Package ‘trafficCAR’

January 27, 2026

Type Package

Title Bayesian CAR Models for Road-Segment Traffic

Version 0.1.0

Author Madison Ell [aut, cre]

Maintainer Madison Ell <mell@stat.tamu.edu>

Description Tools for simulating and modeling traffic flow on road networks using spatial conditional autoregressive (CAR) models. The package represents road systems as graphs derived from 'OpenStreetMap' data <https://www.openstreetmap.org/> and supports network-based spatial dependence, basic preprocessing, and visualization for spatial traffic analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Imports sf, igraph, Matrix, methods, units, stats, graphics, rlang, posterior, ggplot2

Suggests knitr, rmarkdown, microbenchmark, osmdata, lwgeom, leaflet, viridisLite, htmltools, usethis, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.3

Depends R (>= 3.5)

NeedsCompilation no

Repository CRAN

Date/Publication 2026-01-27 20:50:02 UTC

Contents

`augment_roads` *Augment roads with predicted traffic quantities*

Description

Adds predicted traffic outcomes (e.g. speed or volume) and relative congestion measures to a road network.

Usage

```
augment_roads(fit, roads, probs = c(0.025, 0.975), keep_geometry = TRUE)
```

Arguments

`fit`	a ‘traffic_fit‘ object from ‘fit_traffic()‘.
`roads`	an sf object or data.frame with a segment id column.
`probs`	length-2 numeric for equal-tail intervals.
`keep_geometry`	logical; if FALSE drops sf geometry.

Value

roads with added columns: predicted_mean, predicted_lo, predicted_hi, relative_congestion

build_adjacency *Build segment adjacency from segment geometries*

Description

Constructs an undirected adjacency matrix ‘A‘ where segments are neighbors if they share a node (endpoint). Intended to be used after ‘roads_to_segments()‘.

Usage

```
build_adjacency(segments, crs_m = 3857, tol = 0, verbose = FALSE)
```

Arguments

`segments`	An ‘sf‘ with LINESTRING geometries and (optionally) ‘seg_id‘.
`crs_m`	Metric CRS used when ‘segments‘ is lon/lat (for robust node keys). Default 3857.
`tol`	Nonnegative numeric tolerance for snapping node coordinates (in meters if projected). If 0, uses exact coordinates. Default 0.
`verbose`	Logical; emit simple messages. Default FALSE.

Details

Isolates (degree 0) are kept (all-zero rows/cols). Connected components are returned for ICAR sum-to-zero centering per component.

Value

A list with:

A Sparse symmetric adjacency matrix (‘dgCMatrix‘).

components Integer vector component id (length n). Isolates are their own components.

isolates Logical vector (length n).

build_network *Build a road network graph from sf LINESTRING data*

Description

Build a road network graph from sf LINESTRING data

Usage

```
build_network(
  roads_sf,
  crs_out = 3857,
  node_intersections = FALSE,
  snap_tol = 0,
  simplify = TRUE
)
```

Arguments

`roads_sf`	An sf object with LINESTRING geometry
`crs_out`	Integer EPSG code for projected CRS
`node_intersections`	Logical; if TRUE, "node" the linework by splitting at interior intersections/junctions (via ‘sf::st_union()‘), so that crossings and T-junctions become graph nodes even when they are not endpoints. This may increase the number of edge segments.
`snap_tol`	Nonnegative numeric; optional snapping tolerance (in projected CRS units) used to merge nearly identical endpoints. Use 0 to disable.
`simplify`	Logical; if TRUE, remove self-loops and parallel edges.

Value

A list with components:

- roads: cleaned sf object
- nodes: sf POINT object with node_id
- edges: sf LINESTRING object with from, to, length
- graph: igraph object
- A: sparse adjacency matrix

car_precision *CAR precision matrix from an adjacency matrix*

Description

Constructs the precision matrix for an intrinsic CAR (ICAR) or proper CAR model:

$$Q = \tau(D - \rho A), \quad D = \mathrm{diag}(A\mathbf{1}).$$

Usage

```
car_precision(
  A,
  type = c("icar", "proper"),
  rho = 0.99,
  tau = 1,
  symmetrize = FALSE,
  check = TRUE
)
```

Arguments

`A` Square adjacency/weight matrix (base matrix or a ‘Matrix‘ sparse type). Diagonal entries are ignored (set to 0).

`type` Either ‘"icar"‘ or ‘"proper"‘.

`rho` Spatial dependence parameter for proper CAR. Ignored for ICAR.

`tau` Positive scalar precision multiplier.

`symmetrize` If ‘TRUE‘, replaces ‘A‘ by ‘(A + t(A))/2‘ before construction.

`check` If ‘TRUE‘, performs basic validation and warnings.

Details

For ICAR, set ‘type = "icar"‘ (internally uses $\rho = 1$). For proper CAR, set ‘type = "proper"‘ and choose ‘rho‘ so that $D - \rho A$ is positive definite (no automatic spectral checks are performed).

Value

A symmetric sparse precision matrix ‘Q‘ (class ‘"dsCMatrix"‘).

Examples

```
A <- matrix(0, 4, 4)
A[1,2] <- A[2,1] <- 1
A[2,3] <- A[3,2] <- 1
A[3,4] <- A[4,3] <- 1
Q_icar <- car_precision(A, type = "icar", tau = 1)
Q_prop <- car_precision(A, type = "proper", rho = 0.9, tau = 2)
```

fetch_osm_roads *Fetch road geometries from OpenStreetMap*

Description

Convenience wrapper around ‘osmdata‘ to download road geometries and return an ‘sf‘ object that can be passed into ‘roads_to_segments()‘.

Usage

```
fetch_osm_roads(
  place,
  key = "highway",
  value = NULL,
  extra_tags = NULL,
  layer = c("osm_lines", "osm_multilines", "osm_polygons", "osm_multipolygons"),
  quiet = TRUE,
  ...
)
```

Arguments

place	A character place name (passed to ‘osmdata::getbb()‘) or a bounding box object accepted by ‘osmdata::opq()‘.
key	OSM feature key to query. Default is ‘"highway"‘.
value	Optional character vector of OSM feature values. For example, ‘c("primary", "secondary")‘.
extra_tags	Optional named list of additional tags passed to ‘osmdata::add_osm_feature()‘.
layer	Which OSM layer to return. Defaults to ‘"osm_lines"‘.
quiet	Logical; suppress osmdata messages. Default TRUE.
...	Additional arguments passed to ‘osmdata::osmdata_sf()‘.

Value

An ‘sf‘ object with road geometries.

fit_car *Fit Gaussian CAR / ICAR regression via Gibbs sampling*

Description

Fits a Gaussian regression with a CAR/ICAR latent effect: $y = X\beta + x + \epsilon$ with $\epsilon \sim N(0, \sigma^2 I)$ and $x \sim N(0, Q^{-1})$, where $Q = \tau(D - \rho A)$ and $D = diag(A1)$. For ICAR, $\rho = 1$.

Usage

```
fit_car(
  y,
  A,
  X = NULL,
  type = c("icar", "proper"),
  rho = 0.99,
  tau = 1,
  n_iter = 2000,
  burn_in = floor(n_iter/2),
  thin = 1,
  beta_init = NULL,
  x_init = NULL,
  sigma2_init = NULL,
  b0 = NULL,
  B0 = NULL,
  a0 = 2,
  b0_sigma = 1,
  center_icar = TRUE,
  verbose = FALSE
)
```

Arguments

`y`	Numeric response vector of length `n`.
`A`	Square `n x n` adjacency/weight matrix (base matrix or `Matrix`). Diagonal entries are ignored.
`X`	Optional `n x p` design matrix. If `NULL`, no regression is fit.
`type`	Either `"icar"` or `"proper"`.
`rho`	Spatial dependence parameter for proper CAR. Ignored for ICAR.
`tau`	Positive scalar precision multiplier.
`n_iter`	Total MCMC iterations.
`burn_in`	Number of initial iterations to discard.
`thin`	Keep every `thin`-th draw after burn-in.
`beta_init`	Optional initial β (length `p`).

`x_init` Optional initial latent field x (length `n`).

`sigma2_init` Optional initial σ^2 (positive scalar).

`b0` Prior mean for β (length `p`). Default is zero vector.

`B0` Prior covariance for β (`p x p`). Default is large diagonal.

`a0` Shape parameter for inverse-gamma prior on σ^2.

`b0_sigma` Scale parameter for inverse-gamma prior on σ^2.

`center_icar` Logical; if `TRUE` and `type="icar"`, center x to sum-to-zero within each connected component.

`verbose` Logical; print coarse progress updates.

Details

The sampler updates x, β (if `X` is provided), and σ^2 using Gibbs steps.

Value

A list of class `"trafficCAR_fit"` with elements:

`draws` List with MCMC draws `x`, `beta`, `sigma2`.

`keep` Iteration indices that were saved.

`type`**,** `rho`**,** `tau` Model hyperparameters used.

y = X beta + x + epsilon

`fit_traffic` *Fit a Gaussian CAR traffic model (speed or travel time)*

Description

This is a thin wrapper around `fit_car()` that: 1) preprocesses the outcome (log/per-distance options), 2) fits the Gaussian CAR model, 3) returns a traffic-flavored object that can be augmented back onto roads.

Usage

```
fit_traffic(
  data,
  roads = NULL,
  A = NULL,
  segment_id_col = "segment_id",
  outcome = c("speed", "travel_time"),
  outcome_col = NULL,
  distance_col = NULL,
  per_distance = FALSE,
  transform = c("log", "identity"),
  X = NULL,
  ...
)
```

Arguments

`data` data.frame with at least ‘segment_id‘ and the outcome column.

`roads` optional; an sf object or similar that contains adjacency info already used by your ‘fit_car()‘ pipeline (depends on your package design).

`A` adjacency matrix or object accepted by ‘fit_car()‘ (recommended explicit).

`segment_id_col` character; column in ‘data‘ used to join back to roads.

`outcome` character; one of "speed" or "travel_time".

`outcome_col` optional character; if NULL uses ‘outcome‘.

`distance_col` optional character; used only for travel_time when per_distance=TRUE.

`per_distance` logical; only for travel_time.

`transform` character; "log" or "identity".

`X` optional design matrix; if NULL, uses intercept-only.

`...` passed to ‘fit_car()‘ (e.g., type="proper"/"icar", rho, priors, n_iter, burn, etc.)

Value

An object of class ‘traffic_fit‘ containing the underlying fit and transform metadata.

`icar_sum_to_zero` *Apply sum-to-zero constraint to ICAR precision*

Description

Projects onto the subspace orthogonal to the constant vector.

Usage

```
icar_sum_to_zero(Q)
```

Arguments

`Q` ICAR precision matrix

Value

Constrained precision matrix

`intrinsic_car_precision`

Intrinsic CAR (ICAR) precision matrix

Description

Constructs the intrinsic CAR precision matrix

$$Q = \tau\, s(D - A),$$

where s is a scaling constant chosen so that the geometric mean of the marginal variances equals 1.

Usage

```
intrinsic_car_precision(
  A,
  tau = 1,
  scale = TRUE,
  symmetrize = FALSE,
  check = TRUE
)
```

Arguments

`A`	Square adjacency/weight matrix.
`tau`	Positive scalar precision multiplier.
`scale`	Logical; if ‘TRUE‘, applies Besag scaling.
`symmetrize`	If ‘TRUE‘, replaces ‘A‘ by ‘(A + t(A))/2‘.
`check`	If ‘TRUE‘, performs basic validation and warnings.

Details

The resulting precision matrix is singular with rank deficiency equal to the number of connected components.

Value

A symmetric sparse precision matrix (‘"dsCMatrix"‘).

References

Sørbye, S. H. and Rue, H. (2014). Scaling intrinsic Gaussian Markov random field priors.

map_roads_interactive *Interactive map of road-segment traffic measures*

Description

Displays standard traffic quantities such as predicted speed, predicted volume, or relative congestion on an interactive map.

Usage

```
map_roads_interactive(
  sf_aug,
  value = c("predicted_speed", "predicted_volume", "relative_congestion"),
  engine = "leaflet"
)
```

Arguments

sf_aug	An ‘sf‘ object returned by ‘augment_roads()‘.
value	Character scalar. One of: ‘"predicted_speed"‘, ‘"predicted_volume"‘, ‘"relative_congestion"‘.
engine	Currently only ‘"leaflet"‘ is supported.

Value

A leaflet widget.

map_roads_interactive_layers *Interactive map with multiple standard traffic layers*

Description

Interactive map with multiple standard traffic layers

Usage

```
map_roads_interactive_layers(
  sf_aug,
  values = c("predicted_speed", "relative_congestion")
)
```

Arguments

sf_aug	sf object with road geometries
values	Character vector of traffic measures to include.

Value

leaflet widget

moran_residuals *Moran's I for trafficCAR residuals*

Description

Computes Moran's I statistic for model residuals using the model adjacency.

Usage

```
moran_residuals(
  fit,
  type = c("raw", "structured", "unstructured"),
  nsim = 199,
  method = c("analytic", "permutation")
)
```

Arguments

fit	A 'traffic_fit' object.
type	Residual type: "raw" or "unstructured".
nsim	Number of permutations for permutation test.
method	"analytic" or "permutation".

Value

An object of class 'traffic_moran'.

plot_mcmc_diagnostics *MCMC diagnostic plots*

Description

MCMC diagnostic plots

Usage

```
plot_mcmc_diagnostics(fit)
```

Arguments

fit	traffic_fit

Value

A list with components:

plot A ‘ggplot‘ object of diagnostic summaries.

summary A data frame with columns ‘parameter‘ and ‘ess‘, giving the effective sample size for each parameter..

`plot_observed_fitted` *Plot observed vs predicted traffic values*

Description

Plot observed vs predicted traffic values

Usage

```
plot_observed_fitted(fit, data)
```

Arguments

`fit`	traffic_fit
`data`	data.frame

Value

ggplot

`plot_predicted` *Plot predicted traffic outcome on road network*

Description

Plot predicted traffic outcome on road network

Usage

```
plot_predicted(fit, roads)
```

Arguments

`fit`	traffic_fit
`roads`	sf with segment_id

Value

ggplot

plot_relative_congestion *Plot relative congestion on road network*

Description

Shows systematic deviations after accounting for covariates.

Usage

```
plot_relative_congestion(fit, roads)
```

Arguments

`fit`	traffic_fit
`roads`	sf

Value

ggplot

plot_roads_static *Static map of road-segment traffic measures*

Description

Static map of road-segment traffic measures

Usage

```
plot_roads_static(
  sf_aug,
  value = c("predicted_speed", "predicted_volume", "relative_congestion")
)
```

Arguments

`sf_aug`	sf object with road geometries
`value`	One of "predicted_speed", "predicted_volume", or "relative_congestion"

Value

ggplot object

plot_traffic_map *Quick map helper for augmented roads*

Description

Plots road geometries colored by an augmented numeric column (e.g., posterior mean predictions or relative congestion).

Usage

```
plot_traffic_map(roads_aug, fill = c("predicted_mean", "relative_congestion"))
```

Arguments

`roads_aug`	An ‘sf‘ object returned by [augment_roads()].
`fill`	Character scalar. Which column of ‘roads_aug‘ to map. One of ‘"predicted_mean"‘ or ‘"relative_congestion"‘.

Value

An invisible copy of ‘roads_aug‘, returned as an ‘sf‘ object with the augmented columns. The function is called for its plotting side effect.

ppc_summary *Posterior predictive checks for trafficCAR fits*

Description

Computes simple posterior predictive checks comparing observed statistics to replicated data: mean, variance, and tail probabilities.

Usage

```
ppc_summary(fit, stats = c("mean", "var", "tail"), probs = c(0.05, 0.95))
```

Arguments

`fit`	A ‘traffic_fit‘ object.
`stats`	Statistics to compute: "mean", "var", "tail".
`probs`	Tail probabilities for "tail" statistic.

Value

An object of class ‘traffic_ppc‘.

residuals.traffic_fit *Residuals for trafficCAR fits*

Description

Computes raw, structured (spatial), or unstructured residuals from a fitted trafficCAR model.

Usage

```
## S3 method for class 'traffic_fit'
residuals(object, type = c("raw", "structured", "unstructured"), ...)
```

Arguments

`object`	A ‘traffic_fit‘ object.
`type`	Residual type: "raw", "structured", or "unstructured".
`...`	Unused.

Value

Numeric vector of residuals.

roads_datasets *Example road network datasets*

Description

A collection of example road network datasets provided as ‘sf‘ objects. These datasets are intended for demonstration, testing, and benchmarking.

Format

An object of class `sf`.

Details

Each dataset contains LINESTRING road geometries suitable for use with `roads_to_segments()` and related functions.

Included datasets:

roads_small Small example road network.

roads_cstat Road network for College Station, TX.

roads_to_segments *Convert road geometries to modeling segments*

Description

Takes an ‘sf‘ object of LINESTRING/MULTILINESTRING road geometries and returns a segment-level ‘sf‘ with stable segment IDs and metric lengths.

Usage

```
roads_to_segments(
  roads,
  crs_m = 3857,
  keep_attrs = NULL,
  drop_zero = TRUE,
  split_at_intersections = FALSE,
  verbose = FALSE
)
```

Arguments

`roads`	An ‘sf‘ object with LINESTRING or MULTILINESTRING geometries.
`crs_m`	Metric CRS used for length calculation (and intersection splitting) when ‘roads‘ is lon/lat. Default 3857. For best accuracy, pass a local UTM EPSG.
`keep_attrs`	Optional character vector of non-geometry columns to keep. If ‘NULL‘, keeps all attributes.
`drop_zero`	Logical; drop segments with non-positive length. Default TRUE.
`split_at_intersections`	Logical; if TRUE, split lines at all intersections. Implemented via GEOS noding (‘sf::st_union‘ + ‘sf::st_cast‘) Default FALSE.
`verbose`	Logical; emit simple messages about dropped rows. Default FALSE.

Details

v1 behavior: * Drops Z/M dimensions * Casts MULTILINESTRING -> LINESTRING (one row per linestring) * Optionally splits at intersections (noding) when ‘split_at_intersections=TRUE‘ * Computes ‘length_m‘ in meters (projects if lon/lat) * Drops empty and (optionally) zero-length segments

Value

An ‘sf‘ with columns: * ‘seg_id‘ integer 1..n * ‘length_m‘ numeric meters * geometry LINESTRING plus kept attributes.

sample_proper_car *Gibbs sampler for a proper CAR latent Gaussian model*

Description

Model:

$$y \mid x, \tau \sim N(x, \tau^{-1} I)$$

$$x \mid \kappa \sim N(0, (\kappa Q)^{-1}), \quad Q = D - \rho A \text{ (proper CAR)}$$

$$\tau \sim \mathrm{Gamma}(a_\tau, b_\tau) \quad \text{(shape-rate)}$$

$$\kappa \sim \mathrm{Gamma}(a_\kappa, b_\kappa) \quad \text{(shape-rate)}$$

Usage

```
sample_proper_car(
  y,
  A,
  rho = 0.99,
  n_iter,
  burn = 0L,
  thin = 1L,
  a_tau = 1,
  b_tau = 1,
  a_kappa = 1,
  b_kappa = 1,
  init = NULL,
  symmetrize = FALSE,
  check = TRUE
)
```

Arguments

`y`	Numeric vector of observations (length n).
`A`	Adjacency matrix (dense or sparse). Diagonal ignored.
`rho`	Proper CAR dependence parameter (must satisfy car_precision checks).
`n_iter`	Integer number of iterations.
`burn`	Integer burn-in iterations to drop (default 0).
`thin`	Integer thinning interval (default 1).
`a_tau, b_tau`	Gamma(shape, rate) prior for tau.
`a_kappa, b_kappa`	Gamma(shape, rate) prior for kappa.
`init`	Optional list with elements x, tau, kappa.
`symmetrize`	Passed to car_precision().
`check`	Passed to car_precision().

Value

List with x (matrix), tau, kappa, and settings.

simplify_network *Simplify a built network object by removing parallel edges (and loops)*

Description

Simplify a built network object by removing parallel edges (and loops)

Usage

```
simplify_network(net, keep_edge = c("first", "shortest"))
```

Arguments

net	A network list returned by [build_network()].
keep_edge	Which edge to keep when multiple edges connect the same unordered node pair. One of "first" or "shortest".

Value

A network list with updated ‘edges‘, ‘graph‘, and ‘A‘ (and the same ‘nodes‘).

weights_from_adjacency *Construct spatial weights matrix*

Description

Construct spatial weights matrix

Usage

```
weights_from_adjacency(
  A,
  style = c("binary", "row-standardized"),
  symmetrize = FALSE
)
```

Arguments

A	Adjacency matrix.
style	One of "binary" or "row-standardized".
symmetrize	Passed to adjacency coercion.

Value

Sparse weight matrix.

Index