Package 'spCF'

August 28, 2026

Type Package

Title Coarse-to-Fine Spatial and Spatio-Temporal Modeling

Version 0.2.1

Depends R (>= 4.1.0)

Imports FNN, fields, nloptr, dbscan, withr, Matrix, Rcpp

LinkingTo Rcpp

Suggests sp, sf, knitr, rmarkdown, CARBayesdata, ranger, lightgbm, shiny, bslib, leaflet, terra, testthat (>= 3.0.0)

Config/testthat/edition 3

Description
Provides functions for coarse-to-fine spatial and spatio-temporal modeling, enabling fast prediction, regression, and uncertainty quantification for moderate to large datasets. For methodological details, see Murakami et al. (2026) <doi:10.1111/gean.70034> and related work on generalized linear, downscaling, and dynamic spatio-temporal extensions.

URL https://github.com/dmuraka/spCF

BugReports https://github.com/dmuraka/spCF/issues

License GPL (>= 2)

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Daisuke Murakami [aut, cre],
Alexis Comber [aut],
Takahiro Yoshida [aut],
Narumasa Tsutsumida [aut],
Chris Brunsdon [aut],
Tomoki Nakaya [aut],
Jose Luis Blanco-Claraco [ctb, cph] (Author of the bundled 'nanoflann' C++ library (src/nanoflann.h)),
Marius Muja [cph] (Copyright holder of the bundled 'nanoflann' C++

library),
David G. Lowe [cph] (Copyright holder of the bundled 'nanoflann' C++
library)

Maintainer Daisuke Murakami <dmuraka@ism.ac.jp>

Repository CRAN

Date/Publication 2026-08-28 17:40:02 UTC

Contents

cf_dglm 2
cf_dglm_hv 5
cf_downscale 8
cf_downscale_hv 10
cf_glm 12
cf_glm_hv 15
cf_lm 17
cf_lm_hv 20
spCFmap 22
sp_scalewise 23

Index **25**

`cf_dglm` *Coarse-to-fine dynamic (space-time) spatial GLMMs (CF-DGLMMs)*

Description

Prediction and regression via a separable space-time cascade. Given the scales selected by `cf_dglm_hv`, the model is refitted on the full sample and predictions (with standard deviations) are produced at sample and, optionally, prediction sites. The link-scale linear predictor is $g(\mu_{i,t}) = x'_{i,t}\beta + \sum_k f_k(s_i, t) +$ offset, where each scale-k field f_k couples a per-knot AR(1) Kalman smoother in time with kernel kriging in space.

Usage

```
cf_dglm(
  y,
  x = NULL,
  coords,
  time,
  offset = NULL,
  x0 = NULL,
  coords0 = NULL,
  time0 = NULL,
  offset0 = NULL,
  mod_hv,
```

```
  robust_se = TRUE,
  sill_cap = TRUE,
  se_type = c("prediction", "mean"),
  se_method = c("opt", "classic")
)
```

Arguments

y	Vector of response variables (N x 1).
x	Matrix of covariates (N x K).
coords	Matrix of 2-dimensional point coordinates (N x 2). The space-time panel may be unbalanced (observed locations may differ across time points).
time	Vector of time indices (N x 1); must use the same time points as in cf_dglm_hv.
offset	Optional. Offset variable (N x 1), consistent with glm.
x0	Optional. Matrix of covariates at prediction sites (N0 x K).
coords0	Optional. Coordinates at prediction sites (N0 x 2).
time0	Optional. Time indices at prediction sites (N0 x 1). May include time points with no observations: interior time points absent from the training data are interpolated, and time points beyond the last observed one are forecast, via the per-knot AR(1) predict step (the Kalman gain is zero where a time column carries no data). Such time points are added to the working time grid, so predicting at interior gaps slightly re-spaces the AR(1) grid; forecasting beyond the last observed point leaves the training-time fit unchanged.
offset0	Optional. Offset at prediction sites (N0 x 1).
mod_hv	Output object of cf_dglm_hv.
robust_se	Logical; if TRUE (default), the constant-coefficient standard errors (and the coefficient-uncertainty term of the predictive SE) use a spatial-block cluster-robust sandwich that accounts for the cascade field being a correlated random component. The naive model-based covariance treats the field as a known offset and severely understates the SEs; the robust version restores near-nominal coverage. Set FALSE for the naive vcov(glm) SEs.
sill_cap	Logical; if TRUE (default), the field component of the predictive variance is capped at the marginal variance of the fitted total field (the "sill"), var(sum_rz_r) on the link scale. This prevents the gPoE variance from diverging in deep extrapolation, mirroring a stationary GP that reverts to the prior marginal variance far from data. The cap is applied to the total field variance (never per scale) and leaves the predictive mean, RMSE, and the coefficient-uncertainty term unchanged. It is disabled automatically for binomial responses. Set FALSE to leave the field variance uncapped.
se_type	Type of predictive uncertainty in pred/pred_q. "prediction" (default) returns the holdout-calibrated OBSERVATION predictive for a new data point (Gaussian: mean uncertainty + residual variance; Poisson: negative-binomial count predictive; binomial: temperature-calibrated probability with pred_sd = sqrt(p(1-p))); the signal versions are kept in pred_signal/pred_q_signal. "mean" returns the signal (mean) uncertainty only (previous behaviour). See other$calibration.

`se_method` Cluster-robust coefficient-SE estimator (used when `robust_se = TRUE`). `"opt"` (default) splits the sandwich meat into a field-removed observation-noise part and a field part that adds the calibrated field variance back with a within-block `exp(-d/h)` correlation (`h` = median committed bandwidth); this is near-nominal. A refit-free leverage leave-one-out ceiling then caps the field term, preventing over-coverage for count (Poisson) responses while leaving already-calibrated families unchanged. `"classic"` keeps the realised field inside the working residual (the previous behaviour), which is valid but conservative.

Details

The full-sample fit is a SINGLE coarse-to-fine cascade sweep, mirroring the relationship between `cf_glm` and `cf_glm_hv`: it reuses the same single greedy sweep that `cf_dglm_hv` performs for scale selection, plus prediction. Within the sweep, for each band (coarse to fine) the GLM working response/weights are refreshed (IRLS folded into the sweep, as `cf_glm`'s per-band `glm()` does), the scale is fit and accumulated, and the constant and time-varying coefficients are backfit. (The earlier outer-IRLS implementation is archived as `cf_dglm_iter` under misc/.)

Value

A list (class `"cf_dglm"`) mirroring `cf_glm`: `beta`, `sd_summary`, `e_summary`, `pred`, `pred0`, `pred_q`, `pred0_q`, `bands`, `Z`, `Z_sd`, `Z0`, `Z0_sd`, `other`, `call`, plus

beta_tv, beta_tv_sd Time-varying coefficients and their standard deviations, one row per time point and one column per covariate named in `tvc` (plus a `time` column). `NULL` when `tvc` was not used in `cf_dglm_hv`.

pred_signal, pred_q_signal The signal (mean) predictive kept alongside the observation predictive when `se_type = "prediction"`.

The temporal parameters of the fitted cascade are in `other$rho` (AR(1) autocorrelation), `other$Q` (innovation variance) and `other$tau` (holdout-calibrated field-variance factor); the first two are shown by `print`.

Author(s)

Daisuke Murakami

References

Murakami, D. (2026). Fast covariance-free spatiotemporal modeling via coarse-to-fine learning. *ArXiv preprint*.

See Also

`cf_dglm_hv`, `cf_glm`

Examples

```
### Monthly PM10 at 63 German background stations, 2001-2005 (the data set
### behind the "Demo (air, space-time)" entry of spCFmap(); see the
### spCF_dglm vignette for a fuller walk-through).
require(sf)
air    <- read.csv(system.file("shiny", "spCFmap",
                               "example_spacetime_air.csv", package = "spCF"))
pts    <- st_as_sf(air, coords = c("lon", "lat"), crs = 4326)
coords <- st_coordinates(st_transform(pts, 25832))  # UTM 32N, in metres

### The annual cycle is a fixed effect; the space-time process takes the rest
x      <- data.frame(sin12 = sin(2 * pi * air$month / 12),
                     cos12 = cos(2 * pi * air$month / 12))

### Holdout validation optimizing the number of spatial scales
mod_hv <- cf_dglm_hv(y = air$pm10, x = x, coords = coords, time = air$time)

### Space-time modeling; the 63 stations are also predicted one month beyond
### the data (time = 61), which the AR(1) predict step turns into a forecast
uni    <- !duplicated(air$station)
n0     <- sum(uni)
coords0<- coords[uni, ]
time0  <- rep(61, n0)
x0 = data.frame(sin12 = rep(sin(2 * pi / 12), n0),
                                  cos12 = rep(cos(2 * pi / 12), n0))

mod    <- cf_dglm(y = air$pm10, x = x, coords = coords, time = air$time,
                  x0 = x0, coords0 = coords0, time0 = time0,
                  mod_hv = mod_hv)
mod

round(mod$bands / 1000, 1)              # accepted bandwidths, in km
round(c(rho = mod$other$rho, Q = mod$other$Q), 3)  # AR(1) parameters

### Mapping the forecast for January 2006 at the station locations
fc     <- st_as_sf(data.frame(pred = mod$pred0$pred, coords[uni, ]),
                   coords = c("X", "Y"), crs = 25832)
plot(fc[, "pred"], pch = 20, cex = 1.3, axes = TRUE, key.pos = 4, nbreaks = 20)

### Multiscale extraction, averaged over the observed months
mod_s1 <- sp_scalewise(mod, bw_range = c(150000, Inf))  # large scale
mod_s2 <- sp_scalewise(mod, bw_range = c(0, 150000))    # small scale

### The same fit, explored interactively over a basemap
# spCFmap(mod, crs = 25832)
```

---

cf_dglm_hv *Holdout validation for coarse-to-fine dynamic (space-time) spatial GLMMs*

Description

Trains a coarse-to-fine dynamic spatial GLMM (CF-DGLMM) and selects the spatial scales of a separable space-time cascade through progressive holdout validation. The companion `cf_dglm` refits the selected structure on the full sample and predicts. The model decomposes the link-scale linear predictor as $g(\mu_{i,t}) = x'_{i,t}\beta + \sum_k f_k(s_i, t) +$ offset, where each scale-k field f_k is a per-knot AR(1) Kalman smoother in time combined with kernel kriging in space.

Usage

```
cf_dglm_hv(
  y,
  x = NULL,
  coords,
  time,
  offset = NULL,
  train_rat = 0.75,
  id_train = NULL,
  alpha = 0.9,
  kernel = "exp",
  family = gaussian(),
  rho = NULL,
  Q = NULL,
  tvc = NULL,
  q_tvc = NULL,
  seed = 1234
)
```

Arguments

`y`	Vector of response variables (N x 1) including continuous, count, and binary responses, following an exponential family distribution.
`x`	Matrix of covariates (N x K).
`coords`	Matrix of 2-dimensional point coordinates (N x 2). Rows sharing the same coordinates are treated as repeated observations of one location across time. The space-time panel may be unbalanced: the set of observed locations is allowed to differ from one time point to another (knots are placed on the union of locations and the per-knot AR(1) smoother bridges time points at which a knot has no nearby observation).
`time`	Vector of time indices (N x 1) identifying the time point of each observation. Any sortable type (integer, numeric, Date) is accepted.
`offset`	Optional. Vector of offset variable (N x 1) to be included in the linear predictor, consistent with `glm`.
`train_rat`	Training sample ratio (default: 0.75). Holdout is performed at the *location* level: a subset of locations (and all their time points) is held out for validation.
`id_train`	Optional. If specified, the corresponding samples are used as training samples; otherwise locations are chosen based on `train_rat`.

alpha	Decay ratio of the kernel bandwidth in the coarse-to-fine training (default: 0.9).
kernel	Kernel type for spatial dependence. "exp" for the exponential kernel (default) and "gau" for the Gaussian kernel.
family	Error distribution and link function, consistent with the family argument of glm. Functionality has been confirmed for gaussian(), poisson(), and binomial().
rho, Q	Optional AR(1) temporal parameters (autocorrelation and innovation variance). When NULL (default) a single global (rho, Q) is estimated by maximum marginal likelihood.
tvc	Optional. Covariates whose regression coefficients are allowed to vary over time, given as covariate names or as integer column indices into x. The remaining coefficients are constant. The intercept is always kept constant (a time-varying intercept is confounded with the temporal mean of the spatial field). NULL (default) keeps all coefficients constant.
q_tvc	Optional. Innovation (drift) variance of the random walk followed by the time-varying coefficients. When NULL (default) it is estimated from the data.
seed	Random seed for the training/validation split and knot placement (default 1234). Set to NULL for a random split.

Value

A list of class "cf_dglm_hv" with the following elements:

loss_hv Holdout deviance of the selected model, evaluated at the validation locations. Fits of the same data share the same split, so this value compares models directly, whatever number of scales each selected.

loss_hv_all The validation loss after every learning step.

e_summary Out-of-sample accuracy at the validation locations of the model trained on the training locations only: deviance-based pseudo R-squared (validation_Pseudo-R2, ordinary R-squared in the Gaussian case), validation_RMSE and validation_MAE. Unlike the e_summary of cf_dglm, which scores the full-sample refit at those same points, this one never saw them.

val_pred The validation predictions behind e_summary: one row per held-out observation with its location index (loc), time point (time), observed response (y) and predicted mean (pred) on the response scale.

id_train Row indices of the training observations.

other Internal objects reused by cf_dglm.

call The matched call.

Author(s)

Daisuke Murakami

References

Murakami, D. (2026). Fast covariance-free spatiotemporal modeling via coarse-to-fine learning. *ArXiv preprint*.

See Also

cf_dglm, cf_glm_hv

---

cf_downscale *Coarse-to-fine spatial downscaling (CF-DS)*

Description

Scalable downscaling via CF-DS for predicting disaggregate-level responses from aggregate-level response Y, while ensuring that predictions aggregate exactly to the observed aggregate-level values.

Usage

```
cf_downscale(
  Y,
  x = NULL,
  prop_weight = NULL,
  coords,
  agg_id,
  mod_hv,
  adj = TRUE,
  nonneg = TRUE
)
```

Arguments

Y	Vector of aggregate-level response variables (length N).
x	Matrix of disaggregate-level covariates (n x K), assumed to match the x used in cf_downscale_hv.
prop_weight	Vector of disaggregate-level proportional allocation weights (length n), assumed to match the prop_weight used in cf_downscale_hv. See cf_downscale_hv for the role and examples of choices.
coords	Matrix of disaggregate-level coordinates (n x 2).
agg_id	Area ID for each disaggregate-level unit (length n).
mod_hv	Output object from cf_downscale_hv.
adj	Logical (default TRUE). When TRUE, a per-area multiplicative adjustment is applied to satisfy the aggregation constraint so that the downscaled predictions aggregate exactly to the observed 'Y'. When FALSE, the constraint is satisfied only approximately, which may be preferable when 'Y' contains noise.
nonneg	If TRUE (default), clip negative predictions to zero before the multiplicative adjustment.

Value

A list with the following elements:

beta Regression coefficients, their standard errors, and the lower and upper limits of the 95 percent confidence intervals.

sd_summary Standard deviation of the regression term (xb), spatial processes (spatial_scale1, spatial_scale2,...), and residuals.

e_summary Aggregate-level holdout validation accuracy, evaluated on the validation units: R-squared (validation_R2), root mean squared error (validation_RMSE), and mean absolute error (validation_MAE). All are `NA` when no validation areas are available (e.g. `train_rat = 1`).

pred Predictive mean (`pred`) and standard deviation (`pred_sd`) of the disaggregate-level response. The spatial-process contribution to `pred_sd` is rescaled by a holdout-calibrated factor (stored as `other$tau`) estimated on the validation areas.

bands Bandwidth values for each accepted scale during the holdout validation in `cf_downscale_hv`.

Z Predictive mean of each single-scale spatial process at the disaggregate-level (data.frame; one column per scale).

Z_sd Predictive standard deviation of the single-scale process at the disaggregate-level units (data.frame).

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Chun, Y., Yoshida, T., & Seya, H. (2026). Scalable coarse-to-fine spatial downscaling. *ArXiv preprint*.

See Also

`cf_downscale_hv`, `cf_lm`

Examples

```
set.seed(123)
require(sf); require(CARBayesdata)
data(GGHB.IZ)
data(pollutionhealthdata)
d  <- pollutionhealthdata[pollutionhealthdata$year == 2010, ]
ar <- merge(GGHB.IZ, d, by = "IZ")

### Disaggregate-level data (271 units)
coords <- st_coordinates(suppressWarnings(st_centroid(ar)))
x      <- data.frame(pm10 = ar$pm10, jsa = ar$jsa, price = ar$price)
prop_weight <- as.numeric(ar$expected)

### Aggregate-level data (30 units).
```

```
agg_id <- as.integer(stats::kmeans(coords, centers = 30)$cluster)

### Two types of response variables are possible:
# Y_type = "sum"  : Y_I = sum(response variable for each aggregate unit)
# Y_type = "mean" : Y_I = mean(response variable for each aggregate unit)
Y_type <- "sum"   # change to "mean" for the density-type data
Y      <- as.numeric(stats::aggregate(ar$observed, by = list(agg_id),
                       FUN = if (Y_type == "sum") sum else mean)[, 2])

### Downscaling
mh <- cf_downscale_hv(Y = Y, Y_type = Y_type, x = x,
                      prop_weight = prop_weight,
                      coords = coords, agg_id = agg_id)
md <- cf_downscale(Y = Y, x = x, prop_weight = prop_weight,
                   coords = coords, agg_id = agg_id, mod_hv = mh)

### Mapping
ar$agg_id <- agg_id
agg_poly  <- stats::aggregate(ar["agg_id"], by = list(agg_id = agg_id),
                              FUN = function(z) z[1])
agg_poly$Y<- Y
ar$pred   <- md$pred$pred
plot(agg_poly["Y"], nbreaks = 20, main = "Aggregated data")
plot(ar["pred"], nbreaks = 20, main = "Downscaling result")
```

`cf_downscale_hv` *Holdout validation for the coarse-to-fine spatial downscaling (CF-DS)*

Description

Trains the CF-DS model and selects the number of spatial scales through sequential holdout validation.

Usage

```
cf_downscale_hv(
  Y,
  Y_type = "sum",
  x = NULL,
  prop_weight = NULL,
  coords,
  agg_id,
  train_rat = 0.75,
  id_train = NULL,
  alpha = 0.9,
  kernel = "exp",
  rel_tol = 1e-04,
  seed = 123
)
```

Arguments

Y	Vector of aggregate-level response values (length N).
Y_type	Aggregation type of Y: "sum" for extensive (count-like) data (e.g., population) or "mean" for intensive (density-like) data (e.g., population density, average temperature).
x	Matrix of disaggregate-level covariates (n x K).
prop_weight	Vector of disaggregate-level proportional allocation weights (length n) used to distribute the aggregate-level response across the disaggregate-level units. When Y_type="mean", prop_weight should corresponding to the denominator of the intensive response variable. Examples include residential land area for population downscaling, population for morbidity downscaling, and NULL (= rep(1, n)) for temperature downscaling.
coords	Matrix of disaggregate-level coordinates (n x 2).
agg_id	Area ID for each disaggregate-level unit (length n).
train_rat	Ratio of the aggregate-level units used for model training (default 0.75) in the holdout validation.
id_train	Optional. If specified, the corresponding aggregate-level units are used as training units. Otherwise, training units are chosen based on 'train_rat'.
alpha	Decay ratio of the kernel bandwidth in the coarse-to-fine training (default: 0.9). Values closer to one make the optimization more stringent but increase computation time.
kernel	Kernel type for modeling spatial dependence. '"exp"' for the exponential kernel (default) and '"gau"' for the Gaussian kernel.
rel_tol	Relative improvement threshold for validation SSE (default 1e-4). At each scale, the spatial process is retained only if validation SSE improves by more than rel_tol; otherwise a stopping counter is incremented, and learning stops once 5 consecutive scales fail to improve. Larger values stop earlier, whereas smaller values allow finer scales to be selected.
seed	Random seed used for the training/validation split when 'id_train' is not supplied. Default is '123'. Set to 'NULL' to allow a different split at each call (useful for assessing split sensitivity).

Value

A list with the following elements:

sse_hv Final sum-of-squared error (SSE) for validation samples.

sse_hv_all SSEs obtained at each learning step.

id_train ID of training aggregate-level units.

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Chun, Y., Yoshida, T., & Seya, H. (2026). Scalable coarse-to-fine spatial downscaling. *ArXiv preprint*.

See Also

cf_downscale, cf_lm_hv

---

cf_glm *Coarse-to-fine spatial generalized linear mixed models (CF-GLMMs)*

Description

Scalable prediction, regression, and multiscale analysis via CF-GLMMs.

Usage

```
cf_glm(
  y,
  x = NULL,
  coords,
  offset = NULL,
  x0 = NULL,
  coords0 = NULL,
  offset0 = NULL,
  mod_hv,
  robust_se = TRUE,
  se_type = c("prediction", "mean"),
  se_method = c("opt", "classic")
)
```

Arguments

y	Vector of response variables (N x 1), including continuous, count, and binary responses following an exponential family distribution.
x	Matrix of covariates (N x K).
coords	Matrix of 2-dimensional point coordinates (N x 2).
offset	Optional. Vector of offset variables (N x 1) included in the linear predictor, consistent with glm.
x0	Optional. Matrix of covariates at prediction sites (N0 x K).
coords0	Optional. Matrix of 2-dimensional point coordinates at prediction sites (N0 x 2).
offset0	Optional. Vector of offset variables at prediction sites (N0 x 1)
mod_hv	Output object of the cf_glm_hv function.

`robust_se` If `TRUE` (default), coefficient standard errors and predictive uncertainty are computed using a cluster-robust sandwich estimator accounting for local spatial correlation. Set `FALSE` to use naive SEs (not recommended).

`se_type` Type of predictive uncertainty in `pred`/`pred_q`. `"prediction"` (default) returns the OBSERVATION predictive for a new data point, holdout-calibrated on the `cf_glm_hv` validation samples (Gaussian: mean uncertainty + residual variance, split-conformal SD scaling; Poisson: negative-binomial count predictive; binomial: temperature -calibrated probability with `pred_sd = sqrt(p(1-p))`). The mean/signal versions are kept in `pred_signal`/`pred_q_signal`. `"mean"` returns the signal (mean) uncertainty only (previous behaviour). See `other$calibration` for the fitted calibration.

`se_method` Cluster-robust coefficient-SE estimator (used when `robust_se = TRUE`). `"opt"` (default) splits the sandwich meat into a field-removed observation-noise part and a field part that adds the calibrated field variance back with a within-block `exp(-d/h)` correlation (`h` = median committed bandwidth); this is near-nominal. A refit-free leverage leave-one-out ceiling then caps the field term, preventing over-coverage for count (Poisson) responses while leaving already-calibrated families unchanged. `"classic"` keeps the realised field inside the working residual (the previous behaviour), which is valid but conservative.

Value

A list with the following elements:

beta Regression coefficients, their standard errors, and the lower and upper limits of the 95 percent confidence intervals.

sd_summary Standard deviation of the regression term (xb), spatial process (spatial_scale1, spatial_scale2,...), additional learning, and residuals.

e_summary Holdout validation accuracy evaluated on the validation samples: R-squared (validation_Pseudo-R2), root mean squared error (validation_RMSE), and mean absolute error (validation_MAE).

pred Predictive means and standard deviations (sample sites). The spatial-process contribution to the predictive SD is rescaled by a holdout-calibrated factor (stored as `other$tau`) estimated on the validation samples.

pred0 Predictive means and standard deviations (prediction sites).

pred_q Predictive quantiles on the response scale at the sample sites. A data frame whose columns `q0.005`, `q0.025`, `q0.05`, `q0.1`, ..., `q0.9`, `q0.95`, `q0.975`, `q0.995` give the corresponding quantile levels, obtained by Gaussian approximation on the link scale followed by inverse-link transformation.

pred0_q Predictive quantiles on the response scale at the prediction sites. Column structure is identical to `pred_q`. `NULL` when prediction sites are not supplied.

bands Bandwidth values for each scale. The i-th bandwidth corresponds to the i-th column of the Z matrix.

Z Predictive mean of the spatial process at each scale (sample sites; list).

Z_sd Predictive standard deviation of the spatial process at each scale (sample sites; list).

Z0 Predictive mean of the spatial process at each scale (prediction sites; list).

Z0_sd Predictive standard deviation of the spatial process at each scale (prediction sites; list).

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Comber, A., Yoshida, T., Tsutsumida, N., Brunsdon, C., & Nakaya, T. (2025). Coarse-to-fine spatial GLMMs for scalable prediction and multiscale analysis. *ArXiv preprint*, 2605.01157. https://doi.org/10.48550/arXiv.2605.01157

See Also

cf_glm_hv, sp_scalewise

Examples

```
################ Example 1: Count data modeling/Disease mapping/smoothing
set.seed(1234)
require( CARBayesdata )
require( sf )
data(pollutionhealthdata)
data(GGHB.IZ)

### Data
dat      <- pollutionhealthdata[pollutionhealthdata$year==2011,]
y        <- dat[,"observed"]             # count data
x        <- dat[,c("pm10","jsa","price")]
offset   <- log(dat[,"expected"])
coords   <- st_coordinates(st_centroid(GGHB.IZ))

### Holdout validation optimizing the number of spatial scales
mod_hv   <- cf_glm_hv(y = y, x = x, offset=offset, coords = coords, family=poisson())

### Spatial modeling and prediction
mod      <- cf_glm(y = y, x = x, coords = coords, mod_hv = mod_hv)
mod

### Mapping predictive mean and standard deviations (SD)
GGHB.IZ$y      <- y
GGHB.IZ$pred   <- mod$pred$pred
GGHB.IZ$pred_sd<- mod$pred$pred_sd
plot(GGHB.IZ[,c("pred")],lwd=0.2,axes=TRUE, key.pos=4,nbreaks=50)   # Predictive mean
plot(GGHB.IZ[,c("pred_sd")],lwd=0.2,axes=TRUE, key.pos=4,nbreaks=50)# Predictive SD

### Multiscale spatial pattern/feature extraction
mod_s1      <- sp_scalewise(mod,bw_range=c(4000,Inf)) # Large scale (4000 <= bandwidth)
mod_s2      <- sp_scalewise(mod,bw_range=c(0,4000))   # Small scale (bandwidth <= 4000)
GGHB.IZ$z1  <- mod_s1$pred$pred
GGHB.IZ$z2  <- mod_s2$pred$pred
plot(GGHB.IZ[,c("z1","z2")],lwd=0.2,axes=TRUE,key.pos=4, nbreaks=50)# Extracted features

### The same fit, explored interactively over a basemap
# spCFmap(mod, crs = 27700)   # crs = the system the coordinates are in
```

```
################ Example 2: Binary data modeling/spatial prediction
set.seed(1234)
require(sp); require(sf)
data(meuse)
data(meuse.grid)

### Data
y        <- ifelse(meuse$ffreq==1, 1, 0 )# binary data
coords   <- meuse[,c("x","y")]
x        <- meuse["dist"]

### Data at prediction sites
coords0  <- meuse.grid[,c("x","y")]
x0       <- meuse.grid["dist"]

### Holdout validation optimizing the number of spatial scales
mod_hv   <- cf_glm_hv(y = y, x = x, coords = coords, family=binomial())

### Spatial modeling and prediction
mod      <- cf_glm(y = y, x=x, coords = coords, x0=x0, coords0 = coords0,
                   mod_hv = mod_hv)
mod

### Mapping predictive mean and standard deviations (SD)
meuse.grid$pred   <- mod$pred0$pred
meuse.grid$pred_sd<- mod$pred0$pred_sd
meuse.grid_sf     <- st_as_sf(meuse.grid, coords = c("x","y"))
plot(meuse.grid_sf["pred"], pch = 15, cex = 0.8, nbreaks = 20)   # Predictive mean
plot(meuse.grid_sf["pred_sd"], pch = 15, cex = 0.8, nbreaks = 20)# Predictive SD

### Multiscale spatial pattern/feature extraction
mod_s1<- sp_scalewise(mod,bw_range=c(1000,Inf)) # Large scale (1000 <= bandwidth)
mod_s2<- sp_scalewise(mod,bw_range=c(0,1000))   # Small scale (0 <= bandwidth <= 1000)
meuse.grid_sf$z1    <- mod_s1$pred0$pred
meuse.grid_sf$z2    <- mod_s2$pred0$pred
plot(meuse.grid_sf[,c("z1","z2")], pch = 15,
     cex = 0.5, nbreaks = 20,axes=TRUE) # Predictive means

### The same fit, explored interactively over a basemap
# spCFmap(mod, crs = 28992)   # crs = the system the coordinates are in
```

---

cf_glm_hv *Holdout validation for coarse-to-fine spatial generalized linear mixed models (CF-GLMMs)*

Description

Trains CF-GLMMs and selects the number of spatial scales through sequential holdout validation.

Usage

```
cf_glm_hv(
  y,
  x = NULL,
  coords,
  offset = NULL,
  train_rat = 0.75,
  id_train = NULL,
  alpha = 0.9,
  kernel = "exp",
  family = gaussian(),
  seed = 1234
)
```

Arguments

y	Vector of response variables (N x 1) including continuous, count, and binary responses, following an exponential family distribution.
x	Matrix of covariates (N x K).
coords	Matrix of 2-dimensional point coordinates (N x 2).
offset	Optional. Vector of offset variables (N x 1) included in the linear predictor, consistent with glm.
train_rat	Training sample ratio (default: 0.75). For small to moderate samples (N <= 30000), samples closest to the k-means centers are used for validation samples to stabilize training. For larger samples, training samples are drawn at random.
id_train	Optional. ID indicating training samples. If specified, the corresponding samples are used as training samples. Otherwise, training samples are chosen based on 'train_rat'.
alpha	Decay ratio of the kernel bandwidth in the coarse-to-fine training (default: 0.9). Values closer to one make the optimization more stringent but increase computation time.
kernel	Kernel type for modeling spatial dependence. '"exp"' for the exponential kernel (default) and '"gau"' for the Gaussian kernel.
family	Error distribution and link function specification, consistent with the 'family' argument of glm.
seed	Random seed used for the training/validation split when 'id_train' is not supplied. Default is '1234'. Set to 'NULL' to allow a different split at each call (useful for assessing split sensitivity).

Value

A list with the following elements:

loss_hv Final deviance loss for validation samples.

loss_hv_all Deviance losses obtained at each learning step.

id_train ID of training samples.

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Comber, A., Yoshida, T., Tsutsumida, N., Brunsdon, C., & Nakaya, T. (2025). Coarse-to-fine spatial GLMMs for scalable prediction and multiscale analysis. *ArXiv preprint*, 2605.01157. https://doi.org/10.48550/arXiv.2605.01157

See Also

cf_glm

---

cf_lm *Coarse-to-fine spatial modeling (CFSM) for Gaussian response*

Description

Scalable prediction, regression, and multiscale analysis via Gaussian CFSM.

Usage

```
cf_lm(
  y,
  x = NULL,
  coords,
  x0 = NULL,
  coords0 = NULL,
  mod_hv,
  robust_se = TRUE,
  se_type = c("prediction", "mean"),
  se_method = c("opt", "classic")
)
```

Arguments

y	Vector of response variables (N x 1).
x	Matrix of covariates (N x K).
coords	Matrix of 2-dimensional point coordinates (N x 2).
x0	Optional. Matrix of covariates at prediction sites (N0 x K).
coords0	Optional. Matrix of 2-dimensional point coordinates at prediction sites (N0 x 2).
mod_hv	Output object of the cf_lm_hv function.
robust_se	If TRUE (default), coefficient standard errors and predictive uncertainty are computed using a cluster-robust sandwich estimator accounting for local spatial correlation. Set FALSE to use naive SEs (not recommended).
se_type	Type of predictive uncertainty in pred/pred_q. "prediction" (default) returns the holdout-calibrated OBSERVATION predictive (mean uncertainty + residual variance, split-conformal SD scaling on the cf_lm_hv validation samples); the signal versions are kept in pred_signal/pred_q_signal. "mean" returns the signal (mean) uncertainty only (previous behaviour).
se_method	Cluster-robust coefficient-SE estimator (used when robust_se = TRUE). "opt" (default) splits the sandwich meat into a field-removed observation-noise part and a field part that adds the calibrated field variance back with a within-block exp(-d/h) correlation (h = median committed bandwidth); this is near-nominal. A refit-free leverage leave-one-out ceiling then caps the field term, preventing over-coverage for count (Poisson) responses while leaving already-calibrated families unchanged. "classic" keeps the realised field inside the working residual (the previous behaviour), which is valid but conservative.

Value

A list with the following elements:

beta Regression coefficients, their standard errors, and the lower and upper limits of the 95 percent confidence intervals.

sd_summary Standard deviation of the regression term (xb), spatial processes (spatial_scale1, spatial_scale2,...), additional learned components (effective if 'cf_lm_hv/add_learn' is not 'none'), and residuals.

e_summary Holdout validation accuracy evaluated on the validation samples: R-squared (validation_R2), root mean squared error (validation_RMSE), and mean absolute error (validation_MAE).

pred Predictive means and standard deviations (sample sites). When no additional learner is active, the spatial-process contribution to the predictive SD is rescaled by a holdout-calibrated factor (stored as other$tau) estimated on the validation samples.

pred0 Predictive means and standard deviations (prediction sites).

pred_q Predictive quantiles at the sample sites (data.frame with columns q0.005, q0.025, ..., q0.975, q0.995). With add_learn = "rf"/"lightgbm" active, the combined predictive distribution is calibrated by total conformalized quantile regression (CQR) on the validation samples; otherwise the quantiles are Gaussian about the predictive mean using the (tau-calibrated) pred_sd. pred_sd is a Gaussian-equivalent summary of these quantiles.

pred0_q Predictive quantiles at the prediction sites; identical column structure to `pred_q`. `NULL` when prediction sites are not supplied.

bands Bandwidth values for each scale. The i-th bandwidth corresponding to the i-th column of the Z matrix.

Z Predictive means of the single-scale processes at each scale, corresponding to each bandwidth value (sample sites; list).

Z_sd Predictive standard deviation of the spatial processes at each scale (sample sites; list).

Z0 Predictive mean of the spatial process at each scale (prediction sites; list).

Z0_sd Predictive standard deviation of the spatial process at each bandwidth (prediction sites; list).

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Comber, A., Yoshida, T., Tsutsumida, N., Brunsdon, C., & Nakaya, T. (2026). Coarse-to-fine spatial modeling: A scalable, machine-learning-compatible framework. *Geographical Analysis*, 58(2), e70034. https://onlinelibrary.wiley.com/doi/10.1111/gean.70034

See Also

`cf_glm`, `cf_lm_hv`, `sp_scalewise`

Examples

```
set.seed(123)
require(sp); require(sf)
data(meuse)
data(meuse.grid)

### Data
y        <- log(meuse[,"zinc"])
coords   <- meuse[,c("x","y")]
x        <- data.frame(dist   = meuse[,"dist"],
                       ffreq2 = as.integer(meuse$ffreq == 2),
                       ffreq3 = as.integer(meuse$ffreq == 3))

### Data at prediction sites
coords0  <- meuse.grid[,c("x","y")]
x0       <- data.frame(dist   = meuse.grid[,"dist"],
                       ffreq2 = as.integer(meuse.grid$ffreq == 2),
                       ffreq3 = as.integer(meuse.grid$ffreq == 3))

### Holdout validation optimizing the number of spatial scales
mod_hv   <- cf_lm_hv(y = y, x = x, coords = coords, add_learn = "none")

### Spatial modeling and prediction
```

```
mod      <- cf_lm(y = y, x = x, x0 = x0, coords = coords, coords0 = coords0,
                  mod_hv = mod_hv)
mod

### Mapping predictive mean and standard deviations (SD)
meuse.grid$pred   <- mod$pred0$pred
meuse.grid$pred_sd<- mod$pred0$pred_sd
meuse.grid_sf     <- st_as_sf(meuse.grid, coords = c("x","y"))
plot(meuse.grid_sf[,"pred"], pch = 15, cex = 0.5, nbreaks = 20)   # Predictive mean
plot(meuse.grid_sf[,"pred_sd"], pch = 15, cex = 0.5, nbreaks = 20)# Predictive SD

### Multiscale spatial pattern/feature extraction
mod_s1<- sp_scalewise(mod,bw_range=c(1000,Inf)) # Large scale (1000 <= bandwidth)
mod_s2<- sp_scalewise(mod,bw_range=c(500,1000)) # Middle scale (500 <= bandwidth <= 1000)
mod_s3<- sp_scalewise(mod,bw_range=c(0,500))    # Small scale (bandwidth <= 500)
z1    <- mod_s1$pred0$pred                      # Predictive mean
z2    <- mod_s2$pred0$pred
z3    <- mod_s3$pred0$pred
z1_sd <- mod_s1$pred0$pred_sd                   # Predictive SD
z2_sd <- mod_s2$pred0$pred_sd
z3_sd <- mod_s3$pred0$pred_sd
meuse.grid_sf3  <- cbind(meuse.grid_sf, z1, z2, z3, z1_sd, z2_sd, z3_sd)
plot(meuse.grid_sf3[,c("z1","z2","z3")], pch = 15,
     cex = 0.5, nbreaks = 20,key.pos=4,axes=TRUE) # Predictive means
plot(meuse.grid_sf3[,c("z1_sd","z2_sd","z3_sd")], pch = 15,
     cex = 0.5, nbreaks = 20,key.pos=4,axes=TRUE) # Predictive SD

### The same fit, explored interactively over a basemap
# spCFmap(mod, crs = 28992)   # crs = the system the coordinates are in
```

`cf_lm_hv` *Holdout validation for the Gaussian coarse-to-fine spatial modeling (CFSM)*

Description

Trains the CFSM-based Gaussian spatial regression and selects the number of spatial scales through sequential holdout validation.

Usage

```
cf_lm_hv(
  y,
  x = NULL,
  coords,
  train_rat = 0.75,
  id_train = NULL,
  alpha = 0.9,
```

```
  kernel = "exp",
  add_learn = "none",
  seed = 123
)
```

Arguments

y	Vector of response variables (N x 1).
x	Matrix of covariates (N x K).
coords	Matrix of 2-dimensional point coordinates (N x 2).
train_rat	Training sample ratio (default: 0.75). For small to moderate samples (N <= 30000), samples closest to the k-means centers are used for validation samples to stabilize training. For larger samples, training samples are drawn at random.
id_train	Optional. ID indicating training samples. If specified, the corresponding samples are used as training samples. Otherwise, training samples are chosen based on 'train_rat'.
alpha	Decay ratio of the kernel bandwidth in the coarse-to-fine training (default: 0.9). Values closer to one make the optimization more stringent but increase computation time.
kernel	Kernel type for modeling spatial dependence. '"exp"' for the exponential kernel (default) and '"gau"' for the Gaussian kernel.
add_learn	Additional learner trained on the residuals to capture non-linear patterns and/or higher-order interactions. '"rf"' uses a random forest (**ranger**) and '"lightgbm"' uses LightGBM (**lightgbm**); both are tuned by minimizing validation SSE. For '"lightgbm"', the predictive quantiles are conformalized on the validation split so that their uncertainty is calibrated. Both learners are optional: the corresponding package (**ranger** or **lightgbm**) must be installed, and an informative error is raised if it is not. Default is '"none"', meaning no additional training.
seed	Random seed used for the training/validation split when 'id_train' is not supplied. Default is '123'. Set to 'NULL' to allow a different split at each call (useful for assessing split sensitivity).

Value

A list with the following elements:

sse_hv Final sum-of-squared error (SSE) for validation samples.

sse_hv_all SSEs obtained at each learning step.

id_train ID of training samples.

other Other internally used output objects.

Author(s)

Daisuke Murakami

References

Murakami, D., Comber, A., Yoshida, T., Tsutsumida, N., Brunsdon, C., & Nakaya, T. (2026). Coarse-to-fine spatial modeling: A scalable, machine-learning-compatible framework. *Geographical Analysis*, 58(2), e70034. https://onlinelibrary.wiley.com/doi/10.1111/gean.70034

See Also

cf_lm

---

spCFmap *Interactive mapping for coarse-to-fine spatial modelling*

Description

Opens a Shiny web app for mapping CFSM results over a basemap. The function has two modes:

Usage

```
spCFmap(mod = NULL, crs = NULL, launch = TRUE, ...)
```

Arguments

mod	Optional fitted model returned by cf_lm, cf_glm, cf_dglm or cf_downscale. If omitted, the full data-upload and model-fitting application is launched.
crs	Coordinate reference system of the coordinates that were passed to the model: an EPSG code (e.g. 4326 or "EPSG:4326") or a proj/WKT string, used to place the results on the longitude/latitude basemap. Give 4326 when the model was fitted on raw longitude/latitude, and the code of the projected system (e.g. 28992, 27700, 25832) when it was fitted on projected coordinates; an EPSG code assumes the coordinates are in the unit of that system, so rescaled coordinates (metres divided by 1000, say) need a proj string such as "+proj=utm +zone=32 +datum=WGS84 +units=km". There is deliberately no default: coordinates carry no unit of their own, and guessing would silently place the map in the wrong part of the world, so crs must be supplied whenever mod is given. It is ignored when mod is omitted, since the application asks for the coordinate reference system interactively – there, EPSG:4326 (longitude/latitude) is the default offered for uploaded files, while the bundled demo data sets preselect their own systems.
launch	Logical; if TRUE (default) run the app, otherwise return the Shiny app object without launching.
...	Passed to runApp (e.g. port, host, launch.browser). Unused when launch = FALSE.

Details

`spCFmap()` Without `mod`, the full application is launched: models (`cf_lm`, `cf_glm`, `cf_dglm`, `cf_downscale`) are fitted inside the app from demo data (meuse, a space-time air-quality set, and an areal downscaling set) or from user CSV / GeoJSON uploads, and predictions can be exported as CSV or GeoJSON.

`spCFmap(mod, crs)` With a fitted model, a small app maps that model directly. The layer (predictive mean / SD, covariate effect, or a scale-wise spatial component), colour scaling, and - for space-time or downscaling fits - the time range or bandwidth range are chosen interactively.

Value

If `launch = TRUE`, the value returned by `runApp` (invisibly); otherwise a `shiny.appobj`.

See Also

`cf_lm`, `cf_glm`, `cf_dglm`, `cf_downscale`, `sp_scalewise`

Examples

```
## Not run:
spCFmap()                       # full app, opens in the browser
spCFmap(launch.browser = FALSE) # print the local URL instead

m <- cf_lm(y = y, x = x, x0 = x0, coords = coords, coords0 = coords0,
           mod_hv = cf_lm_hv(y = y, x = x, coords = coords))
spCFmap(m, crs = 28992)         # map an already-fitted model

## End(Not run)
```

---

sp_scalewise *Extract scale-wise spatial processes*

Description

Evaluate mean and standard deviation of the (multiscale) spatial process for bandwidth values within a pre-specified range. For a spatio-temporal fit from `cf_dglm`, the process can additionally be averaged over a user-specified time range, returning the temporally averaged spatial process at each (sample / prediction) location.

Usage

```
sp_scalewise(mod, bw_range = c(0, Inf), time_range = c(-Inf, Inf))
```

Arguments

mod	Output object from the cf_lm, cf_glm or cf_dglm function.
bw_range	Range of bandwidth values of the synthesized spatial processes, treated as the half-open interval [min, max). For example, bw_range = c(10, 20) synthesizes scales with bandwidth b such that 10 <= b < 20. The half-open convention lets contiguous ranges (e.g. c(0, 10) and c(10, Inf)) partition the scales without double-counting a scale whose bandwidth equals the shared endpoint. The default c(0, Inf) synthesizes all scales.
time_range	Range of time points over which the spatio-temporal process is averaged. Only used when mod is a cf_dglm fit (which carries a time index for every row of Z). For example, time_range = c(5, 10) averages the process over time points 5 to 10 at each location. The default c(-Inf, Inf) averages over all time points. Ignored (with a warning if set to a non-default value) for purely spatial fits.

Value

A list with the following elements:

pred Means and standard deviations of the spatial process at the sample sites. For a cf_dglm fit, one row per (unique) location with the temporally averaged process, together with its coordinates and the number of averaged time points.

pred0 The same at the prediction sites. NULL when mod was fitted without prediction sites, and also (with a warning) when no prediction site falls inside time_range.

Author(s)

Daisuke Murakami

See Also

cf_lm, cf_glm, cf_dglm

Index

cf_dglm, 2, *6–8*, *22–24*
cf_dglm_hv, *2–4*, 5
cf_downscale, 8, *12*, *22, 23*
cf_downscale_hv, *8, 9*, 10
cf_glm, *4*, 12, *17*, *19*, *22–24*
cf_glm_hv, *8*, *12*, *14*, 15
cf_lm, *9*, 17, *22–24*
cf_lm_hv, *12*, *18, 19*, 20

glm, *6, 7*, *12*, *16*

runApp, *22, 23*

sp_scalewise, *14*, *19*, *23*, 23
spCFmap, 22