

Package ‘spCF’

October 7, 2026

Type Package

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

Version 0.2.2.1

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, MASS, 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](https://doi.org/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 2026-10-07 12:50:02 UTC

Contents

cf_dglm	2
cf_dglm_hv	6
cf_downscale	9
cf_downscale_hv	11
cf_glm	13
cf_glm_hv	17
cf_lm	19
cf_lm_hv	22
negbin	24
spCF-predict	25
spCFmap	27
sp_scalewise	28
Index	30

cf_dglm	<i>Coarse-to-fine dynamic (space-time) spatial GLMMs (CF-DGLMMs)</i>
---------	--

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 $g(\mu_{i,t}) = x'_{i,t}\beta + \sum_k f_k(s_i, t) + \text{offset}$, 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,
se_type = c("prediction", "mean"),
se_method = c("opt", "classic"),
keep_scales = TRUE
)

```

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 points at prediction sites (N0 x 1). May include time points with no observations. They are predicted, after the fit and without changing it, from the smoothed per-knot AR(1) states: a time point between two training time points is bridged between their states, the step being split in proportion to the time differences; a time point after (before) the training period is forecast (backcast), the number of AR(1) steps being the time difference over the median spacing of the training time points. The same applies to the time-varying coefficients (random walk). The fit, and beta_tv, therefore do not depend on time0, and predict.cf_dglm gives the same predictions for new sites and times later.
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.
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))). Negative binomial (negbin): negative-binomial count predictive with the fitted dispersion. Other families use a moment-matched observation predictive with the holdout-estimated dispersion (Gamma: gamma; inverse.gaussian: inverse Gaussian; quasipoisson: negative binomial; quasibinomial: beta for proportions, as binomial for 0/1 data; otherwise normal), with the mean-uncertainty scale calibrated to 95% holdout coverage. The signal versions

	are kept in <code>pred_signal/pred_q_signal</code> . "mean" returns the signal (mean) uncertainty only (previous behaviour). See <code>other\$calibration</code> .
<code>se_method</code>	Cluster-robust coefficient-SE estimator (used when <code>robust_se = TRUE</code>). "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.
<code>keep_scales</code>	If TRUE (default), the scale-wise processes <code>Z</code> , <code>Z_sd</code> , <code>Z0</code> and <code>Z0_sd</code> are kept in the output. They hold one column per selected scale for every observation (site x time point), which makes them the largest part of the fitted object for large data, and above all for long panels. With FALSE they are dropped (NULL); predictions, standard errors, <code>sd_summary</code> and the maps of <code>spCFmap</code> for the total prediction are unchanged, but <code>sp_scalewise</code> needs them.

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/`.)

The field variance of the mean is assembled scale by scale so that it grows smoothly with the distance to the data and reaches the marginal variance of the fitted total field (the sill, $\text{var}(\text{sum}_r z_r)$ on the link scale) far from it, as a stationary process reverts to its marginal variance. For scale r , $r_r = V_r^d / P_{0,r}$ is the fraction of the scale's prior variance left after the data, from a distance-aware gPoE variance (each knot informs a site through its kernel correlation w , conditional variance $w^2 P + (1 - w^2) P_0$); the calibrated variance is $c_r \tau r_r / (1 + r_r(\tau - 1))$, where the caps c_r are proportional to the variance of each fitted scale and sum to the sill, and τ (`mod_hv$other$tau_stage`) scales the information of the data and is solved from the holdout moment equation in `cf_dglm_hv`. The sill is floored at a direct estimate of the field variance (working residual variance of the GLM minus a nearest-neighbour nugget), and when no scale is accepted that estimate is added as unmodeled field variance. Point predictions and coefficient estimates do not depend on these bounds. For binomial responses the field variance is left uncapped.

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 training 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"`.

As in `cf_glm`, the quantile tables `pred_q`, `pred0_q` and `pred_q_signal` are not stored but computed on access (at the 15 levels 0.005, 0.025, 0.05, 0.1, ..., 0.9, 0.95, 0.975, 0.995; `predict.cf_dglm` gives other levels), and `Z`, `Z_sd`, `Z0`, `Z0_sd` are NULL when `keep_scales = FALSE`. 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**, 2608.03449.

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)

### Prediction sites: a regular 25 km grid covering the convex hull of the
### network (as in the spCF_dglm vignette), at 10 time points every six months
### over the observed period: June 2001 (time = 6), December 2001 (12), ...,
### December 2005 (60)
uni <- unique(as.data.frame(coords))
hull <- st_convex_hull(st_union(st_as_sf(uni, coords = c("X", "Y"))))
gcen <- st_make_grid(hull, cellsize = 25000, what = "centers")
gcen <- gcen[st_intersects(gcen, hull, sparse = FALSE)[, 1]]
gxy <- st_coordinates(gcen)
ng <- nrow(gxy)
tp <- seq(6, 60, by = 6)
month0 <- rep(c(6, 12), length.out = length(tp)) # calendar month (June, December)
coords0 <- do.call(rbind, replicate(length(tp), gxy, simplify = FALSE))
time0 <- rep(tp, each = ng)
x0 <- data.frame(sin12 = sin(2 * pi * rep(month0, each = ng) / 12),
```

```

cos12 = cos(2 * pi * rep(month0, each = ng) / 12))

### Space-time modeling and prediction
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 predictions for June 2005 and December 2005
grid_sf <- st_sf(Jun2005 = mod$pred0$pred[time0 == 54],
                 Dec2005 = mod$pred0$pred[time0 == 60],
                 geometry = gcen, crs = 25832)
plot(grid_sf, pch = 15, cex = 1.5, 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)

### Prediction with predict(): the model can be fitted WITHOUT prediction
### sites and times (no x0, coords0, time0) and used to predict at any sites
### and time points later, without the training data: e.g. the grid in
### December 2005 (as above) and in March 2006 (time = 63, three months beyond
### the data), with a 90 percent prediction interval
mod_f <- cf_dglm(y = air$pm10, x = x, coords = coords, time = air$time,
                 mod_hv = mod_hv) # no x0, coords0, time0
p60 <- predict(mod_f, x0 = x0[time0 == 60, ], coords0 = gxy,
               time0 = rep(60, ng))
all.equal(p60$pred, mod$pred0$pred[time0 == 60]) # same as cf_dglm(..., time0)
p <- predict(mod_f, x0 = data.frame(sin12 = rep(sin(2 * pi * 3 / 12), ng),
                                   cos12 = rep(cos(2 * pi * 3 / 12), ng)),
             coords0 = gxy, time0 = rep(63, ng), probs = c(0.05, 0.95))

head(p)

```

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 $g(\mu_{i,t}) = x'_{i,t}\beta + \sum_k f_k(s_i, t) + \text{offset}$, 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 <i>location</i> 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() . Negative binomial responses: negbin() estimates the dispersion θ (re-estimated on the training samples after each accepted scale); negbin(theta) or <code>MASS::negative.binomial(theta)</code>

	keeps it fixed. <code>poisson(link = "identity")</code> is supported with the mean floored at a small positive value.
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*, 2608.03449.

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**, 2606.29798.

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 correspond 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**, 2606.29798.

See Also[cf_downscale](#), [cf_lm_hv](#)

cf_glm	<i>Coarse-to-fine spatial generalized linear mixed models (CF-GLMMs)</i>
--------	--

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"),
  keep_scales = TRUE
)
```

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 $\text{pred_sd} = \sqrt{p(1-p)}$). Negative binomial (negbin): negative-binomial count predictive with the fitted dispersion. Other families use a moment-matched observation predictive with the holdout-estimated dispersion (Gamma: gamma; inverse.gaussian: inverse Gaussian; quasipoisson: negative binomial; quasibinomial: beta for proportions, as binomial for 0/1 data; otherwise normal), with the mean-uncertainty scale calibrated to 95% holdout coverage. 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.
keep_scales	If TRUE (default), the scale-wise processes Z, Z_sd, Z0 and Z0_sd are kept in the output. They hold one column per selected scale for every sample (and prediction) site, which makes them the largest part of the fitted object for large data. With FALSE they are dropped (NULL); predictions, standard errors, sd_summary and the maps of spCFmap for the total prediction are unchanged, but sp_scalewise needs them.

Details

The link-scale spatial-process predictive variance is bounded stage by stage as in [cf_lm](#): $\min(\tau p v_r, \kappa s_r^2)$ with the stage caps rescaled to sum to the marginal field variance, and the holdout factor τ solving the working-weighted moment equation. For the binomial family the field variance is left uncapped.

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_PseudoR2), 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 (with `se_type = "prediction"`, from the calibrated observation predictive). Not stored in the object: `mod$pred_q` computes it on access, at the 15 levels listed above; `predict.cf_glm` gives them at other levels and at new sites.
- 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). `Z`, `Z_sd`, `Z0` and `Z0_sd` are NULL when `keep_scales = FALSE`.
- 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

##### 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

### Prediction with predict(): the model can be fitted WITHOUT prediction
### sites (no x0, coords0) and used to predict at any sites later; the
### training data are not needed then. For a binary response,
### se_type = "mean" gives the quantiles of the probability (those of a
### single 0/1 observation are degenerate).
mod_f <- cf_glm(y = y, x = x, coords = coords, mod_hv = mod_hv) # no x0, coords0
p <- predict(mod_f, x0 = x0, coords0 = coords0,
             probs = c(0.025, 0.975), se_type = "mean")

head(p)
all.equal(p$pred, mod$pred0_signal$pred) # same as cf_glm(..., coords0 = coords0)

```

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 . Negative binomial responses: negbin() estimates the dispersion θ (re-estimated on the training samples after each accepted scale); negbin(theta) or <code>MASS::negative.binomial(theta)</code> keeps it fixed. <code>poisson(link = "identity")</code> is supported with the mean floored at a small positive value.
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"),
  keep_scales = TRUE
)
```

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. For cf_lm the noise part is rescaled to a nugget (observation-noise variance)

	estimated from nearest-neighbour differences of the fixed-effect residuals, because the in-sample residual is shrunk by the fitted field. 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.
keep_scales	If TRUE (default), the scale-wise processes Z , Z_sd , $Z0$ and $Z0_sd$ are kept in the output. They hold one column per selected scale for every sample (and prediction) site, which makes them the largest part of the fitted object for large data. With FALSE they are dropped (NULL); predictions, standard errors, sd_summary and the maps of spCFmap for the total prediction are unchanged, but sp_scalewise needs them.

Details

The spatial-process predictive variance is bounded stage by stage: the calibrated variance of stage r is $\min(\tau pv_r, \kappa s_r^2)$, where pv_r is the stage's predictive variance (infinite where no knot with data reaches the site), s_r^2 the variance of the stage's fitted field over the sample sites, and κ rescales the caps to sum to the marginal field variance. The holdout factor τ solves the corresponding moment equation. The predictive SD thus grows smoothly with the distance to the data and reaches the marginal field variance far from it.

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). validation_R2 is NA when the holdout predictions are constant (no covariates and no accepted scale).
- 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. Not stored in the object: mod\$pred_q computes it on access, at the 15 levels 0.005, 0.025, 0.05, 0.1, ..., 0.9, 0.95, 0.975, 0.995; [predict.cf_lm](#) gives them at other levels and at new sites.
- 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).
Z, Z_sd, Z0 and Z0_sd are NULL when keep_scales = FALSE.

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

### Prediction with predict(): the model can be fitted WITHOUT prediction
### sites (no x0, coords0) and used to predict at any sites later; the
### training data are not needed then
mod_f <- cf_lm(y = y, x = x, coords = coords, mod_hv = mod_hv) # no x0, coords0
p <- predict(mod_f, x0 = x0, coords0 = coords0, probs = c(0.025, 0.975))
head(p) # pred, pred_sd, q0.025, q0.975
all.equal(p$pred, mod$pred0$pred) # same as cf_lm(..., coords0 = coords0)
head(predict(mod_f, probs = c(0.1, 0.9))) # sample sites, 80 percent interval

```

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](#)

negbin

*Negative binomial family with an estimated (or fixed) dispersion***Description**

A [family](#) object for negative binomial responses, for use as the family argument of [cf_glm_hv](#) and [cf_dglm_hv](#). With `theta = NULL` (default) the dispersion parameter θ (variance $\mu + \mu^2/\theta$) is estimated by maximum likelihood: it is re-estimated on the training samples after the initial GLM and after every accepted spatial scale, so that it reflects the over-dispersion that remains once the multiscale spatial process is modeled. A numeric `theta` keeps the dispersion fixed (equivalent to `MASS::negative.binomial(theta)`, which is also accepted).

Usage

```
negbin(theta = NULL, link = "log")
```

Arguments

<code>theta</code>	NULL (estimate) or a positive number (fixed).
<code>link</code>	Link function: "log" (default), "sqrt" or "identity".

Value

An object of class "family". The fitted value of θ is stored in `family$theta` of the family returned in `mod_hv$other$family` (and `mod$other$family`).

See Also[cf_glm_hv](#), [cf_dglm_hv](#)

Examples

```
set.seed(1)
n      <- 300
coords <- cbind(runif(n), runif(n))
y      <- rnbino(n, size = 2, mu = exp(1 + sin(4 * coords[, 1])))
mod_hv <- cf_glm_hv(y = y, coords = coords, family = negbin())
mod_hv$other$family$theta # estimated dispersion
```

spCF-predict

Prediction from a fitted coarse-to-fine model

Description

Predicts at new sites from a `cf_lm` or `cf_glm` fit. The fitted object keeps, for every selected scale, the local estimates at the knots of that scale; prediction only spreads these to the new sites, so its cost does not depend on the size of the training data, and the training data are not needed. The result is identical to fitting the model with the same sites given as `coords0` (and `x0`, `offset0`).

Usage

```
## S3 method for class 'cf_lm'
predict(object, x0 = NULL, coords0 = NULL, probs = NULL, se_type = NULL, ...)

## S3 method for class 'cf_glm'
predict(
  object,
  x0 = NULL,
  coords0 = NULL,
  offset0 = NULL,
  probs = NULL,
  se_type = NULL,
  ...
)

## S3 method for class 'cf_dglm'
predict(
  object,
  x0 = NULL,
  coords0 = NULL,
  time0 = NULL,
  offset0 = NULL,
  probs = NULL,
  se_type = NULL,
  ...
)
```

Arguments

<code>object</code>	A fitted model from <code>cf_lm</code> or <code>cf_glm</code> .
<code>x0</code>	Covariates at the prediction sites, with the same columns as <code>x</code> in the fit. Required when the model has covariates.
<code>coords0</code>	Coordinates of the prediction sites (matrix or data.frame with two columns). If NULL, the predictions at the sample sites are returned.
<code>probs</code>	Probability levels of the predictive quantiles. Defaults to the levels of <code>pred_q</code> in the fit (0.005, 0.025, 0.05, 0.1, ..., 0.9, 0.95, 0.975, 0.995).
<code>se_type</code>	"prediction" for the predictive distribution of a new observation or "mean" for that of the mean. Defaults to the <code>se_type</code> of the fit; "prediction" needs a fit with <code>se_type = "prediction"</code> .
<code>...</code>	Not used.
<code>offset0</code>	Offset at the prediction sites (<code>cf_glm</code> only; zero if NULL).
<code>time0</code>	Time points of the prediction sites (<code>cf_dglm</code> only), one per row of <code>coords0</code> . They may be training time points, time points between them (bridged between the smoothed states of the neighbouring training times), or time points before or after the training period (AR(1) backcast or forecast, the number of steps being the time difference over the median spacing of the training times).

Details

With an additional learner (`add_learner` in `cf_lm_hv`), the learner's model is kept in the fit and the quantiles of the combined predictive are simulated, as in `cf_lm`; they then vary slightly from call to call.

Value

A data.frame with one row per site: the predictive mean (`pred`), the predictive standard deviation (`pred_sd`) and the predictive quantiles (`q<level>`, e.g. `q0.025`), on the response scale.

Examples

```
set.seed(1)
n      <- 300
coords <- cbind(px = runif(n), py = runif(n))
x      <- data.frame(x1 = rnorm(n))
y      <- 0.5 * x$x1 + sin(4 * coords[, 1]) + rnorm(n, sd = 0.3)
hv     <- cf_lm_hv(y = y, x = x, coords = coords)
mod    <- cf_lm(y = y, x = x, coords = coords, mod_hv = hv)

coords0 <- cbind(px = runif(5), py = runif(5))
x0      <- data.frame(x1 = rnorm(5))
predict(mod, x0 = x0, coords0 = coords0, probs = c(0.025, 0.975))
```

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

<code>mod</code>	Optional fitted model returned by <code>cf_lm</code> , <code>cf_glm</code> , <code>cf_dglm</code> or <code>cf_downscale</code> . If omitted, the full data-upload and model-fitting application is launched.
<code>crs</code>	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 <code>crs</code> must be supplied whenever <code>mod</code> is given. It is ignored when <code>mod</code> 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.
<code>launch</code>	Logical; if TRUE (default) run the app, otherwise return the Shiny app object without launching.
<code>...</code>	Passed to <code>runApp</code> (e.g. port, host, <code>launch.browser</code>). Unused when <code>launch = FALSE</code> .

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, <code>bw_range = c(10, 20)</code> synthesizes scales with bandwidth b such that $10 \leq b < 20$. The half-open convention lets contiguous ranges (e.g. <code>c(0, 10)</code> and <code>c(10, Inf)</code>) partition the scales without double-counting a scale whose bandwidth equals the shared endpoint. The default <code>c(0, Inf)</code> synthesizes all scales.
time_range	Range of time points over which the spatio-temporal process is averaged. Only used when <code>mod</code> is a cf_dglm fit (which carries a time index for every row of Z). For example, <code>time_range = c(5, 10)</code> averages the process over time points 5 to 10 at each location. The default <code>c(-Inf, Inf)</code> 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](#), [27–29](#)
`cf_dglm_hv`, [2–5](#), [6](#), [24](#)
`cf_downscale`, [9](#), [13](#), [27](#), [28](#)
`cf_downscale_hv`, [9](#), [10](#), [11](#)
`cf_glm`, [4](#), [5](#), [13](#), [18](#), [21](#), [25–29](#)
`cf_glm_hv`, [8](#), [13](#), [15](#), [17](#), [24](#)
`cf_lm`, [10](#), [14](#), [19](#), [24–29](#)
`cf_lm_hv`, [13](#), [19](#), [21](#), [22](#), [26](#)

`family`, [24](#)

`glm`, [7](#), [13](#), [18](#)

`negbin`, [3](#), [7](#), [14](#), [18](#), [24](#)

`predict.cf_dglm`, [3](#), [5](#)
`predict.cf_dglm (spCF-predict)`, [25](#)
`predict.cf_glm`, [15](#)
`predict.cf_glm (spCF-predict)`, [25](#)
`predict.cf_lm`, [20](#)
`predict.cf_lm (spCF-predict)`, [25](#)

`runApp`, [27](#)

`sp_scalewise`, [4](#), [14](#), [15](#), [20](#), [21](#), [28](#), [28](#)
`spCF-predict`, [25](#)
`spCFmap`, [4](#), [14](#), [20](#), [27](#)