Package ‘RSTr’

January 17, 2026

Type Package

Title Gibbs Samplers for Discrete Bayesian Spatiotemporal Models

Version 1.0.0

Date 2026-01-08

Maintainer David DeLara <sfq1@cdc.gov>

URL https://cehi-code-repos.github.io/RSTr/

Description Takes Poisson or Binomial discrete spatial data and runs a Gibbs sampler for a variety of Spatiotemporal Conditional Autoregressive (CAR) models. Includes measures to prevent estimate over-smoothing through a restriction of model informativeness for select models. Also provides tools to load output and get median estimates. Implements methods from Besag, York, and Mollié (1991) ``Bayesian image restoration, with two applications in spatial statistics'' <doi:10.1007/BF00116466>, Gelfand and Vounatsou (2003) ``Proper multivariate conditional autoregressive models for spatial data analysis'' <doi:10.1093/biostatistics/4.1.11>, Quick et al. (2017) ``Multivariate spatiotemporal modeling of age-specific stroke mortality'' <doi:10.1214/17-AOAS1068>, and Quick et al. (2021) ``Evaluating the informativeness of the Besag-York-Mollié CAR model'' <doi:10.1016/j.sste.2021.100420>.

License GPL (>= 3)

Encoding UTF-8

LinkingTo Rcpp (>= 1.1.0), RcppArmadillo (>= 15.2.2.1), RcppDist (>= 0.1.1.1)

Suggests ggplot2, knitr, rmarkdown, sf, testthat (>= 3.0.0)

Imports abind, Rcpp (>= 1.1.0), RcppArmadillo (>= 15.2.2.1), RcppDist (>= 0.1.1.1), spdep

Depends R (>= 4.3.0)

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr, rmarkdown

Config/testthat/edition 3

NeedsCompilation yes

Author David DeLara [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0485-7549>),
Centers for Disease Control and Prevention [aut, cph]
(https://ror.org/042twtr12)

Repository CRAN

Date/Publication 2026-01-17 20:10:16 UTC

Contents

`RSTr-package` *Gibbs Samplers for Discrete Bayesian Spatiotemporal Models*

Description

Takes Poisson or Binomial discrete spatial data and runs a Gibbs sampler for a variety of Spatiotemporal Conditional Autoregressive (CAR) models. Includes measures to prevent estimate oversmoothing through a restriction of model informativeness for select models. Also provides tools to load output and get median estimates. Implements methods from Besag, York, and Mollié (1991) "Bayesian image restoration, with two applications in spatial statistics" <doi:10.1007/BF00116466>, Gelfand and Vounatsou (2003) "Proper multivariate conditional autoregressive models for spatial data analysis" <doi:10.1093/biostatistics/4.1.11>, Quick et al. (2017) "Multivariate spatiotemporal modeling of age-specific stroke mortality" <doi:10.1214/17-AOAS1068>, and Quick et al. (2021) "Evaluating the informativeness of the Besag-York-Mollié CAR model" <doi:10.1016/j.sste.2021.100420>.

Details

The RSTr package uses Bayesian spatiotemporal modeling to spatially smooths discrete small-area event rates using information from neighboring spatial regions. See ‘browseVignettes("RSTr")‘ for a series of tutorials on basic usage of the RSTr functions.

Author(s)

David DeLara [aut, cre] (ORCID: <https://orcid.org/0000-0003-0485-7549>), Centers for Disease Control and Prevention [aut, cph] (https://ror.org/042twtr12)

Maintainer: David DeLara <sfq1@cdc.gov>

References

Besag, J., York, J., and Mollié, A. (1991). Bayesian Image Restoration with Two Applications in Spatial Statistics (with Discussion). Annals of the Institute of Statistical Mathematics, 43, 1–59. doi:10.1007/BF00116466

Gelfand, A. E., & Vounatsou, P. (2003). Proper multivariate conditional autoregressive models for spatial data analysis. Biostatistics, 4(1), 11–25. doi:10.1093/biostatistics/4.1.11

Quick, et al. (2017). Multivariate spatiotemporal modeling of age-specific stroke mortality. Annals of Applied Statistics, 11(4), 2165–2177. doi:10.1214/17AOAS1068

Quick, et al. (2021). Evaluating the informativeness of the Besag-York-Mollié CAR model. Spatial and Spatio-temporal Epidemiology, 37, 100420. doi:10.1016/j.sste.2021.100420

`add_neighbors` *Add neighbors to adjacency information*

Description

Modifies `adjacency` to indicate that `neighs` they should be treated as neighbors.

Usage

```
add_neighbors(adjacency, neighs)
```

Arguments

`adjacency`	Adjacency information generated by `spdep::poly2nb()`.
`neighs`	A vector of regions to mark as adjacent. Accepts a vector of indices or names assigned to `adjacency`.

Details

`add_neighbors()` is useful when adjacency information generated by `spdep::poly2nb()` indicates lone regions without links/neighbors, particularly in island counties such as the Hawaiian islands, Nantucket in Massachusetts, or San Juan in Washington. Note that `add_neighbors()` marks all listed counties as adjacent, so if you have a set of chaining counties where the first may not be connected to the last, several instances of `add_neighbors()` will be needed.

Value

A modified `adjacency` `list`.

Examples

```
mamap <- sf::st_as_sf(mamap[order(mamap$GEOID), ])
ma_adj <- spdep::poly2nb(mamap)
new_neighs <- c(1, 4, 10) # attach regions 1, 4, and 10
ma_adj <- add_neighbors(ma_adj, new_neighs)

# Add neighbors by FIPS code instead of index
ma_adj <- suppressWarnings(spdep::poly2nb(mamap))
names(ma_adj) <- mamap$GEOID
ma_adj <- add_neighbors(ma_adj, neighs = c("25001", "25007", "25019"))

ma_adj <- suppressWarnings(spdep::poly2nb(mamap))
ma_adj <- add_neighbors(ma_adj, c(1, 4)) # only attach 1 and 4
ma_adj <- add_neighbors(ma_adj, c(4, 10)) # only attach 4 and 10
```

`age_standardize` *Age-standardize model objects*

Description

Age-standardizes samples using a standard population for an `RSTr` model object.

Usage

```
age_standardize(RSTr_obj, std_pop, new_name, groups = NULL)
```

Arguments

`RSTr_obj`	An `RSTr` model object.
`std_pop`	A vector of standard populations.
`new_name`	The name to assign to the age-standardized group.
`groups`	A vector of either indices for each group or a vector of strings for each group name. If set to `NULL`, will use all groups in the dataset.

Value

An `RSTr` object with age-standardized estimates.

Examples

```
std_pop <- c(113154, 100640, 95799)
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
# age-standardize by all age groups
mod_mst <- age_standardize(mod_mst, std_pop, "35-64")
# Add onto age-standardized estimates. Age-standardize only by the first two age groups
mod_mst <- age_standardize(mod_mst, std_pop[1:2], "35-54", groups = 1:2)
```

---

aggregate_count *Aggregate count arrays*

Description

Sums counts over event/population arrays. Useful when manually generating group-aggregated/age-standardized estimates and a population threshold is needed for suppression.

Usage

```
aggregate_count(
  count,
  margin,
  groups = NULL,
  bind_new = FALSE,
  new_name = NULL
)
```

Arguments

count	The `array` to aggregate.
margin	For `arrays`, The margin on which the groups of interest are stratified.
groups	A vector of either indices for each group or a vector of strings for each group name. If set to `NULL`, will use all groups in the dataset.
bind_new	If set to `TRUE`, will bind an `array` to the original sample dataset. Otherwise, will generate a standalone array of samples.
new_name	The name to assign to the age-standardized group.

Value

An `array` of aggregated count data.

Examples

```
margin_time <- 3
# aggregate population from all years for each county-group
pop_7988 <- aggregate_count(miheart$n, margin_time)
# aggregate population from 1980-1984 for each county-group
pop_8084 <- aggregate_count(miheart$n, margin_time, groups = as.character(1980:1984))
# bind aggregated pop from all years to population data
pop_agg <- aggregate_count(miheart$n, margin_time, bind_new = TRUE, new_name = "1979-1988")
```

---

aggregate_samples *Aggregate samples by non-age group*

Description

Consolidates a set of samples over non-age groups using a population array to create weighted-average samples.

Usage

```
aggregate_samples(
  sample,
  pop,
  margin,
  groups = NULL,
  bind_new = FALSE,
  new_name = NULL
)
```

Arguments

sample	an `array` of samples imported with `load_samples()`
pop	The population array to be used for weighted averages.
margin	For `array`s, The margin on which the groups of interest are stratified.
groups	A vector of either indices for each group or a vector of strings for each group name. If set to `NULL`, will use all groups in the dataset.
bind_new	If set to `TRUE`, will bind an `array` to the original sample dataset. Otherwise, will generate a standalone array of samples.
new_name	The name to assign to the age-standardized group.

Details

`aggregate_samples()` is only meant for non-age group data, such as spatial regions, time periods, or other sociodemographic groups (race, sex, etc.). If you are interested in consolidating samples by age group, use `age_standardize()` instead. Additionally, if you plan on doing age-standardization along with aggregating by other groups, always aggregate groups first before doing age-standardization to ensure that the samples are properly standardized.

Value

An `array` of weighted-average samples.

Examples

```
pop <- miheart$n[1:2, 1:3, 1:3]
time_margin <- 3
# calculate prevalence by aggregating over time periods
samples_3564 <- aggregate_samples(minsample, pop, margin = time_margin)
# calculate prevalence of only the first two time periods
samples_3554 <- aggregate_samples(minsample, pop, time_margin, groups = 1:2)
# bind prevalence samples to original samples
samples_prev <- aggregate_samples(
  minsample,
  pop,
  time_margin,
  bind_new = TRUE,
  new_name = "1979-1981"
)
```

`car` *Create CAR model*

Description

`*car()` generates an `RSTr` model object, samples, and estimates for either an MSTCAR, MCAR, RCAR, or CAR model.

Usage

```
car(
  name,
  data,
  adjacency,
  dir = tempdir(),
  seed = NULL,
  perc_ci = 0.95,
  iterations = 6000,
  show_plots = TRUE,
  verbose = TRUE,
  ignore_checks = FALSE,
  method = c("binomial", "poisson"),
  impute_bounds = NULL,
  inits = NULL,
  priors = NULL
)

rcar(
  name,
  data,
  adjacency,
  dir = tempdir(),
  seed = NULL,
  perc_ci = 0.95,
  A = NULL,
  m0 = NULL,
  iterations = 6000,
  show_plots = TRUE,
  verbose = TRUE,
  ignore_checks = FALSE,
  method = c("binomial", "poisson"),
  impute_bounds = NULL,
  inits = NULL,
  priors = NULL
)

mcar(
  name,
  data,
  adjacency,
  dir = tempdir(),
  seed = NULL,
  perc_ci = 0.95,
  iterations = 6000,
  show_plots = TRUE,
  verbose = TRUE,
  ignore_checks = FALSE,
  method = c("binomial", "poisson"),
  impute_bounds = NULL,
  inits = NULL,
  priors = NULL
)

mstcar(
  name,
  data,
  adjacency,
  dir = tempdir(),
  seed = NULL,
  perc_ci = 0.95,
  iterations = 6000,
  show_plots = TRUE,
  verbose = TRUE,
  ignore_checks = FALSE,
  method = c("binomial", "poisson"),
  impute_bounds = NULL,
  inits = NULL,
  priors = NULL,
  update_rho = FALSE
)
```

Arguments

`name`	Name of model and corresponding folder.
`data`	Dataset including mortality (Y) and population (n) information.
`adjacency`	Dataset including adjacency information.
`dir`	Directory where model will live.
`seed`	Set of random seeds to use for data replication.
`perc_ci`	The percentage of the desired estimate credible interval. Defaults to 95 percent (0.95).
`iterations`	The number of iterations to run the model for.
`show_plots`	If set to `FALSE`, suppresses traceplots.
`verbose`	If set to `FALSE`, suppresses model progress messages.
`ignore_checks`	If set to `TRUE`, skips model validation.
`method`	Run model with either Binomial data or Poisson data.
`impute_bounds`	If counts are suppressed for privacy reasons, `impute_bounds` is the lower/upper bound of suppression, typically 0 or 1 and 10, respectively.
`inits`	Optional list of initial conditions for each parameter.
`priors`	Optional list of priors for updates.
`A`	For RCAR models, describes maximum intensity of smoothing between regions.
`m0`	For RCAR models, baseline neighbor count by region.
`update_rho`	For MSTCAR models, controls whether rho update is performed.

Value

An `RSTr` model object.

Examples

```
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
# MSTCAR model
mod_mst <- mstcar(
  name = "test",
  data = data_min,
  adjacency = adj_min,
  dir = tempdir(),
  show_plots = FALSE,
  verbose = FALSE
)
```

get_estimates *Extract estimates from RSTr model object*

Description

Gathers model and estimate information for an RSTr model object, exported as a long table. Estimate rates and their respective credible intervals are displayed by default in rates per 100,000.

Usage

```
get_estimates(RSTr_obj, rates_per = 1e+05, standardized = TRUE)
```

Arguments

RSTr_obj	An RSTr model object.
rates_per	The desired scaling for estimate rates.
standardized	If RSTr_obj contains age-standardized rates, shows the age-standardized rates. If set to FALSE, always shows the non-age-standardized rates.

Value

A long table containing region/group/time period names, estimates, credible intervals, relative precisions, and the associated event/population counts.

Examples

```
std_pop <- c(113154, 100640, 95799)
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
estimates_table <- get_estimates(mod_mst)
mod_mst <- age_standardize(mod_mst, std_pop, "35-64")
estimates_table_as <- get_estimates(mod_mst)
```

get_medians *Generate medians, credible intervals, and relative precisions*

Description

get_medians() generates median estimates for array of samples loaded from load_samples().

get_credible_interval() generates the credible interval of each estimate using samples loaded from load_samples().

get_relative_precision() generates the relative precision of each estimate using samples loaded from load_samples(). The relative precision for an estimate is defined as the ratio of the estimate's median divided by the width of its credible interval.

Usage

```
get_medians(sample)

get_credible_interval(sample, perc_ci = 0.95)

get_relative_precision(medians, ci)
```

Arguments

`sample`	array of samples generated by `load_samples`.
`perc_ci`	Number from 0 to 1. Determines width of credible interval.
`medians`	Array of medians generated from samples.
`ci`	Credible interval generated by `get_credible_interval()`.

Value

An `array` of estimates/credible intervals/relative precisions.

Examples

```
minmedians <- get_medians(minsample)
minci <- get_credible_interval(minsample)
# Reducing perc_ci narrows the credible interval
minci_75 <- get_credible_interval(minsample, perc_ci = 0.75)
# low relative precision due to small data size
minrp <- get_relative_precision(minmedians, minci)
# reducing CI increases relative precision
minrp_75 <- get_relative_precision(minmedians, minci_75)
# find estimates with low relative precision
low_rp <- minrp_75 < 1
```

`load_model` *Load model*

Description

`load_model()` imports an `RSTr` object with name `name` in directory `dir`.

Usage

```
load_model(name, dir = tempdir())
```

Arguments

`name`	The name of the model to load.
`dir`	The directory in which the model lives.

Value

An `RSTr` model object.

Examples

```
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
mod_mst <- load_model(name = "test", dir = tempdir())
```

`load_samples` *Load MCMC samples*

Description

`load_samples()` gathers samples saved for model `RSTr_obj`. By default, loads the rate estimate samples `lambda`, but any model parameters can be loaded. Users can also specify a burn-in period.

Usage

```
load_samples(RSTr_obj, param = "lambda", burn = 2000)
```

Arguments

`RSTr_obj`	`RSTr` model object to load in samples from.
`param`	Which parameter samples to load.
`burn`	Number of burn-in samples to discard.

Value

An `array` of samples from model `RSTr_obj`.

Examples

```
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
samples <- load_samples(mod_mst) * 1e5
```

`long_to_list_matrix` *Generate count data for RSTr object*

Description

`long_to_list_matrix()` converts a long `table` featuring event counts across regions and other optional margins into a `list` that is readable by `*car()`.

Usage

```
long_to_list_matrix(
  table,
  event,
  population,
  region,
  group = NULL,
  time = NULL
)
```

Arguments

`table`	A `table` containing event and mortality counts stratified by group/region/time.
`event`	The column containing event counts.
`population`	The column containing population counts.
`region`	The column containing region names.
`group`	An optional column containing sociodemographic group names.
`time`	An optional column containing time period names.

Details

`long_to_list_matrix()` will sum along any group/time stratifications that aren't specified; for example, if your dataset contains time periods and time is not specified in `long_to_list_matrix()`, the output will be a sum of all time periods. Filter data by desired groups and time periods before running `long_to_list_matrix()`.

Value

A `list` of mortality and population counts organized into multi-dimensional arrays.

Examples

```
ma_data <- maexample[!is.na(maexample$Year), ]
# Generates data from 1979-1981 stratified by sex
ma_data_mst <- long_to_list_matrix(ma_data, Deaths, Population, County.Code, Sex.Code, Year.Code)
ma_data_79 <- ma_data[ma_data$Year == 1979, ]
# Generates 1979 data stratified by sex
```

```
ma_data_m <- long_to_list_matrix(ma_data_79, Deaths, Population, County.Code, Sex.Code)
# Generates 1979 data summarized for all sexes
ma_data_u <- long_to_list_matrix(ma_data_79, Deaths, Population, County.Code)
```

`maexample` *Massachusetts Heart Attack Mortality Data*

Description

An example dataset from CDC WONDER containing counts for Myocardial Infarction (ICD-9 code 410.0) deaths in all 14 Massachusetts counties for individuals in 2 sex groups across 3 years.

Usage

```
maexample
```

Format

'maexample' a data frame with 117 rows and 10 variables:

Notes Dataset notes, starts at line 85

Year Year label

Year.Code Year label

County County name

County.Code County FIPS code

Sex Sex group

Sex Abbreviated sex group

Deaths Death count

Population Population count

Crude.Rate Crude rate generated by death and population count

Source

<https://wonder.cdc.gov/cmf-icd9.html>

`mamap` *Massachusetts Shapefile*

Description

A dataset containing U.S. Census TIGER shape data for Massachusetts

Usage

```
mamap
```

Format

'mamap' a data frame with 14 rows and 13 variables:

STATEFP State FIPS code

COUNTYFP County FIPS code

COUNTYNS County GNIS code

AFFGEOID Census Unique Identifier

GEOID Census Unique Identifier, truncated

NAME County Name

NAMELSAD County LSAD code

STUSPS Abbreviated State Name

STATE_NAME State Name

LSAD State LSAD code

ALAND amount of land in square meters

AWATER amount of water in square meters

geometry shape data for each county

Source

<https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html>

`miadj` *Michigan Adjacency Data*

Description

A dataset containing the adjacency structure for each county in Michigan

Usage

```
miadj
```

Format

‘miadj‘ a list containing neighbor adjacency vectors.

`miheart` *Michigan Heart Attack Mortality Data*

Description

A dataset containing counts for Myocardial Infarction (ICD-9 code 410.0) deaths in all 83 Michigan counties for individuals in 6 age groups across 10 years. This dataset also contains the corresponding population counts.

Usage

```
miheart
```

Format

‘miheart‘ a list with two array objects:

Y death count (0-1005)

n population count (26-292828)

Source

<https://wonder.cdc.gov/cmf-icd9.html>

`minsample` *Samples Generated for Michigan data*

Description

A 4-dimensional `array` of samples generated by the MSTCAR Gibbs sampler for use in testing and examples

Usage

```
minsample
```

Format

'minsample' a small sample dataset to be used in examples, generated by an MSTCAR model with two regions, three age groups, and three time periods.

`mishp` *Michigan Shapefile*

Description

A dataset containing U.S. Census TIGER shape data for Michigan

Usage

```
mishp
```

Format

'mishp' a data frame with 83 rows and 13 variables:

STATEFP State FIPS code
COUNTYFP County FIPS code
COUNTYNS County GNIS code
AFFGEOID Census Unique Identifier
GEOID Census Unique Identifier, truncated
NAME County Name
NAMELSAD County LSAD code
STUSPS Abbreviated State Name
STATE_NAME State Name
LSAD State LSAD code
ALAND amount of land in square meters
AWATER amount of water in square meters
geometry shape data for each county

Source

<https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html>

`standardize_samples` *Age-standardize samples*

Description

Age-standardizes samples using a standard population.

Usage

```
standardize_samples(
  sample,
  std_pop,
  margin,
  groups = NULL,
  bind_new = FALSE,
  new_name = NULL
)
```

Arguments

`sample`	an `array` of samples imported with `load_samples()`
`std_pop`	A vector of standard populations.
`margin`	For `arrays`, The margin on which the groups of interest are stratified.
`groups`	A vector of either indices for each group or a vector of strings for each group name. If set to `NULL`, will use all groups in the dataset.
`bind_new`	If set to `TRUE`, will bind an `array` to the original sample dataset. Otherwise, will generate a standalone array of samples.
`new_name`	The name to assign to the age-standardized group.

Value

An `array` of age-standardized samples.

Examples

```
std_pop <- c(113154, 100640, 95799)
age_margin <- 2
# age-standardize by all age groups
samples_3564 <- standardize_samples(minsample, std_pop, age_margin)
# age-standardize only by the first two age groups
samples_3554 <- standardize_samples(minsample, std_pop[1:2], age_margin, groups = 1:2)
# bind age-standardized samples to original samples
samples_as <- standardize_samples(
  minsample,
  std_pop,
  age_margin,
  bind_new = TRUE,
  new_name = "35-64"
)
```

suppress_estimates *Suppress estimates based on reliability criteria*

Description

Generates suppressed estimates for an `RSTr` model object with a given relative precision and population/event threshold.

Usage

```
suppress_estimates(RSTr_obj, threshold = 0, type = c("population", "event"))
```

Arguments

`RSTr_obj`	An `RSTr` model object.
`threshold`	The population/event suppression threshold.
`type`	Determines whether suppression threshold is based on population counts or event counts.

Details

While the `threshold` argument is optional, population/event thresholds are necessary for non-restricted models. Population/event thresholds should only be omitted for restricted CAR models, such as the RCAR.

Value

An `RSTr` model object with suppressed estimates.

Examples

```
std_pop <- c(113154, 100640, 95799)
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
mod_mst <- suppress_estimates(mod_mst, threshold = 1000, type = "population")
estimates_table <- get_estimates(mod_mst)
```

update_model *Update model*

Description

`update_model()` generates additional samples for model `RSTr_obj`.

Usage

```
update_model(RSTr_obj, iterations = 6000, show_plots = TRUE, verbose = TRUE)
```

Arguments

`RSTr_obj`	The `RSTr` model object to generate samples for.
`iterations`	Number of iterations to run.
`show_plots`	If set to `FALSE`, hides traceplots.
`verbose`	If set to `FALSE`, hides progress bar and other messages.

Value

An `RSTr` model object.

Examples

```
data_min <- lapply(miheart, \(x) x[1:2, 1:3, 1:3])
adj_min <- list(2, 1)
mod_mst <- mstcar("test", data_min, adj_min, tempdir(), show_plots = FALSE, verbose = FALSE)
mod_mst <- update_model(mod_mst, iterations = 1000, show_plots = FALSE, verbose = FALSE)
```

Index