# Package: BSTFA (via r-universe)

November 1, 2026

**Title** Bayesian Spatio-Temporal Factor Analysis Model

**Description** Implements Bayesian spatio-temporal factor analysis models for multivariate data observed across space and time. The package provides tools for model fitting via Markov chain Monte Carlo (MCMC), spatial and temporal interpolation, and visualization of latent factors and loadings to support inference and exploration of underlying spatio-temporal patterns. Designed for use in environmental, ecological, or public health applications, with support for posterior prediction and uncertainty quantification. Includes functions such as BSTFA() for model fitting and plot_factor() to visualize the latent processes. Functions are based on and extended from methods described in Berrett, et al. (2020) <doi:10.1002/env.2609>.

**Version** 0.1.2

**License** GPL (>= 2)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**VignetteBuilder** knitr

**Depends** R (>= 3.5)

**LinkingTo** Rcpp, RcppArmadillo

**LazyData** true

**LazyDataCompression** xz

**Imports** MASS, RColorBrewer, ggplot2, ggpubr, mgcv, MCMCpack, coda, npreg, matrixcalc, scatterplot3d, sf, Rcpp, lubridate, Matrix, stats, methods, RcppArmadillo, geosphere

**Suggests** knitr, rmarkdown, devtools, kableExtra, bookdown, magick, maps, loo

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

**Repository** https://cberrettstat.r-universe.dev

**Date/Publication** 2026-10-02 21:14:48 UTC

**RemoteUrl** https://github.com/cberrettstat/bstfa

**RemoteRef** HEAD

**RemoteSha** 7b555a4d0af484a87d05db71ae741457e4c12c9f

# Contents

bisquare1d . . . 2
bisquare2d . . . 3
BSTFA . . . 4
BSTFAfull . . . 9
coef.bstfa . . . 16
compute_summary . . . 16
computeLogLik . . . 17
convergence_diag . . . 18
defineRegion . . . 19
makeNewS . . . 19
makePredS . . . 20
map_spatial_param . . . 21
out.sim . . . 22
plot.bstfa . . . 23
plot_annual . . . 24
plot_factor . . . 25
plot_fourier_bases . . . 26
plot_location . . . 27
plot_spatial_param . . . 28
plot_trace . . . 29
predictBSTFA . . . 30
print.bstfa . . . 31
summary.bstfa . . . 32
utahDataList . . . 33

**Index** **34**

---

`bisquare1d` *Bisquare bases for 1-dimensional space*

---

## Description

Evaluate bisquare bases for 1-dimensional space. Used internally within `makeNewS`.

## Usage

```
bisquare1d(locs, knots)
```

### Arguments

| | |
|---|---|
| locs | Matrix of 1-dimensional coordinates for locations of interest. |
| knots | A vector of 1-dimensional knot coordinates for a given resolution. |

### Value

A matrix containing the bisquare bases at a given resolution evaluated at the 1-dimensional input locations.

### Author(s)

Candace Berrett and Adam Simpson

---

## bisquare2d *Bisquare bases for 2-dimensional space*

---

### Description

Function to evaluate bisquare bases for 2-dimensional space. Used internally within makeNewS.

### Usage

```
bisquare2d(locs, knots, latlon = FALSE)
```

### Arguments

| | |
|---|---|
| locs | Matrix of 2-dimensional coordinates for locations of interest. |
| knots | A matrix of 2-dimensional knot coordinates for a given resolution. |
| latlon | Logical scalar indicating whether to use euclidean distance (FALSE; default) or great circle distance (TRUE).#' @importFrom stats dist |

### Value

A matrix containing the bisquare bases for a given resolution evaluated at the input locations.

### Author(s)

Candace Berrett and Adam Simpson

BSTFA *Reduced BSTFA function*

## Description

This function uses MCMC to draw from posterior distributions of a Bayesian spatio-temporal factor analysis model. All spatial processes use one of Fourier, thin plate spline, or multiresolution basis functions. The temporally-dependent factors use Fourier bases. The default values are chosen to work well for many data sets. Thus, it is possible to use this function using only three arguments: `ymat`, `dates`, and `coords`. The default number of MCMC iterations is 10000 (saving 5000); however, depending on the number of observations and processes modeled, it may need more draws than this to ensure the posterior draws are representative of the entire posterior distribution space.

## Usage

```
BSTFA(
  ymat,
  dates,
  coords,
  iters = 10000,
  n.times = nrow(ymat),
  n.locs = ncol(ymat),
  x = NULL,
  mean = FALSE,
  linear = TRUE,
  seasonal = TRUE,
  factors = TRUE,
  n.seasn.knots = min(7, ceiling(length(unique(yday(dates)))/3)),
  spatial.style = "eigen",
  n.spatial.bases = min(10, ceiling(n.locs/3)),
  knot.levels = 2,
  max.knot.dist = mean(dist(coords)),
  premade.knots = NULL,
  plot.knots = FALSE,
  n.factors = min(4, ceiling(n.locs/20)),
  factors.fixed = NULL,
  plot.factors = FALSE,
  load.style = "eigen",
  n.load.bases = 4,
  freq.lon = 2 * diff(range(coords[, 1])),
  freq.lat = 2 * diff(range(coords[, 2])),
  n.temp.bases = ifelse(floor(n.times * 0.1)%%2 == 1, floor(n.times * 0.1) - 1,
    floor(n.times * 0.1)),
  freq.temp = n.times,
  alpha.prec = 1/1e+05,
  tau2.gamma = 2,
  tau2.phi = 1e-07,
```

```
  sig2.gamma = 2,
  sig2.phi = 1e-05,
  sig2 = NULL,
  beta = NULL,
  xi = NULL,
  Fmat = matrix(0, nrow = n.times, ncol = n.factors),
  Lambda = matrix(0, nrow = n.locs, n.factors),
  thin = 1,
  burn = floor(iters * 0.5),
  latlon = FALSE,
  verbose = TRUE,
  save.missing = TRUE,
  save.time = FALSE,
  marginalize = FALSE
)
```

## Arguments

| | |
|---|---|
| ymat | Data matrix of size n.times by n.locs. Any missing data should be marked by NA. The model works best if the data are zero-centered for each location. |
| dates | n.times length vector of class 'Date' corresponding to each date of the observed data. For now, the dates should be regularly spaced (e.g., daily). |
| coords | n.locs by 2 matrix or data frame of coordinates for the locations of the observed data. If using longitude and latitude, longitude is assumed to be the first coordinate. |
| iters | Number of MCMC iterations to draw. Default value is 10000. Function only saves (iters-burn)/thin drawn values. |
| n.times | Number of observations for each location. Default is nrow(ymat). |
| n.locs | Number of observed locations. Default is ncol(ymat). |
| x | Optional n.locs by p matrix of covariates for each location. If there are no covariates, set to NULL (default). |
| mean | Logical scalar. If TRUE, the model will fit a spatially-dependent mean for each location. Otherwise, the model will assume the means are zero at each location (default). |
| linear | Logical scalar. If TRUE (default), the model will fit a spatially-dependent linear increase/decrease (or slope) in time. Otherwise, the model will assume a zero change in slope across time. |
| seasonal | Logical scalar. If TRUE (default), the model will use circular b-splines to model a spatially-dependent annual process. Otherwise, the model will assume there is no seasonal (annual) process. |
| factors | Logical scalar. If TRUE (default), the model will fit a spatio-temporal factor analysis model with temporally-dependent factors and spatially-dependent loadings. |
| n.seasn.knots | Numeric scalar indicating the number of knots to use for the seasonal basis components. The default value is min(7, ceiling(length(unique(yday(dates)))/3)), where 7 will capture approximately 2 peaks during the year. |

`spatial.style` Character scalar indicating the style of bases to use for the linear and seasonal components. Style options are `'fourier'` (default), `'tps'` for thin plate splines, and `'grid'` for multiresolution bisquare bases using knots from a grid across the space.

`n.spatial.bases` Numeric scalar indicating the number of spatial bases to use when `spatial.style` is either `'fourier'` or `'tps'`. Default value is `min(16, ceiling(n.locs/3))`. When `spatial.style` is `'fourier'`, this value must be an even square number.

`knot.levels` Numeric scalar indicating the number of resolutions to use for when `spatial.style='grid'` and/or `load.style='grid'`. Default is 2.

`max.knot.dist` Numeric scalar indicating the maximum distance at which a basis value is greater than zero when `spatial.style='grid'` and/or `load.style='grid'`. Default value is `mean(dist(coords))`.

`premade.knots` Optional list of length `knot.levels` with each list element containing a matrix of longitude-latitude coordinates of the knots to use for each resolution when `spatial.style='grid'` and/or `load.style='grid'`. Otherwise, when `premade.knots = NULL` (default), the knots are determined by using the standard multiresolution grids across the space.

`plot.knots` Logical scalar indicating whether to plot the knots used when `spatial.style='grid'` and/or `load.style='grid'`. Default is `FALSE`.

`n.factors` Numeric scalar indicating how many factors to use in the model. Default is `min(4,ceiling(n.locs/20))`.

`factors.fixed` Numeric vector of length `n.factors` indicating the locations to use for the fixed loadings. This is needed for model identifiability. If `factors.fixed=NULL` (default), the code will select locations with less than 20% missing data and that are far apart in the space.

`plot.factors` Logical scalar indicating whether to plot the fixed factor locations. Default is `FALSE`.

`load.style` Character scalar indicating the style of spatial bases to use for the spatially-dependent loadings. Options are `'fourier'` (default) for the Fourier bases, `'tps'` for thin plate splines, and `'grid'` for multiresolution bases. This can be the same as or different than `spatial.style`.

`n.load.bases` Numeric scalar indicating the number of bases to use for the spatially-dependent loadings when `load.style` is either `'fouier'` or `'tps'`. This can be the same as or different than `n.spatial.bases`. Default is 4. When `load.style='fourier'`, this value must be an even square number.

`freq.lon` Numeric scalar used for `spatial.style` or `load.style` equal to `'fourier'` or `'eigen'`. For `'fourier'`, this is the frequency used for the first column of `coords` (assumed to be longitude) for the Fourier bases. For `'eigen'`, this is the range parameter of the exponential spatial correlation matrix used to create the eigenvectors. Default value is `2*diff(range(coords[,1]))`.

`freq.lat` Numeric scalar used for `spatial.style` or `load.style` equal to `'fourier'`. This is the frequency to use for the second column of `coords` (assumed to be latitude) for the Fourier bases. Default value is `2*diff(range(coords[,2]))`.

| | |
|---|---|
| n.temp.bases | Numeric scalar indicating the number of Fourier bases to use for the temporally-dependent factors. The default value is 10% of n.times. |
| freq.temp | Numeric scalar indicating the frequency to use for the Fourier bases of the temporally-dependent factors. The default value is n.times. |
| alpha.prec | Numeric scalar indicating the prior precision for all model process coefficients. Default value is 1/100000. |
| tau2.gamma | Numeric scalar indicating the prior shape for the precision of the model coefficients. Default value is 2. |
| tau2.phi | Numeric scalar indicating the prior rate for the precision of the model coefficients. Default value is 1e-07. |
| sig2.gamma | Numeric scalar indicating the prior shape for the residual precision. Default value is 2. |
| sig2.phi | Numeric scalar indicating the prior rate for the residual precision. Default value is 1e-05. |
| sig2 | Numeric scalar indicating the starting value for the residual variance. If NULL (default), the function will select a reasonable starting value. |
| beta | Numeric vector of length n.locs + p indicating starting values for the slopes. If NULL (default), the function will select reasonable starting values. |
| xi | Numeric vector of length (n.locs + p)*n.seasn.knots indicating starting values for the coefficients of the seasonal component. If NULL (default), the function will select reasonable starting values. |
| Fmat | Numeric matrix of size n.times by n.factors indicating starting values for the factors. Default value is to start all factor values at 0. |
| Lambda | Numeric matrix of size n.locs by n.factors indicating starting values for the loadings. Default value is to start all loadings at 0. |
| thin | Numeric scalar indicating how many MCMC iterations to thin by. Default value is 1, indicating no thinning. |
| burn | Numeric scalar indicating how many MCMC iterations to burn before saving. Default value is one-half of iters. |
| latlon | Logical scalar indicating whether to use euclidean distance (FALSE; default) or great circle distance (TRUE). |
| verbose | Logical scalar indicating whether or not to print the status of the MCMC process. If TRUE (default), the function will print every time an additional 10% of the MCMC process is completed. |
| save.missing | Logical scalar indicating whether or not to save the MCMC draws for the missing observations. If TRUE (default), the function will save an additional MCMC object containing the MCMC draws for each missing observation. Use FALSE to save file space and memory. |
| save.time | Logical scalar indicating whether to save the computation time for each MCMC iteration. Default value is FALSE. When FALSE, the function compute_summary() will not be useful. |
| marginalize | Logical scalar indicating whether to sample hyper-coefficients by marginalizing out the corresponding parameters. Default value is FALSE. Setting to TRUE will be slower but can improve effective sample sizes. |

## Value

A bstfa object containing the following elements (any elements that are the same as in the function input are removed here for brevity):

**mu** An mcmc object of size `draws` by `n.locs` containing posterior draws for the mean of each location. If `mean=FALSE` (default), then it is `NA`.

**alpha.mu** An mcmc object of size `draws` by `n.spatial.bases + p` containing posterior draws for the coefficients modeling the mean process. If `mean=FALSE` (default), then it is `NA`.

**tau2.mu** An mcmc object of size `draws` by `1` containing the posterior draws for the variance of the mean process. If `mean=FALSE` (default), then it is `NA`.

**beta** An mcmc object of size `draws` by `n.locs` containing the posterior draws for the increase/decrease (slope) across time for each location. If `linear=FALSE`, then it is `NA`.

**alpha.beta** An mcmc object of size `draws` by `n.spatial.bases + p` containing posterior draws for the coefficients modeling the slope. If `linear=FALSE`, then it is `NA`.

**tau2.beta** An mcmc object of size `draws` by `1` containing posterior draws of the variance of the slopes. If `linear=FALSE`, then it is `NAL`.

**xi** An mcmc object of size `draws` by `n.seasn.knots*n.locs` containing posterior draws for the coefficients of the seasonal process. If `seasonal=FALSE`, then it is `NA`.

**alpha.xi** An mcmc object of size `draws` by `(n.spatial.bases + p)*n.seasn.knots` containing posterior draws for the coefficients modeling each coefficient of the seasonal process. If `seasonal=FALSE`, then it is `NA`.

**tau2.xi** An mcmc object of size `draws` by `n.seasn.knots` containing posterior draws of the variance of the coefficients of the seasonal process. If `seasonal=FALSE`, then it is `NA`.

**F.tilde** An mcmc object of size `draws` by `n.times*n.factors` containing posterior draws of the residual factors. If `factors=FALSE`, then it is `NA`.

**alphaT** An mcmc object of size `draws` by `n.factors*n.temp.bases` containing posterior draws of the coefficients for the factor temporally-dependent process. If `factors=FALSE`, then it is `NA`.

**Lambda.tilde** An mcmc object of size `draws` by `n.factors*n.locs` containing posterior draws of the loadings for each location. If `factors=FALSE`, then it is `NA`.

**alphaS** An mcmc object of size `draws` by `n.factors*n.load.bases` containing posterior draws of the coefficients for the loadings spatial process. If `factors=FALSE`, then it is `NA`.

**tau2.lambda** An mcmc object of size `draws` by `1` indicating the residual variance of the loadings spatial process. If `factors=FALSE`, then it is `NA`.

**sig2** An mcmc object of size `draws` by `1` containing posterior draws of the residual variance of the data.

**y.missing** If `save.missing=TRUE`, a matrix of size `sum(missing)` by `draws` containing posterior draws of the missing observations. Otherwise, the object is `NA`.

**time.data** A data frame of size `iters` by `6` containing the time it took to sample each parameter for every iteration.

**setup.time** An object containing the time the model setup took.

**model.matrices** A list containing the matrices used for each modeling process. `newS` is the matrix of spatial basis coefficients for the mean, linear, and seasonal process coefficients. `linear.Tsub` is the matrix used to enforce a linear increase/increase (slope) across time. `seasonal.bs.basis` is the matrix containing the circular b-splines of the seasonal process. `confoundingPmat.prime` is the matrix that enforces orthogonality of the factors from the mean, linear, and seasonal processes. `QT` contains the Fourier bases used to model the temporal factors. `QS` contains the bases used to model the spatial loadings.

**factors.fixed** A vector of length `n.factors` giving the location indices of the fixed loadings.

**iters** A scalar returning the number of MCMC iterations.

**y** An `n.times*n.locs` vector of the observations.

**missing** A logical vector indicating whether that element's observation was missing or not.

**doy** A numeric vector of length `n.times` containing the day of year for each element in the original `dates`.

**knots.spatial** For `spatial.style='grid'`, a list of length `knot.levels` containing the coordinates for all knots at each resolution.

**knots.load** For `load.style='grid'`, a list of length `knot.levels` containing the coordinates for all knots at each resolution.

**draws** The number of saved MCMC iterations after removing the burn-in and thinning.

## Author(s)

Adam Simpson and Candace Berrett

## Examples

```
set.seed(240)
out <- BSTFA(ymat=out.sim$ymat[1:30,1:10],
  dates=out.sim$dates[1:30],
  coords=out.sim$coords[1:10,],
  iters=250,
  save.missing=FALSE,
 n.factors=2)
```

---

`BSTFAfull` *Full BSTFA function*

---

## Description

This function uses MCMC to draw from posterior distributions of a Bayesian spatio-temporal factor analysis model. The spatial processes for the mean, linear, and seasonal behavior use one of Fourier, thin plate spline, or multiresolution basis functions. The temporal dependence of the factors is modeled using a vector autoregressive model. The spatially-dependent loadings are modeled using a mean-zero Gaussian process with an exponential covariance structure. The default values are chosen to work well for many data sets. Thus, it is possible to use this function using only three arguments: `ymat`, `dates`, and `coords`. The default number of MCMC iterations is 10000 (saving 5000); however, depending on the number of observations and processes modeled, it may need more draws than this to ensure the posterior draws are representative of the entire posterior distribution space.

## Usage

```
BSTFAfull(
  ymat,
  dates,
  coords,
  iters = 10000,
  n.times = nrow(ymat),
  n.locs = ncol(ymat),
  x = NULL,
  mean = FALSE,
  linear = TRUE,
  seasonal = TRUE,
  factors = TRUE,
  n.seasn.knots = min(7, ceiling(length(unique(yday(dates)))/3)),
  spatial.style = "grid",
  n.spatial.bases = ceiling(n.locs/2),
  knot.levels = 2,
  max.knot.dist = mean(dist(coords)),
  premade.knots = NULL,
  plot.knots = FALSE,
  freq.lon = diff(range(coords[, 1])),
  freq.lat = diff(range(coords[, 2])),
  n.factors = min(4, ceiling(n.locs/20)),
  factors.fixed = NULL,
  plot.factors = FALSE,
  alpha.prec = 1/1e+05,
  tau2.gamma = 2,
  tau2.phi = 1e-07,
  sig2.gamma = 2,
  sig2.phi = 1e-05,
  omega.ii.mean = 1,
  omega.ii.var = 1,
  omega.ij.mean = 0,
  omega.ij.var = 2,
  S.F = diag(1, n.factors),
  nu.F = n.factors,
  phi.gamma = 3,
  phi.phi = 0.5,
  sig2 = NULL,
  beta = NULL,
  xi = NULL,
  Fmat = matrix(0, nrow = n.times, ncol = n.factors),
  Omega = diag(1, n.factors),
  Sigma.F = diag(1, n.factors),
  Lambda = matrix(0, nrow = n.locs, n.factors),
  phi.lambda = rep(1, n.factors),
  thin = 1,
  burn = floor(iters * 0.5),
```

```
  c.omega = matrix(0.001, n.factors, n.factors),
  c.phi.lambda = rep(0.001, n.factors),
  adapt.iter = (burn + 10),
  adapt.epsilon = 1e-20,
  latlon = FALSE,
  verbose = TRUE,
  save.missing = TRUE,
  save.time = FALSE
)
```

## Arguments

| | |
|---|---|
| ymat | Data matrix of size n.times by n.locs. Any missing data should be marked by NA. The model works best if the data are zero-centered for each location. |
| dates | n.times length vector of class 'Date' corresponding to each date of the observed data. For now, the dates should be regularly spaced (e.g., daily). |
| coords | n.locs by 2 matrix or data frame of coordinates for the locations of the observed data. If using longitude and latitude, longitude is assumed to be the first coordinate. |
| iters | Number of MCMC iterations to draw. Default value is 10000. Function only saves (iters-burn)/thin drawn values. |
| n.times | Number of observations for each location. Default is nrow(ymat). |
| n.locs | Number of observed locations. Default is ncol(ymat). |
| x | Optional n.locs by p matrix of covariates for each location. If there are no covariates, set to NULL (default). |
| mean | Logical scalar. If TRUE, the model will fit a spatially-dependent mean for each location. Otherwise, the model will assume the means are zero at each location (default). |
| linear | Logical scalar. If TRUE (default), the model will fit a spatially-dependent linear increase/decrease (or slope) in time. Otherwise, the model will assume a zero change in slope across time. |
| seasonal | Logical scalar. If TRUE (default), the model will use circular b-splines to model a spatially-dependent annual process. Otherwise, the model will assume there is no seasonal (annual) process. |
| factors | Logical scalar. If TRUE (default), the model will fit a spatio-temporal factor analysis model with temporally-dependent factors and spatially-dependent loadings. |
| n.seasn.knots | Numeric scalar indicating the number of knots to use for the seasonal basis components. The default value is min(7, ceiling(length(unique(lubridate::yday(dates)))/3)), where 7 will capture approximately 2 peaks during the year. |
| spatial.style | Character scalar indicating the style of bases to use for the mean, linear, and seasonal components. Style options are 'fourier', 'tps' for thin plate splines, and 'grid' (default) for multiresolution bisquare bases using knots from a grid across the space. |
| n.spatial.bases | Numeric scalar indicating the number of spatial bases to use when spatial.style is either 'fourier' or 'tps'. Default value is min(8, ceiling(n.locs/3)). |

knot.levels — Numeric scalar indicating the number of resolutions to use for when spatial.style='grid'. Default is 2.

max.knot.dist — Numeric scalar indicating the maximum distance at which a basis value is greater than zero when spatial.style='grid'. Default value is mean(dist(coords)).

premade.knots — Optional list of length knot.levels with each list element containing a matrix of longitude-latitude coordinates of the knots to use for each resolution when spatial.style='grid'. Otherwise, when premade.knots = NULL (default), the knots are determined by using the standard multiresolution grids across the space.

plot.knots — Logical scalar indicating whether to plot the knots used when spatial.style='grid'. Default is FALSE.

freq.lon — Numeric scalar indicating the frequency to use for the first column of coords (assumed to be longitude) for the Fourier bases when spatial.style='fourier'. Default value is diff(range(coords[,1])).

freq.lat — Numeric scalar indicating the frequency to use for the second column of coords (assumed to be latitude) for the Fourier bases when spatial.style='fourier'. Default value is diff(range(coords[,2])).

n.factors — Numeric scalar indicating how many factors to use in the model. Default is min(4,ceiling(n.locs/20)).

factors.fixed — Numeric vector of length n.factors indicating the locations to use for the fixed loadings. This is needed for model identifiability. If factors.fixed=NULL (default), the code will select locations with less than 20% missing data and that are far apart in the space.

plot.factors — Logical scalar indicating whether to plot the fixed factor locations. Default is FALSE.

alpha.prec — Numeric scalar indicating the prior precision for all model process coefficients. Default value is 1/100000.

tau2.gamma — Numeric scalar indicating the prior shape for the precision of the model coefficients. Default value is 2.

tau2.phi — Numeric scalar indicating the prior rate for the precision of the model coefficients. Default value is 1e-07.

sig2.gamma — Numeric scalar indicating the prior shape for the residual precision. Default value is 2.

sig2.phi — Numeric scalar indicating the prior rate for the residual precision. Default value is 1e-05.

omega.ii.mean — Numeric scalar indicating the prior mean for the diagonal elements of the autoregressive correlation matrix of the factors. Default is 1.

omega.ii.var — Numeric scalar indicating the prior variance for the diagonal elements of the autoregressive correlation matrix of the factors. Default is 1.

omega.ij.mean — Numeric scalar indicating the prior mean for the off-diagonal elements of the autoregressive correlation matrix of the factors. Default is 0.

omega.ij.var — Numeric scalar indicating the prior variance for the off-diagonal elements of the autoregressive correlation matrix of the factors. Default is 2.

| | |
|---|---|
| S.F | Numeric matrix of size n.factors by n.factors indicating the prior residual covariance matrix for the factors. Default is diag(1,n.factors). |
| nu.F | Numeric scalar indicating the prior degrees of freedom for the residual covariance matrix of the factors. Default is n.factors; must be greater than or equal to n.factors. |
| phi.gamma | Numeric scalar indicating the prior shape of the spatial range parameter for the spatially-dependent loadings. Default value is 3. |
| phi.phi | Numeric scalar indicating the prior rate of the spatial range parameter for the spatially-dependent loadings. Default is 0.5. |
| sig2 | Numeric scalar indicating the starting value for the residual variance. If NULL (default), the function will select a reasonable starting value. |
| beta | Numeric vector of length n.locs + p indicating starting values for the slopes. If NULL (default), the function will select reasonable starting values. |
| xi | Numeric vector of length (n.locs + p)*n.seasn.knots indicating starting values for the coefficients of the seasonal component. If NULL (default), the function will select reasonable starting values. |
| Fmat | Numeric matrix of size n.times by n.factors indicating starting values for the factors. Default value is to start all factor values at 0. |
| Omega | Numeric matrix of size n.factors by n.factors indicating the starting value for the autoregressive correlation of the factors. Default value is the identity matrix. |
| Sigma.F | Numeric matrix of size n.factors by n.factors indicating the starting value for the residual covariance matrix of the factors. Default value is the identity matrix. |
| Lambda | Numeric matrix of size n.locs by n.factors indicating starting values for the loadings. Default value is to start all loadings at 0. |
| phi.lambda | Numeric vector of length n.factors indicating the starting values for the spatial range parameters for each loading. Default value is a vector of 1's. |
| thin | Numeric scalar indicating how many MCMC iterations to thin by. Default value is 1, indicating no thinning. |
| burn | Numeric scalar indicating how many MCMC iterations to burn before saving. Default value is one-half of iters. |
| c.omega | Numeric matrix of starting values for the proposal standard deviations (for the Metropolis random walk algorithm) for sampling proposal values of the autoregressive correlation matrix for the factors. Default is matrix(0.001, n.factors, n.factors). |
| c.phi.lambda | Numeric vector of starting values for the proposal standard deviations (for the Metropolis random walk algorithm) for sampling proposal values of the range of the spatially-dependent loadings. Default is rep(0.001, n.factors). |
| adapt.iter | Numeric scalar indicating the number of iterations to start adjusting the proposal standard deviations for the Metropolis random walk algorithms. Value must be at least 2 larger than burn. Default value is burn+10. |

| | |
|---|---|
| adapt.epsilon | Numeric scalar indicating the small value to add to the proposal standard deviations when using the adaptive Metropolis random walk algorithms. Default is 1e-20. |
| latlon | Logical scalar indicating whether to use euclidean distance (FALSE; default) or great circle distance (TRUE). |
| verbose | Logical scalar indicating whether or not to print the status of the MCMC process. If TRUE (default), the function will print every time an additional 10% of the MCMC process is completed. |
| save.missing | Logical scalar indicating whether or not to save the MCMC draws for the missing observations. If TRUE (default), the function will save an additional MCMC object containing the MCMC draws for each missing observation. Use FALSE to save file space and memory. |
| save.time | Logical scalar indicating whether to save the computation time for each MCMC iteration. Default value is FALSE. When FALSE, the function compute_summary() will not be useful. |

## Value

A list containing the following elements (any elements that are the same as in the function input are removed here for brevity):

**mu** An mcmc object of size draws by n.locs containing posterior draws for the mean of each location. If mean=FALSE (default), the object is NA.

**alpha.mu** An mcmc object of size draws by n.spatial.bases + p containing posterior draws for the coefficients modeling the mean process. If mean=FALSE (default), the object is NA.

**tau2.mu** An mcmc object of size draws by 1 containing the posterior draws for the variance of the mean process. If mean=FALSE (default), the object is NA.

**beta** An mcmc object of size draws by n.locs containing the posterior draws for the increase/decrease (slope) across time for each location.

**alpha.beta** An mcmc object of size draws by n.spatial.bases + p containing posterior draws for the coefficients modeling the slope.

**tau2.beta** An mcmc object of size draws by 1 containing posterior draws of the variance of the slopes.

**xi** An mcmc object of size draws by n.seasn.knots*n.locs containing posterior draws for the coefficients of the seasonal process.

**alpha.xi** An mcmc object of size draws by (n.spatial.bases + p)*n.seasn.knots containing posterior draws for the coefficients modeling each coefficient of the seasonal process.

**tau2.xi** An mcmc object of size draws by 1 containing posterior draws of the variance of the coefficients of the seasonal process.

**F.tilde** An mcmc object of size draws by n.times*n.factors containing posterior draws of the residual factors.

**alphaT** An mcmc object of size draws by n.factors*n.temp.bases containing posterior draws of the coefficients for the factor temporally-dependent process.

**Lambda.tilde** An mcmc object of size draws by n.factors*n.locs containing posterior draws of the loadings for each location.

**alphaS** An mcmc object of size `draws` by `n.factors*n.load.bases` containing posterior draws of the coefficients for the loadings spatial process.

**tau2.lambda** An mcmc object of size `draws` by `1` indicating the residual variance of the loadings spatial process.

**sig2** An mcmc object of size `draws` by `1` containing posterior draws of the residual variance of the data.

**y.missing** If `save.missing=TRUE`, a matrix of size `sum(missing)` by `draws` containing posterior draws of the missing observations. Otherwise, the object is `NULL`.

**time.data** A data frame of size `iters` by `6` containing the time it took to sample each parameter for every iteration.

**setup.time** An object containing the time the model setup took.

**model.matrices** A list containing the matrices used for each modeling process. `newS` is the matrix of spatial basis coefficients for the mean, linear, and seasonal process coefficients. `linear.Tsub` is the matrix used to enforce a linear increase/increase (slope) across time. `seasonal.bs.basis` is the matrix containing the circular b-splines of the seasonal process. `confoundingPmat.prime` is the matrix that enforces orthogonality of the factors from the mean, linear, and seasonal processes. `QT` contains the fourier bases used to model the temporal factors. `QS` contains the bases used to model the spatial loadings.

**factors.fixed** A vector of length `n.factors` giving the location indices of the fixed loadings.

**iters** A scalar returning the number of MCMC iterations.

**y** An `n.times*n.locs` vector of the observations.

**missing** A logical vector indicating whether that element's observation was missing or not.

**doy** A numeric vector of length `n.times` containing the day of year for each element in the original `dates`.

**knots.spatial** For `spatial.style='grid'`, a list of length `knot.levels` containing the coordinates for all knots at each resolution.

**draws** The number of saved MCMC iterations after removing the burn-in and thinning.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
set.seed(240)
out <- BSTFAfull(ymat=out.sim$ymat[1:30,1:10],
dates=out.sim$dates[1:30],
coords=out.sim$coords[1:10,],
iters=100,
spatial.style="eigen",
save.missing=FALSE,
n.factors=2)
```

## coef.bstfa *Coefficients for 'bstfa' Object*

### Description

This function provides the posterior mean values for main-effects coefficients.

### Usage

```
## S3 method for class 'bstfa'
coef(object, ...)
```

### Arguments

| | |
|---|---|
| object | A bstfa object. |
| ... | Additional arguments passed to or from other methods. |

### Value

A numeric vector for BSTFA model coefficients. Names of the values are labeled using the station name then the parameter.

### Author(s)

Candace Berrett and Adam Simpson

### Examples

```
coef(out.sim)
```

## compute_summary *Print computation summary*

### Description

Print computation summary

### Usage

```
compute_summary(out)
```

### Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |

## Value

Prints the computation time per iteration for each parameter.

## Author(s)

Adam Simpson

## Examples

```
data(utahDataList)
attach(utahDataList)
low.miss <- which(apply(is.na(TemperatureVals), 2, mean)<.02)
out <- BSTFA(ymat=TemperatureVals[1:50,low.miss],
  dates=Dates[1:50],
  coords=Coords[low.miss,],
  n.factors=2,
  iters=10,
  save.time=TRUE)
compute_summary(out)
```

---

# computeLogLik *Compute log-likelihood*

---

## Description

Compute log-likelihood

## Usage

```
computeLogLik(out, verbose = FALSE, addthin = 1)
```

## Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |
| verbose | Logical scalar indicating whether to print status of the log-likelihood computation. Default is FALSE. |
| addthin | Numeric scalar indicating the number of additional draws to thin by to reduce the computation time. Default is 1 (no additional thinning). |

## Value

A matrix of size n.times*n.locs by draws log-likelihood values for each observation and each posterior draw.

## Author(s)

Adam Simpson and Candace Berrett

## Examples

```
data(out.sim)
attach(out.sim)
loglik <- computeLogLik(out.sim, addthin=2)
```

---

`convergence_diag` *Check effective sample size and geweke diagnostic*

---

## Description

Check effective sample size and geweke diagnostic

## Usage

```
convergence_diag(
  out,
  type = "eSS",
  onlyfail = TRUE,
  cutoff = ifelse(type == "eSS", 100, 1.96)
)
```

## Arguments

| | |
|---|---|
| `out` | Output from BSTFA or BSTFAfull. |
| `type` | Character specifying which diagnostic to compute. Options are `ess` and `geweke`. |
| `onlyfail` | Logical scalar indicating whether to save only those parameters that fail to meet the designated `cutoff` value. Default `TRUE`. |
| `cutoff` | Numeric scalar indicating the cutoff value to flag parameters that haven't converged. Used when `onlyfail=TRUE`, ignored otherwise. |

## Value

A list containing the parameters not meeting the convergence cutoff criteria.

## Author(s)

Adam Simpson and Candace Berrett

## Examples

```
data(out.sim)
attach(out.sim)
convergence_diag(out.sim)
```

## defineRegion *Define the initial region of interest.*

### Description

Define the region of interest. Used internally in the makePredS function.

### Usage

```
defineRegion(out, location)
```

### Arguments

| | |
|---|---|
| out | Output object from BSTFA or BSTFAfull functions. |
| location | Matrix of coordinates for the new locations. |

### Value

A list where each element of the list returns a vector of defining the "region" for each prediction location for the given knot resolution.

### Author(s)

Candace Berrett and Adam Simpson

## makeNewS *Create a matrix of bisquare bases for a set of locations.*

### Description

Create a matrix of bisquare bases for a set of locations. Used internally in BSTFA and BSTFAfull.

### Usage

```
makeNewS(
  coords,
  n.locations,
  knot.levels = 2,
  max.knot.dist = mean(dist(coords)),
  x = NULL,
  premade.knots = NULL,
  plot.knots = FALSE,
  regions = FALSE,
  latlon = FALSE
)
```

### Arguments

| | |
|---|---|
| coords | Matrix of coordinates for locations of interest. |
| n.locations | Number of observed locations. |
| knot.levels | Number of levels for the knots. Default is 2. |
| max.knot.dist | Maximum distance between an observation and a knot. Default is mean(dist(coords)). |
| x | Optional matrix of covariates to include in the model. |
| premade.knots | Optional list of length knot.levels each list item containing pre-chosen knots for that level. |
| plot.knots | Logical scalar indicating whether or not to plot the knots. Default is FALSE. |
| regions | Logical scalar indicating if the space should be divided into multiresolution regions. Default is FALSE. |
| latlon | Logical scalar indicating whether to use euclidean distance (FALSE; default) or great circle distance (TRUE). |

### Value

A matrix containing all the multiresolution bisquare bases evaluated at the input coordinates.

### Author(s)

Candace Berrett and Adam Simpson

---

## makePredS *Create a matrix of bisquare bases for new locations.*

---

### Description

Create a matrix of bisquare bases for new locations. Used internally within the predictBSTFA function. Currently only implemented for 2-dimensional coordinates.

### Usage

```
makePredS(out, location)
```

### Arguments

| | |
|---|---|
| out | Output object from BSTFA or BSTFAfull functions. |
| location | Matrix of coordinates for the new locations. |

### Value

A matrix containing the appropriate multiresolution bisquare bases evaluated at the input locations.

### Author(s)

Candace Berrett and Adam Simpson

map_spatial_param *Plot a map of interpolated spatially-dependent parameter values.*

## Description

Plot a map of interpolated spatially-dependent parameter values.

## Usage

```
map_spatial_param(
  out,
  parameter = "slope",
  loadings = 1,
  type = "mean",
  yearscale = TRUE,
  new_x = NULL,
  ci.level = c(0.025, 0.975),
  fine = 100,
 color.gradient = (grDevices::colorRampPalette(rev(RColorBrewer::brewer.pal(9, name =
    "RdBu"))))(fine),
  with.uncertainty = FALSE,
  map = FALSE,
  state = FALSE,
  location = NULL,
  addthin = 1,
  collims = NULL,
  printmap = FALSE
)
```

## Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |
| parameter | One of 'slope' (default), 'loading', or 'mean'. |
| loadings | If parameter='loading', an integer indicating which factor loading to plot. |
| type | One of mean (default), median, ub, or lb indicating which summary statistic to plot at each location. |
| yearscale | If parameter='slope', a logical scalar indicating whether to translate it to a yearly scale (TRUE; default). |
| new_x | If the original model included covariates x, include the same covariates for prediction location. |
| ci.level | If type='lb' or 'ub', the percentiles for the posterior interval. |
| fine | Integer specifying the number of grid points along both the longitude and latitude directions used to interpolate the parameter. The resulting interpolation grid will contain fine*fine total locations. If map=TRUE, state=TRUE, and location is specified, the grid will be clipped to the boundaries of the specified state, removing locations outside of it. |

| | |
|---|---|
| color.gradient | The color palette to use for the plot. Default is colorRampPalette(rev(RColorBrewer::brewer.pal(9 name='RdBu')))(fine). |
| with.uncertainty | Logical scalar indicating whether to include lower and upper credible interval bounds for the parameter. Default is FALSE. |
| map | Logical scalar indicating whether to include a map. Default value is FALSE. If TRUE, location must be specified. |
| state | Logical scalar used when map=TRUE indicating whether the location is a state in the United States (TRUE) or a country (FALSE). |
| location | Name of region to include in the map. Fed to region in the function ggplot2::map_data. |
| addthin | Integer indicating the number of saved draws to thin. Default is to not thin any addthin=1. This can save time when the object is from BSTFAfull and parameter='loading'. |
| collims | Numeric vector of length 2 providing the lower and upper limits for the color scale. If NULL (default), the limits are set to be symmetric around zero based on the maximum absolute value of the parameter being plotted. |
| printmap | Logical scalar indicating whether to plot the resulting image. Default is FALSE. |

## Value

A plot of spatially-dependent parameter values for a grid of interpolated locations.

## Author(s)

Adam Simpson and Candace Berrett

## Examples

```
data(out.sim)
attach(out.sim)
map_spatial_param(out.sim, parameter='slope', fine=25)
```

---

# out.sim *Output of BSTFA evaluated on a simulated data set*

---

## Description

'bstfa' object named out.sim containing the output from running the BSTFA function on a small simulated data set.

## Usage

```
out.sim
```

## Format

See help(BSTFA) for details of what is included as output from the BSTFA function.

## Examples

```
data(out.sim)
dim(out.sim$ymat)
```

---

`plot.bstfa` *Basic Plotting for 'bstfa' Objects*

---

## Description

This function calls appropriate plotting functions for different bstfa parameters. For more control over specific plotting features, use `plot_spatial_param`, `plot_factor`, `plot_annual`, `plot_location`, and `map_spatial_param`.

## Usage

```
## S3 method for class 'bstfa'
plot(
  x,
  ...,
  plot.param = "beta",
  type = "map",
  location = 1,
  loadings = NULL,
  factor = 1
)
```

## Arguments

| | |
|---|---|
| `x` | A bstfa object. |
| `...` | Additional arguments passed to or from other methods. |
| `plot.param` | Parameter to plot: one of 'mu', 'beta', 'lambda', 'xi', 'annual', or 'factors'. |
| `type` | One of 'map' (default; see `map_spatial_param`), 'basic' (see `plot_spatial_param`), 'location' (see `plot_location`), 'annual' (see `plot_annual`), or 'factor' (see `plot_factor`). |
| `location` | If `type='annual'` or `type='location'`, either a single integer indicating the location in the data set to plot or a vector of length 2 providing the longitude and latitude of the new location. |
| `loadings` | If `plot.param='lambda'`, an integer indicating which factor loading to plot. |
| `factor` | If `type='factor'`, an integer or vector of integers specifying which factor(s) to plot. |

## Value

A plot of the selected parameter or variable.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
plot(out.sim, fine=25)
```

---

`plot_annual` *Plot annual/seasonal behavior at a specific location.*

---

## Description

Plot annual/seasonal behavior at a specific location.

## Usage

```
plot_annual(
  out,
  location,
  add = FALSE,
  years = "one",
  interval = 0.95,
  yrange = NULL,
  new_x = NULL
)
```

## Arguments

| | |
|---|---|
| `out` | Output from BSTFA or BSTFAfull. |
| `location` | Either a single integer indicating the location in the data set to plot or a vector of length 2 providing the longitude and latitude of the new location. |
| `add` | Logical scalar indicating whether the annual/seasonal process should be added to the existing plot. Default is `FALSE`. |
| `years` | Either `'one'` (indicating to plot just a single year; default) or `'all'` (indicating to plot all years in the observed time period). |
| `interval` | Numeric value between 0 and 1 specifying the probability of the credible interval. |
| `yrange` | Numeric vector of length 2 providing the lower and upper bounds of the y-axis. If `NULL` (default), the y-axis limits are chosen using the range of the seasonal process and data. |
| `new_x` | If the original model included covariates `x`, include the same covariates for `location`. |

## Value

A plot of the annual/seasonal process at `location`.

### Author(s)

Candace Berrett and Adam Simpson

### Examples

```
data(out.sim)
attach(out.sim)
plot_annual(out.sim, location=1)
```

---

`plot_factor` *Plot the temporally-dependent factors.*

---

### Description

Plot the temporally-dependent factors.

### Usage

```
plot_factor(
  out,
  factor = 1,
  together = FALSE,
  include.legend = TRUE,
  type = "mean",
  uncertainty = TRUE,
  ci.level = c(0.025, 0.975),
  xrange = NULL
)
```

### Arguments

| | |
|---|---|
| `out` | Output from BSTFA or BSTFAfull. |
| `factor` | Integer or vector of integers specifying which factor(s) to plot. |
| `together` | If `length(factor)>1`, logical scalar specifying whether to plot all factors on a single plot. Default is `FALSE`. |
| `include.legend` | If `length(factor)>1` and `together=TRUE`, a logical scalar specifying whether to include a legend. Default is `TRUE`. |
| `type` | One of `mean` (default), `median`, `ub`, or `lb` indicating which summary statistic to plot at each location. |
| `uncertainty` | Logical scalar indicating whether to include lower and upper credible interval bounds for the parameter. Default is `FALSE`. |
| `ci.level` | A vector of length 2 specifying the quantiles to use for lower and upper bounds for `type='lb'`, `type='ub'`, or `uncertainty=TRUE`. |
| `xrange` | A date vector of length 2 providing the lower and upper bounds of the dates to include in the plot. |

## Value

A plot of spatially-dependent parameter values for a grid of interpolated locations.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
data(out.sim)
attach(out.sim)
plot_factor(out.sim, factor=1:4, together=TRUE)
```

---

`plot_fourier_bases` *Visualize fourier bases.*

---

## Description

Visualize fourier bases.

## Usage

```
plot_fourier_bases(
  coords,
  R = 6,
  fine = 100,
  plot.3d = FALSE,
  freq.lon = diff(range(coords[, 1])),
  freq.lat = diff(range(coords[, 2])),
  par.mfrow = c(2, 3)
)
```

## Arguments

| | |
|---|---|
| `coords` | A matrix of coordinates of observed locations. |
| `R` | Integer indicating the number of bases to compute. |
| `fine` | Number of grid points to include on both axes. Total grid size will be `fine^2`. Default is `100`. |
| `plot.3d` | Logical scalar indicating wether to produce a 3D plot (`TRUE`) or not (`FALSE`; default). |
| `freq.lon` | Numeric value indicating the frequency to use for the Fourier bases in the longitude direction. Default is `diff(range(coords[,1]))`. |
| `freq.lat` | Numeric value indicating the frequency to use for the Fourier bases in the latitude direction. Default is `diff(range(coords[,2]))`. |
| `par.mfrow` | If `plot.3d=TRUE`, how to divide the plotting window. See `help(par)` for more details. |

## Value

A plot of the Fourier bases for a given frequency.

## Author(s)

Adam Simpson

---

`plot_location` *Plot a location's time series of estimated/predicted values.*

---

## Description

Plot a location's time series of estimated/predicted values.

## Usage

```
plot_location(
  out,
  location,
  new_x = NULL,
  type = "mean",
  pred.int = FALSE,
  ci.level = c(0.025, 0.975),
  uncertainty = TRUE,
  xrange = NULL,
  truth = FALSE,
  ylim = NULL
)
```

## Arguments

| | |
|---|---|
| `out` | Output from BSTFA or BSTFAfull. |
| `location` | Either a single integer indicating the location in the data set to plot or a vector of length 2 providing the longitude and latitude of the new location. |
| `new_x` | If the original model included covariates `x`, include the same covariates for prediction `location`. |
| `type` | One of `mean` (default), `median`, `ub`, or `lb` indicating which summary statistic of the predicted values to return. |
| `pred.int` | Logical scalar indicating whether the interval should be a posterior predictive interval (`TRUE`) or a posterior credible interval (`FALSE`; default). |
| `ci.level` | If `type='lb'` or `'ub'`, the percentiles for the posterior interval. |
| `uncertainty` | Logical scalar indicating whether to plot the uncertainty bounds (`TRUE`; default) or not. |
| `xrange` | A date vector of length 2 providing the lower and upper bounds of the dates to include in the plot. |

| | |
|---|---|
| truth | Logical scalar indicating whether to include the observed measurements (TRUE) or not (default). If TRUE, location must be an integer corresponding to the column of the data matrix for the in-sample prediction location. |
| ylim | Numeric vector of length 2 providing the lower and upper bounds of the y-axis. If NULL (default), the y-axis limits are chosen using the range of the predictions. |

## Value

A plot of predicted values for location.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
data(out.sim)
attach(out.sim)
plot_location(out.sim, location=1, pred.int=FALSE)
```

---

# plot_spatial_param *Plot the spatially-dependent parameter for in-sample locations.*

---

## Description

Plot the spatially-dependent parameter for in-sample locations.

## Usage

```
plot_spatial_param(
  out,
  parameter,
  loadings = 1,
  type = "mean",
  ci.level = c(0.025, 0.975),
  yearscale = TRUE,
 color.gradient = (grDevices::colorRampPalette(rev(RColorBrewer::brewer.pal(9, name =
    "RdBu"))))(50),
  collims = NULL,
  printmap = FALSE
)
```

### Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |
| parameter | One of 'slope' (default), 'loading', or 'mean'. |
| loadings | If parameter='loading', an integer indicating which factor loading to plot. |
| type | One of mean (default), median, ub, or lb indicating which summary statistic to plot at each location. |
| ci.level | If type='lb' or 'ub', the percentiles for the posterior interval. |
| yearscale | If parameter='slope', a logical scalar indicating whether to translate it to a yearly scale (TRUE; default). |
| color.gradient | The color palette to use for the plot. Default is colorRampPalette(rev(RColorBrewer::brewer.pal(9 name='RdBu')))(50). |
| collims | Numeric vector of length 2 providing the lower and upper limits for the color scale. If NULL (default), the limits are set to be symmetric around zero based on the maximum absolute value of the parameter being plotted. |
| printmap | Logical scalar indicating whether to plot the resulting map or not. Default is TRUE. |

### Value

A plot of spatially-dependent parameter values for the observed locations.

### Author(s)

Adam Simpson and Candace Berrett

### Examples

```
data(out.sim)
attach(out.sim)
plot_spatial_param(out.sim, parameter='slope')
```

---

## plot_trace *Plot trace plots*

---

### Description

Plot trace plots

### Usage

```
plot_trace(
  out,
  parameter,
  param.range = NULL,
  par.mfrow = c(1, 1),
  density = TRUE
)
```

### Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |
| parameter | Parameter to plot. See BSTFA and BSTFAfull for parameter names. |
| param.range | Indices of the named parameter to plot. Default is to plot all relevant parameters. |
| par.mfrow | Vector of length 2 indicating the number of rows and columns to divide the plotting window. |
| density | Logical scalar indicating whether to include the density plot of the posterior draws. Default is TRUE. |

### Value

A plot containing the trace plot (and density plot when density=TRUE) of the listed parameters.

### Author(s)

Adam Simpson

### Examples

```
data(out.sim)
attach(out.sim)
plot_trace(out.sim, parameter='beta', param.range=1)
```

---

## predictBSTFA *Estimate/predict values of the time series at a specific location.*

---

### Description

Estimate/predict values of the time series at a specific location.

### Usage

```
predictBSTFA(
  out,
  location = NULL,
  type = "mean",
  ci.level = c(0.025, 0.975),
  new_x = NULL,
  pred.int = FALSE
)
```

### Arguments

| | |
|---|---|
| out | Output from BSTFA or BSTFAfull. |
| location | Either a single integer indicating the location in the data set to provide predictions or a vector of length 2 providing the longitude and latitude of the new location. If location=NULL (default), the function will return predictions for all in-sample locations. |
| type | One of all, mean (default), median, ub, or lb indicating which summary statistic of the predicted values to return. |
| ci.level | If type='lb' or 'ub', the percentiles for the posterior interval. |
| new_x | If the original model included covariates x, include the same covariates for prediction location. |
| pred.int | Logical scalar indicating whether to include additional uncertainty for posterior predictive intervals (TRUE) or not (FALSE; default – intervals represent posterior probabilities around the mean of location). |

### Value

A matrix or vector of estimated/predicted values for location.

### Author(s)

Candace Berrett and Adam Simpson

### Examples

```
data(out.sim)
attach(out.sim)
loc1means <- predictBSTFA(out.sim, location=1, pred.int=FALSE)
```

---

## print.bstfa *Print for 'bstfa' Objects*

---

### Description

This function prints the names and properties of a 'bstfa' object.

### Usage

```
## S3 method for class 'bstfa'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | A bstfa object. |
| ... | Additional arguments passed to or from other methods. |

## Value

A data frame containing the following elements:

**post.mean** Posterior mean of parameters for each observed location.

**post.sd** Posterior standard deviation of parameters for each observed location.

**post.pgt0** Posterior probability that the parameter is greater than 0, if relevant.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
print(out.sim)
```

---

`summary.bstfa` *Summary for 'bstfa' Objects*

---

## Description

This function provides posterior mean, standard deviations, and probabilities of being greater than 0 of interpretable parameters.

## Usage

```
## S3 method for class 'bstfa'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | A bstfa object. |
| `...` | Additional arguments passed to or from other methods. |

## Value

A data frame containing the following elements:

**post.mean** Posterior mean of parameters for each observed location.

**post.sd** Posterior standard deviation of parameters for each observed location.

**post.pgt0** Posterior probability that the parameter is greater than 0, if relevant.

## Author(s)

Candace Berrett and Adam Simpson

## Examples

```
summary(out.sim)
```

`utahDataList` *Utah Minimum Temperatures*

## Description

Zero-centered daily minimum temperatures averaged across 30-day windows from 1919 to 2014 across the US state of Utah; also includes dates, coordinates, and station names. Originally collected from the Utah Climate Center.

## Usage

```
utahDataList
```

## Format

A data set with 4 variables:

**TemperatureVals** A 1251 by 146 matrix of zero-centered 30-day average daily minimum temperatures from 1912 through 2014. Missing observations are denoted using `NA`.

**Dates** A vector of length 1251 and class `Date` providing the day of each observation in the same order as the rows of `TemperatureVals`. Note that this package expects observation times to be regularly spaced.

**Coords** A 142 by 2 data frame containing the longitude (first variable) and latitude (second variable) of measured locations.

**Locations** A character vector of length 146 containing the station names for measured locations in the same order as the columns of `TemperatureVals` and the rows of `Coords`.

## Source

https://climate.usu.edu

# Index

∗ **datasets**
out.sim, 22
utahDataList, 33

bisquare1d, 2
bisquare2d, 3
BSTFA, 4
BSTFAfull, 9

coef.bstfa, 16
compute_summary, 16
computeLogLik, 17
convergence_diag, 18

defineRegion, 19

makeNewS, 19
makePredS, 20
map_spatial_param, 21

out.sim, 22

plot.bstfa, 23
plot_annual, 24
plot_factor, 25
plot_fourier_bases, 26
plot_location, 27
plot_spatial_param, 28
plot_trace, 29
predictBSTFA, 30
print.bstfa, 31

summary.bstfa, 32

utahDataList, 33