `phi`. The alternative hypothesis can be specified via the `alternative` argument. This can be useful to test for overdispersion or underdispersion in the data.

Value

An object of class `summary.dglmstarma` which contains the following elements

- call: The function call to fit the model
- coefficients_mean: The estimated coefficients of the mean model with approximate standard errors, z- and p-values. See details.
- coefficients_dispersion: The estimated coefficients of the dispersion model with approximate standard errors, z- and p-values. See details.
- distribution: The marginal distribution of the conditional observations.
- link: The link-function used to connect the conditional mean with the linear process of the mean model.
- dispersion_link: The link-function used to connect the dispersion with the linear process of the dispersion model.
- dispersion_disp_parameter: The dispersion parameter of the dispersion family
- df_mean: Number of estimated coefficients in the mean model
- df_dispersion: Number of estimated coefficients in the dispersion model
- log_likelihood: The quasi-log-likelihood of the estimated model. See details.
- aic: Akaike Information Criterion of the estimated model, see information_criteria with `adjust = TRUE`.
- bic: Bayesian Information Criterion of the estimated model, see information_criteria with `adjust = TRUE`.
- qic: Quasi Information Criterion of the estimated model, see QIC with `adjust = TRUE`.

See Also

dglmstarma, summary.glmstarma

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
```

```
W_hungary <- dat$W_hungary
mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                  dispersion_link = "log", wlist = W_hungary,
                  mean_covariates = list(population = population_hungary))
summary(fit)
```

`summary.glmstarma` *Summarize the results of a glmstarma model*

Description

This functions summarizes the model fit of a glmstarma model

Usage

```
## S3 method for class 'glmstarma'
summary(object, ...)
```

Arguments

`object`	An object of class `glmstarma`
`...`	Additional arguments passed to specific methods.

Details

Standard errors, z-values and p-values are computed assuming asymptotic normality of the parameter estimation. The variance estimation is based on the sandwich estimator to adjust for quasi-maximum-likelihood estimation.

If the model requires non-negative parameters, the p-values are adjusted accordingly. Note that this adjustment is only valid for testing single parameters against the null hypothesis of being zero. If multiple parameters are tested simultaneously, or a linear combination of them, a different adjustment is necessary.

Value

An object of class `summary.glmstarma` which contains the following elements

- call: The function call to fit the model
- coefficients: The estimated coefficients of the model with approximate standard errors, z- and p-values. See details.
- distribution: The marginal distribution of the conditional observations.
- link: The link-function used to connect the conditional mean with the linear process.
- dispersion: The dispersion parameter of the conditional distribution

- estimate_dispersion: logical value indicating whether `dispersion` was estimated (`TRUE`) or fixed by the distribution or user (`FALSE`)
- df: Number of estimated coefficients in the model
- log_likelihood: The quasi-log-likelihood of the estimated model. See details.
- aic: Akaike Information Criterion of the estimated model, see information_criteria with `adjust = TRUE`.
- bic: Bayesian Information Criterion of the estimated model, see information_criteria with `adjust = TRUE`.
- qic: Quasi Information Criterion of the estimated model, see QIC with `adjust = TRUE`.

See Also

glmstarma, logLik, AIC, BIC, QIC, logLik.glmstarma, AIC.glmstarma, BIC.glmstarma

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
summary(fit)
```

TimeConstant *Creates a time constant covariate*

Description

This functions assigns a `const` attribute set to `"time"` to a numeric vector.

Usage

```
TimeConstant(x)
```

Arguments

x	(numeric vector) covariate values for each location. Values are used for each time-point.

Details

A time-constant covariate has the form

$$\mathbf{x}_t = \mathbf{x},$$

for some fixed vector $\mathbf{x} = (x_1, \ldots, x_p)'$, i.e., it has the same value for all time-points t, where x_i is the value of the covariate at location i.

Value

The input vector with an additional attribute `const` set to `"time"`.

See Also

SpatialConstant

`vcov.dglmstarma` *Variance-Covariance Matrix for glmstarma and dglmstarma objects*

Description

Computes the variance-covariance matrix for `glmstarma` and `dglmstarma` objects.

Usage

```
## S3 method for class 'dglmstarma'
vcov(object, return_value = c("mean", "dispersion", "both"))

## S3 method for class 'glmstarma'
vcov(object)
```

Arguments

`object`	An object of class `glmstarma` or `dglmstarma` for which the variance-covariance matrix is to be computed.
`return_value`	A character string specifying which variance-covariance matrix to return. Options are "mean", "dispersion", or "both". Default is "mean".

Details

The variance-covariance matrix is computed using a sandwich estimator approach, which accounts for potential misspecification of the model. The sandwich variance estimation is defined as $V = G^{-1}HG^{-1}$, where G is the expected information matrix and H is the empirical covariance matrix of the score functions. In case of `dglmstarma` objects, separated variance-covariance matrices are computed for the mean and dispersion models because of the alternating estimation procedure.

Value

For `glmstarma` objects, the function returns the variance-covariance matrix of the mean model coefficients. For `dglmstarma` objects, the function return depends on the `return_value` argument:

`mean`	Variance-covariance matrix for the mean model coefficients.
`dispersion`	Variance-covariance matrix for the dispersion model coefficients.
`both`	A list containing both variance-covariance matrices.

See Also

vcov, glmstarma, dglmstarma

Examples

```
dat <- load_data("chickenpox", directory = tempdir())
chickenpox <- dat$chickenpox
population_hungary <- dat$population_hungary
W_hungary <- dat$W_hungary

model_autoregressive <- list(past_obs = rep(1, 7))
fit <- glmstarma(chickenpox, model_autoregressive, W_hungary, family = vpoisson("log"),
                 covariates = list(population = population_hungary))
vcov(fit)

mean_model <- list(past_obs = rep(1, 7))
dispersion_model <- list(past_obs = 1)
fit2 <- dglmstarma(chickenpox, mean_model, dispersion_model, mean_family = vquasipoisson("log"),
                   dispersion_link = "log",
                   wlist = W_hungary,
                   mean_covariates = list(population = population_hungary))
vcov(fit2)
vcov(fit2, return_value = "dispersion")
vcov(fit2, return_value = "both")
```

Index