

Package ‘stgam’

July 23, 2025

Title Spatially and Temporally Varying Coefficient Models Using Generalized Additive Models

Version 1.0.2

Author Lex Comber [aut, cre],
Paul Harris [ctb],
Chris Brunsdon [ctb]

Maintainer Lex Comber <a.comber@leeds.ac.uk>

Description A framework for specifying spatially, temporally and spatially-and-temporally varying coefficient models using Generalized Additive Models with smooths. The smooths are parameterised with location, time and predictor variables. The framework supports the investigation of the presence and nature of any space-time dependencies in the data by evaluating multiple model forms (specifications) using a Generalized Cross-Validation score. The workflow sequence is to: i) Prepare the data by lengthening it to have a single location and time variables for each observation. ii) Evaluate all possible spatial and/or temporal models in which each predictor is specified in different ways. iii) Evaluate each model and pick the best one. iv) Create the final model. v) Calculate the varying coefficient estimates to quantify how the relationships between the target and predictor variables vary over space, time or space-time. vi) Create maps, time series plots etc. For more details see: Comber et al (2023) <doi:10.4230/LIPIcs.GIScience.2023.22>, Comber et al (2024) <doi:10.1080/13658816.2023

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests cols4all, knitr, ggplot2, cowplot, purrr, rmarkdown, sf,
testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

URL <https://github.com/lexcomber/stgam>

BugReports <https://github.com/lexcomber/stgam/issues>

Depends R (>= 4.1.0), mgcv (>= 1.9-1), glue

LazyData true

Imports foreach, doParallel, parallel, dplyr

VignetteBuilder knitr

NeedsCompilation no
Repository CRAN
Date/Publication 2025-06-12 14:00:02 UTC

Contents

calculate_vcs	2
evaluate_models	3
gam_model_rank	4
hp_data	5
lb	6
Index	8

calculate_vcs	<i>Extracts varying coefficient estimates (for SVC, TVC and STVC models).</i>
---------------	---

Description

Extracts varying coefficient estimates (for SVC, TVC and STVC models).

Usage

```
calculate_vcs(input_data, mgcv_model, terms = NULL)
```

Arguments

- input_data the data used to create the GAM model in data.frame, tibble or sf format. This can be the original data used to create the model or another surface with location and time attributes.
- mgcv_model a GAM model with smooths created using the mgcv package
- terms a vector of names starting with "Intercept" plus the names of the covariates used in the GAM model (these are the names of the variables in the input_data used to construct the model).

Value

A data.frame of the input data and the coefficient and standard error estimates for each covariate. It can be used to generate coefficient estimates for specific time slices and over grided surfaces as described in the package vignette.

Examples

```
require(dplyr)
require(doParallel)
# define input data
data("hp_data")
input_data <-
  hp_data |>
  # create Intercept as an addressable term
  mutate(Intercept = 1)
# create a model for example as result of running `evaluate_models`
gam.m = gam(priceper ~ Intercept - 1 + s(X, Y, by = Intercept) +
  s(X, Y, by = pef) + s(X, Y, by = beds), data = input_data)
# calculate the Varying Coefficients
terms = c("Intercept", "pef", "beds")
vcs = calculate_vcs(input_data, gam.m, terms)
vcs |> select(priceper, X, Y, starts_with(c("b_", "se_")), yhat)
```

evaluate_models	<i>Evaluates multiple models with each predictor variable specified in different ways in order to determining model form</i>
-----------------	--

Description

Evaluates multiple models with each predictor variable specified in different ways in order to determining model form

Usage

```
evaluate_models(
  input_data,
  target_var,
  vars,
  coords_x,
  coords_y,
  VC_type = "SVC",
  time_var = NULL,
  ncores = 2
)
```

Arguments

input_data	the data to be used used to create the GAM model in (data.frame or tibble format), containing an Intercept column to allow it be treated as an addressable term in the model.
target_var	the name of the target variable.
vars	a vector of the predictor variable names (without the Intercept).

coords_x	the name of the X, Easting or Longitude variable in input_data.
coords_y	the name of the Y, Northing or Latitude variable in input_data.
VC_type	the type of varying coefficient model: options are "TVC" for temporally varying, "SVC" for spatially varying and "STVC" for space-time .
time_var	the name of the time variable if undertaking STVC model evaluations.
ncores	the number of cores to use in parallelised approaches (default is 2 to overcome CRAN package checks). This can be determined for your computer by running parallel::detectCores()-1. Parallel approaches are only undertaken if the number of models to evaluate is greater than 30.

Value

a data.frame with indices for each predictor variable, a GCV score (gcv) for each model and the associated formula (f), which should be passed to the gam_model_rank function.

Examples

```
require(dplyr)
require(doParallel)
# define input data
data("hp_data")
input_data <-
  hp_data |>
  # create Intercept as an addressable term
  mutate(Intercept = 1)
# evaluate different model forms
svc_mods <-
  evaluate_models(
    input_data = input_data,
    target_var = "priceper",
    vars = c("pef"),
    coords_x = "X",
    coords_y = "Y",
    VC_type = "SVC",
    time_var = NULL,
    ncores = 2
  )
head(svc_mods)
```

gam_model_rank	<i>Ranks models by GCV, giving the model form for each predictor variable.</i>
----------------	--

Description

Ranks models by GCV, giving the model form for each predictor variable.

Usage

```
gam_model_rank(res_tab, n = 10)
```

Arguments

res_tab a data.frame returned from the `evaluate_models()` function.
n the number of ranked models to return.

Value

a tibble of the 'n' best models, ranked by GCV, with the form of each predictor variable where '—' indicates the absence of a predictor, 'Fixed' that a parametric form was specified, 's_S' a spatial smooth, 's_T' a temporal smooth and 't2_ST' a spatio-temporal smooth.

Examples

```
require(dplyr)
require(doParallel)
# define input data
data("hp_data")
input_data <-
  hp_data |>
  # create Intercept as an addressable term
  mutate(Intercept = 1)
# evaluate different model forms
svc_mods <-
  evaluate_models(
    input_data = input_data,
    target_var = "priceper",
    vars = c("pef"),
    coords_x = "X",
    coords_y = "Y",
    VC_type = "SVC",
    time_var = NULL,
    ncores = 2
  )
gam_model_rank(svc_mods)
```

hp_data

London House Price dataset (Terraced, 2018-2024)

Description

A dataset of a sample terraced houses sales in the London area for 2018 to 2024.

Usage

```
hp_data
```

Format

A tibble with 1888 rows and 13 columns.

price The house price in £1000s

priceper The house price per square metre in £s

tfa Total floor area

dot Date of transfer (sale))

yot Year of transfer (sale)

beds Number of bedrooms

type House type - here all T (terraced)

cef Current energy efficiency rating (values from 0-100)

pef Potential energy efficiency rating (values from 0-100)

ageb The age band of the house construction

lad The local authority district code of the property location

X Easting in metres derived from the geometric centroid (in OSGB projection - EPSG 27700) of the postcode of the sale

Y Northing in metres derived from the geometric centroid (in OSGB projection - EPSG 27700) of the postcode of the sale

Source

Chi, Bin, Dennett, Adam, Oléron-Evans, Thomas and Robin Morphet. 2025. House Price per Square Metre in England and Wales (<https://data.london.gov.uk/dataset/house-price-per-square-metre-in-england-and-wales>)

Examples

```
data("hp_data")
```

 lb

London borough boundaries

Description

A spatial dataset of the boundaries of the 33 London Boroughs extracted from the GWModel package, cleaned and converted to sf.

Usage

lb

Format

A sf polygon (MULTIPOLYGON) dataset with 33 observations and 2 fields.

name The name of the London borough

lad The ONS local authority district code for the borough

Source

Lu, Binbin, Harris, Paul, Charlton, Martin, Brunsdon, Chris, Nakaya, Tomoki, Murakami, Daisuke, Hu, Yigong, Evans, Fiona H, Høglund, Hjalmar. 2024. Geographically-Weighted Models

Examples

```
data("lb")
```

Index

* **datasets**

hp_data, [5](#)

lb, [6](#)

calculate_vcs, [2](#)

evaluate_models, [3](#)

gam_model_rank, [4](#)

hp_data, [5](#)

lb, [6](#)