Package ‘bandle’

March 6, 2024

Type  Package
Title  An R package for the Bayesian analysis of differential subcellular localisation experiments
Version  1.6.0
Description  The Bandle package enables the analysis and visualisation of differential localisation experiments using mass-spectrometry data. Experimental methods supported include dynamic LOPIT-DC, hyperLOPIT, Dynamic Organellar Maps, Dynamic PCP. It provides Bioconductor infrastructure to analyse these data.
License  Artistic-2.0
Encoding  UTF-8
Depends  R (>= 4.1), S4Vectors, Biobase, MSnbase, pRoloc
Imports  Rcpp (>= 1.0.4.6), pRolocdata, lbfgs, ggplot2, dplyr, plyr, knitr, methods, BiocParallel, robustbase, BiocStyle, ggalluvial, ggrepel, tidyr, circlize, graphics, stats, utils, grDevices, rlang
Suggests  coda (>= 0.19-4), testthat, interp, fields, pheatmap, viridis, rmarkdown, spelling
VignetteBuilder  knitr
LinkingTo  Rcpp, RcppArmadillo, BH
Roxygen  list(markdown=TRUE)
RoxygenNote  7.2.0
biocViews  Bayesian, Classification, Clustering, ImmunoOncology, QualityControl, DataImport, Proteomics, MassSpectrometry
BugReports  https://github.com/ococrook/bandle/issues
URL  http://github.com/ococrook/bandle
Language  en-US
git_url  https://git.bioconductor.org/packages/bandle
git_branch  RELEASE_3_18
git_last_commit  94e6ac6
The Bandle package enables the analysis and visualisation of differential localisation experiments using mass-spectrometry data. Experimental methods supported include dynamic LOPIT-DC, hyperLOPIT, Dynamic Organellar Maps, Dynamic PCP. It provides Bioconductor infrastructure to analyse these data.
Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Oliver M. Crook [aut, cre] (<https://orcid.org/0000-0001-5669-8506>), Lisa Breckels [aut] (<https://orcid.org/0000-0001-8918-7171>)

Maintainer: Oliver M. Crook <oliver.crook@stats.ox.ac.uk>

References

~~ Literature or other references for background information ~~

bandle


differential localisation experiments using the bandle method

Description

These function implement the bandle model for dynamic mass spectrometry based spatial proteomics datasets using MCMC for inference

These functions implement the bandle model for dynamic mass spectrometry based spatial proteomics datasets using MCMC for inference, this is an internal sampling function

Usage

bandle(
  objectCond1,
  objectCond2,
  fcol = "markers",
  hyperLearn = "LBFGS",
  numIter = 1000,
  burnin = 100L,
  thin = 5L,
  u = 2,
  v = 10,
  lambda = 1,
  gpParams = NULL,
  hyperIter = 20,
  hyperMean = c(0, 0, 0),
  hyperSd = c(1, 1, 1),
  seed = NULL,
  pg = FALSE,
  pgPrior = NULL,
tau = 0.2,
dirPrior = NULL,
maternCov = TRUE,
PC = TRUE,
pcPrior = matrix(c(0.5, 3, 100), nrow = 1),
nu = 2,
propSd = c(0.3, 0.1, 0.05),
numChains = 4L,
BPPARAM = BiocParallel::bpparam()
)

diffLoc(
  objectCond1,
  objectCond2,
  fcol = "markers",
  hyperLearn = "MH",
  numIter = 1000,
  burnin = 100L,
  thin = 5L,
  u = 2,
  v = 10,
  lambda = 1,
  gpParams = NULL,
  hyperIter = 20,
  hyperMean = c(0, 0, 0),
  hyperSd = c(1, 1, 1),
  seed = NULL,
  pg = TRUE,
  pgPrior = NULL,
  tau = 0.2,
  dirPrior = NULL,
  maternCov = TRUE,
  PC = TRUE,
  nu = 2,
  pcPrior = NULL,
  propSd = c(0.3, 0.1, 0.05)
)

Arguments

objectCond1  A list of MSnbase::MSnSets where each is an experimental replicate for the first condition, usually a control

objectCond2  A list of MSnbase::MSnSets where each is an experimental replicate for the second condition, usually a treatment

fcol  The feature meta-data containing marker definitions. Default is markers

hyperLearn  Algorithm to learn posterior hyperparameters of the Gaussian processes. Default is LBFGS and MH for metropolis-hastings is also implemented.
**numIter**  The number of iterations of the MCMC algorithm. Default is 1000. Though usually much larger numbers are used

**burnin**  The number of samples to be discarded from the beginning of the chain. Default is 100.

**thin**  The thinning frequency to be applied to the MCMC chain. Default is 5.

**u**  The prior shape parameter for Beta(u, v). Default is 2

**v**  The prior shape parameter for Beta(u, v). Default is 10.

**lambda**  Controls the variance of the outlier component. Default is 1.

**gpParams**  Object of class gpParams. parameters from prior fitting of GPs to each niche to accelerate inference. Default is NULL.

**hyperIter**  The frequency of MCMC iteration to update the hyper-parameters default is 20

**hyperMean**  The prior mean of the log normal prior of the GP parameters. Default is 0 for each. Order is length-scale, amplitude and noise variance

**hyperSd**  The prior standard deviation of the log normal prior for the GP parameters. Default is 1 for each. Order is length-scale, amplitude and noise variance.

**seed**  The random number seed.

**pg**  logical indicating whether to use polya-gamma prior. Default is FALSE.

**pgPrior**  A matrix generated by pgPrior function. If param pg is TRUE but pgPrior is NULL then a pgPrior is generated on the fly.

**tau**  The tau parameter for the polya-Gamma prior (is used). Defaults to 0.2

**dirPrior**  A matrix generated by dirPrior function. Default is NULL and dirPrior is generated on the fly.

**maternCov**  logical indicated whether to use a matern or gaussian covariance. Default is TRUE and matern covariance is used

**PC**  logical indicating whether to use a penalised complexity prior. Default is TRUE.

**pcPrior**  matrix with 3 columns indicating the lambda parameters for the penalised complexity prior. Default is null which internally sets the penalised complexity prior to c(0.5, 3, 100) for each organelle and the order is length-scale, amplitude and variance. See vignette for more details.

**nu**  integer indicating the smoothness of the matern prior. Default is 2.

**propSd**  If MH is used to learn posterior hyperparameters then the proposal standard deviations. A Gaussian random-walk proposal is used.

**numChains**  integer indicating the number of parallel chains to run. Defaults to 4.

**BPPARAM**  BiocParallel parameter. Defaults to machine default backend using bpparam()

### Details

The `bandle` function generate the sample from the posterior distributions (object or class `bandleParams`) based on an annotated quantitative spatial proteomics datasets (object of class `MSnbase::MSnSet`). Both are then passed to the `bandlePredict` function to predict the sub-cellular localisation and compute the differential localisation probability of proteins. See the vignette for examples.
The `diffloc` function generates the sample from the posterior distributions (object or class `bandleParam`) based on an annotated quantitative spatial proteomics dataset (object of class `MSnbase::MSnSet`). Both are then passed to the `bandlePredict` function to predict the sub-cellular localisation and compute the differential localisation probability of proteins. See the vignette for examples.

### Value

- `bandle` returns an instance of class `bandleParams`
- `bandle` returns an instance of class `bandleParams`

### Examples

```r
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
numRep = 6L,
numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPMaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1,
objectCond2 = treatment1, gpParams = gpParams,
fcol = "markers", numIter = 5L, burnin = 1L, thin = 2L,
numChains = 1, BPPARAM = SerialParam(RNGseed = 1))
```

```r
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
numRep = 6L,
numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPMaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- diffLoc(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
fcol = "markers", numIter = 5L, burnin = 1L, thin = 2L)
```

---

### bandleChains-class

**Infrastructure to to store and process MCMC results**

### Description

The `bandleChains` infrastructure is used to store and process MCMC results for bandle model from Crook et al 2021.
Usage

chains(object)

## S4 method for signature 'bandleParams'
show(object)

## S4 method for signature 'nicheParam'
show(object)

## S4 method for signature 'bandleChain'
show(object)

## S4 method for signature 'bandleChains'
length(x)

## S4 method for signature 'bandleParams'
length(x)

## S4 method for signature 'bandleSummaries'
length(x)

## S4 method for signature 'nicheParams'
length(x)

## S4 method for signature 'nicheParams'
length(x)

posteriorEstimates(object)

## S4 method for signature 'bandleSummary'
posteriorEstimates(object)

summaries(object)

params(object)

bandleJoint(object)

## S4 method for signature 'bandleSummary'
bandleJoint(object)

## S4 method for signature 'bandleChains,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'bandleParams,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'bandleChains,ANY,ANY,ANY'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'bandleParams,ANY,ANY,ANY'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'bandleChains'
show(object)

## S4 method for signature 'bandleSummaries'
show(object)

## S4 method for signature 'bandleSummaries,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'bandleSummaries,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'bandleSummaries,ANY,ANY,ANY'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'nicheParams,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'nicheParams,ANY,ANY'
x[[i, j = "missing", drop = "missing"]]

## S4 method for signature 'nicheParams,ANY,ANY,ANY'
x[i, j = "missing", drop = "missing"]

## S4 method for signature 'nicheParams'
show(object)

Arguments

- **object**: object of class nicheParams.
- **x**: Object to be subset.
- **i**: An integer(). Should be of length 1 for [].
- **j**: Missing.
- **drop**: Missing.

Details

Objects of the bandleParams class are created with the bandle() function. These objects store the priors for the model and the results of the MCMC chains, which themselves are stored as an instance of class bandleChains and can be accessed with the chains() function. A summary of the bandleChains (or class bandleSummary) can be further computed with the bandleProcess function.

see the bandle vignette for examples
Value

An object of class `bandleParams` which stores the main results for the analysis when using `bandle`.

Slots

- `chains` list() containing the individual full MCMC chain results in a `bandleChains` instance. Each element must be a valid `bandleChain` instance.
- `posteriorEstimates` A `data.frame` documenting the posteriors in a `bandleSummary` instance diagnostics A matrix of dimensions 1 by 2 containing the `bandleSummary` diagnostics.
- `bandle.join` A matrix of dimensions N by K storing the joint probability in an `bandleSummary` instance for each of the first condition chains list() containing the individual `bandleSummary` instances for different conditions results in an `bandleSummary` instance. Each element must be a valid `bandleSummary` instance.
- `method` A character() storing the `bandle` method name
- `priors` A list() with the priors for the parameters
- `seed` An integer() with the random number generation seed.
- `summary` Object of class `bandleSummary` the summarised MCMC results available in the `bandleParams` instance.
- `chains` Object of class `bandleChains` containing the full MCMC results in the `bandleParams` instance
- `dataset` character indicating which dataset i.e control or treatment
- `replicate` integer an integer indicating which replicate
- `K` integer(1) indicating the number of components.
- `D` integer(1) indicating the number of samples.
- `method` character(1) defining the method used. Currently `bandle`
- `mk` matrix(K, D)
- `lambda_k` numeric(K)
- `nuk` numeric(K)
- `sk` array(K, D, D)
- `params` list() containing the individual `nicheParam` objects results in an `bandleParams` instance. Each element must be a valid `bandleParam` instance.
- `dataset` character indicating the dataset usually control or treatment
- `replicate` integer indicating the number of dataset replicate
- `n` integer(1) indicating the number of MCMC interactions. Stored in an `bandleChain` instance.
- `K` integer(1) indicating the number of components. Stored in an `bandleChain` instance.
- `N` integer(1) indicating the number of proteins. Stored in an `bandleChain` instance.
- `niche` matrix(N, n) component allocation results of an `bandleChain` instance.
- `nicheProb` matrix(N, n, K) component allocation probabilities of an `bandleChain` instance.
- `outlier` matrix(N, n) outlier allocation results.
- `outlierProb` matrix(N, n, 2) outlier allocation probabilities of an `bandleChain` instance.
bandlePredict  

Make predictions from a bandle analysis

Description

Make predictions from a bandle analysis

Usage

bandlePredict(objectCond1, objectCond2, params, fcol = "markers")

Arguments

objectCond1    A list of instances of class MSnbase::MSnSets where each is an experimental replicate for the first condition, usually a control
objectCond2    A list of instance of class MSnbase::MSnSets where each is an experimental replicate for the second condition, usually a treatment
params         An instance of class bandleParams, as generated by bandle().
fcol            A feature column indicating the markers. Defaults to "markers"

Value

bandlePredict returns an instance of class MSnbase::MSnSet containing the localisation predictions as a new bandle.allocation feature variable. The allocation probability is encoded as bandle.probability (corresponding to the mean of the distribution probability). In addition the upper and lower quantiles of the allocation probability distribution are available as bandle.probability.lowerquantile and bandle.probability.upperquantile feature variables. The Shannon entropy is available in the bandle.mean.shannon feature variable, measuring the uncertainty in the allocations (a high value representing high uncertainty; the highest value is the natural logarithm of the number of classes). An additional variable indicating the differential localization probability is also added as bandle.differential.localisation

Examples

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
                      numRep = 6L,
                      numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x)
                   fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
                fcol = "markers", numIter = 5L, burnin = 1L, thin = 2L,
bandleProcess 11

numChains = 1, BPPARAM = SerialParam(RNGseed = 1))

mcmc1 <- bandleProcess(mcmc1)
out <- bandlePredict(objectCond1 = control1, objectCond2 = treatment1, params = mcmc1)

bandleProcess

process bandle results

Description

process bandle results

Usage

bandleProcess(params)

Arguments

params An object of class bandleParams

Value

bandleProcess returns an instance of class bandleParams with its summary slot populated.

Examples

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 6L,
  numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x)
  fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1))
  d1 <- tansim$lopitrep
  control1 <- d1[1:3]
  treatment1 <- d1[4:6]
  mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
    fcol = "markers", numIter = 5L, burnin = 1L, thin = 2L,
    numChains = 1, BPPARAM = SerialParam(RNGseed = 1))
  mcmc1 <- bandleProcess(mcmc1)
**besselK boost**  
*bessel function of the second kind from boost library*

**Description**
Leapfrog routine
Leapfrog routine

**Usage**

```
besselK_boost(x, v)
besselK(x, v)
matern(nu, a, rho, tau, D)
trenchDetcpp(c)
trenchInvcpp(v)
loglikeGPcpp(Y, Z, A, logcovDet, sigmak, nk, D, Y2)
likelihoodGPcpp(Xk, tau, h, nk, D, materncov = 0L, nu = 2)
gradientrhomatern(Y, drvrhomatern, nk, D, Z, A, sigmak)
gradientamatern(Y, amatern, nk, D, Z, A, sigmak)
gradientGPcppmatern(Xk, tau, h, nk, D, nu)
LeapfrogGPcppPC(Xk, lambda, tau, p, x, m, nk, D, L, delta, nu)
sampleGPmeanmaterncpp(Xk, tau, h, nk, D, nu)
makeComponent(X, BX, Y, BY, j)
sampleGPmeancpp(Xk, tau, h, nk, D)
normalisedData(Xknown, BX, Xunknown, BXun, hypers, nk, tau, D, j)
normalisedDatamatern(Xknown, BX, Xunknown, BXun, hypers, nk, tau, D, j, nu)
centeredDatamatern(Xknown, BX, Xunknown, BXun, hypers, nk, tau, D, K, nu)
componentloglike(centereddata, sigmak)
```
besselK_boost

comploglike(centereddata, sigmak)
comploglikelist(centereddata, sigmak)
sampleDirichlet(numSamples, alpha)
sampleOutlierCpp(allocoultierprob)
sampleAllocCpp(allocreprob)
centeredData(Xknown, BX, Xunknown, BXun, hypers, nk, tau, D, K)
mahaInt(X, mu, sigma, isChol = FALSE)
dmvInt(X, mu, cholDec, log, df)
dmvIntCpp(X, mu_, sigma_, df_, log_, isChol_)
gradientsGPcpp(Xk, tau, h, nk, D)
LeapfrogGPcpp(Xk, tau, p, x, m, nk, D, L, delta)
rcpp_pgdraw(b, c)

Arguments

x position
v argument of trench algorithm
nu smoothness parameter of matern covariance
a amplitude
rho length-scale
tau indexing term
D number of samples
c parameter of PG distribution
Y pointer to data to be subset. X and Y will be joined
Z special matrix from trench algorithm (see Crook et al arxiv 2019)
A special matrix from trench algorithm (see Crook et al arxiv 2019)
logcovDet log determine of the covariancematrix
sigmak variance term
nk number of observations
Y2 vectorised data (see Crook et al arxiv 2019)
Xk The data
h vector of hyperparamters
materncov logical indicating whether to use matern or gaussian covariance. Defaults to Gaussian covariance
drvrhomatern deterivate of matern covariance wrt to rho
amatern deterivate of matern covariance wrt to amplitude
lambda parameters of penalised complexity prior
p momentum
m mass
L iterations
delta stepsize
X data
BX indexing set to make component
BY pointer to subsetting matrix
j indicator of localisations i.e. niche j
Xknown data with known localisations
Xunknown data with unknown localisations
BXun indexing set for unknown localisations
hypers vector of hyperparameters
K number of components
centereddata pointer to centered data
numSamples The number of samples desired
alpha The concentration parameter
allocoutlierprob The probabilities of being allocated to the outlier component
allocprob probability of being allocated to particular component
mu mean
sigma variance matrix
isChol boolean indicated whether sigma is cholesky decomposition
cholDec Cholesky decomposition of variance matrix
log boolean of log density
df degrees of freedom for t distribution
X_ the data
mu_ the mean
sigma_ the variance matrix
df_ the degrees of freedom
log_ return log density (boolean).
isChol_ is variance matrix in cholesky decomposition
b parameter of PG distribution
diffLocalisationProb

Value

A numeric indicating the density of the t-distribution

Examples

dmvCcpp(diag(1,1,1), 1, diag(1,1,1), 1, TRUE, TRUE)

diffLocalisationProb

Compute differential localisation probabilities from ms-based experiments using the bandle method

Description

These functions implement helper functions for the bandle method

Usage

diffLocalisationProb(params)

bootstrapdiffLocprob(params, top = 20, Bootsample = 5000, decreasing = TRUE)

binomialDiffLocProb(params, top = 20, nsample = 5000, decreasing = TRUE)

Arguments

params An instance of bandleParams

top The number of proteins for which to sample from the binomial distribution

Bootsample Number of Bootstrap samples. Default is 5000

decreasing Starting at protein most likely to be differentially localization

nsample how many samples to return from the binomial distribution

Value

returns a named vector of differential localisation probabilities

returns a matrix of size Bootsample * top containing bootstrap

returns a list containing empirical binomial samples
Examples

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
    fcol = "markers", numIter = 10L, burnin = 1L, thin = 2L,
    numChains = 1, BPPARAM = SerialParam(RNGseed = 1))

dp <- diffLocalisationProb(mcmc1)

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
    fcol = "markers", numIter = 10L, burnin = 1L, thin = 2L,
    numChains = 1, BPPARAM = SerialParam(RNGseed = 1))

dp <- diffLocalisationProb(mcmc1)

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
    fcol = "markers", numIter = 10L, burnin = 1L, thin = 2L,
    numChains = 1, BPPARAM = SerialParam(RNGseed = 1))

dp <- binomialDiffLocProb(mcmc1)
EFDR

Compute the expected False Discovery Rate

Description

The EFDR for a given threshold is equal to the sum over all proteins that exceed that threshold of one minus the posterior probability of differential localisations, divides by the total number of proteins with probabilities of differential localisation greater than that threshold.

Usage

EFDR(prob, threshold = 0.9)

Arguments

prob A numeric indicating probabilities of differential localisation
threshold A numeric indicating the probability threshold. The default is 0.90.

Value

The expected false discovery rate for a given threshold

Examples

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 6L,
  numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x)
  fitGPMaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mcmc1 <- bandle(objectCond1 = control1, objectCond2 = treatment1, gpParams = gpParams,
  fcol = "markers", numIter = 10L, burnin = 1L, thin = 2L,
  numChains = 1, BPPARAM = SerialParam(RNGseed = 1))
mcmc1 <- bandleProcess(mcmc1)
dp <- diffLocalisationProb(mcmc1)
EFDR(dp, threshold = 0.5)
fitGP

Fit a Gaussian process to spatial proteomics data

Description

The fitGP function is a helper function to fit GPs with squared exponential co-variances, maximum marginal likelihood.

The fitGPmaternPC function is a helper function to fit matern GPs to data with penalised complexity priors on the hyperparameters.

The fitGPmatern function fits matern GPs to data.

The plotGPmatern function plots matern GPs.

Usage

fitGP(object = object, fcol = "markers")

fitGPmaternPC(
  object = object,
  fcol = "markers",
  materncov = TRUE,
  nu = 2,
  hyppar = matrix(c(10, 60, 250), nrow = 1)
)

fitGPmatern(object = object, fcol = "markers", materncov = TRUE, nu = 2)

plotGPmatern(object = object, params = params, fcol = "markers")

Arguments

object A instance of class MSnSet

fcol feature column to indicate markers. Default is "markers".

materncov logical indicating whether matern covariance is used.

nu matern smoothness parameter. Default is 2.

hyppar The vector of penalised complexity hyperparameters, you must provide a matrix with 3 columns and 1 row. The order is hyperparameters on length-scale, amplitude, variance.

params The output of running fitGPmatern, fitGPmaternPC or fitGP which is of class gpParams

Details

This set of functions allow users to fit GPs to their data. The fitGPmaternPC function allows users to pass a vector of penalised complexity hyperparameters using the hyppar argument. You must provide a matrix with 3 columns and 1 row. The order of these 3 columns represent the...
fitGP

hyperparameters length-scale, amplitude, variance. We have found that the matrix(c(10, 60, 250), nrow = 1) worked well for the spatial proteomics datasets tested in Crook et al (2021). This was visually assessed by passing these values and visualising the GP fit using the plotGPmatern function (please see vignette for an example of the output). Generally, (1) increasing the lengthscale parameter (the first column of the hyppar matrix) increases the spread of the covariance i.e. the similarity between points, (2) increasing the amplitude parameter (the second column of the hyppar matrix) increases the maximum value of the covariance and lastly (3) decreasing the variance (third column of the hyppar matrix) reduces the smoothness of the function to allow for local variations. We strongly recommend users start with the recommended parameters and change and assess them as necessary for their dataset by visually evaluating the fit of the GPs using the plotGPmatern function. Please see the vignettes for more details and examples.

Value

Returns an object of class gpParams which stores the posterior predictive means, standard deviations, variances and also the MAP hyperparamters for the GP.

The functions plotGPmatern plot the posterior predictives overlayed with the markers for each subcellular class.

Examples

```r
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGP(x))
```

```r
## ====== fitGPmaternPC =====
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
## Please note that hyppar should be chosen carefully and tested
## by checking the GP fit with the plotGPmatern function
## (please see details above)
gpParams <- lapply(tansim$lopitrep,
    function(x) fitGPmaternPC(x, hyppar = matrix(c(10, 60, 100), nrow = 1)))
```

```r
## ====== fitGPmatern =====
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
gpParams <- lapply(tansim$lopitrep, function(x) fitGPmatern(x))
```
## ====== plotGPmatern =====
## generate example data
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
    numRep = 6L,
    numDyn = 100L)
## fit a GP
gpParams <- lapply(tansim$lopitrep, function(x) fitGP(x))

## Overlay posterior predictives onto profiles
## Dataset 1
par(mfrow = c(2, 3))
plotGPmatern(tansim$lopitrep[[1]], gpParams[[1]])

## Dataset 2, etc.
par(mfrow = c(2, 3))
plotGPmatern(tansim$lopitrep[[2]], gpParams[[2]])

---

**gpParams-class**  
*Container for GP results*

### Description

The gpParams infrastructure is used to store and process the GP results for output from using the `fitGP` functions in `bandle`.

### Details

Objects of the `gpParams` class are created with the `fitGP`, `fitGPmaternPC` or `fitGPmatern` functions.

These objects a list of posterior predictive means and standard deviations. As well as maximum marginal likelihood for the GP.

### Slots

- `method` character indicating the GP method used
- `M` A list of the posterior predictive means for each K components of GPs fitted to the data
- `sigma` A numeric of length K standard deviations fitted to the data
- `V` A list of the variance fitted to the data
- `params` A matrix array of the MAP hyperparameters for the GP
gradientGP

Compute GP gradient

Description
Internal R function to pass R to C++, not for external use.
Internal R function to pass R to C++, not for external use.
Function to perform Metropolis-Hastings for GP hyperparameters with different priors

Usage
gradientGP(Xk, tau, h, nk, D)
gradientGPmatern(Xk, tau, h, nk, D, materncov, nu)
posteriorgradientGPmatern(Xk, tau, h, nk, D, materncov, nu, hyppar)
gradientlogprior(h, hyppar)
likelihoodGP(Xk, tau, h, nk, D)
likelihoodGPmatern(Xk, tau, h, nk, D, materncov, nu)
posteriorGPmatern(Xk, tau, h, nk, D, materncov, nu, hyppar)
Gumbel(x, lambda, log = TRUE)
PCrhomvar(rho, a, lambda1, lambda2, log = TRUE)

metropolisGP(
  inith,
  X,
  tau,
  nk,
  D,
  niter,
  hyperMean = c(0, 0, 0),
  hyperSd = c(1, 1, 1)
)

metropolisGPmatern(
  inith,
  X,
  tau,
  nk,
  D,
niter,
nu = 2,
hyppar = c(1, 1, 1),
propSd = c(0.3, 0.1, 0.1)
)

Gumbel(x, lambda, log = TRUE)

PCrhomvar(rho, a, lambda1, lambda2, log = TRUE)

Arguments

- **Xk** The data
- **tau** The indexing parameters
- **h** GP hyperparameters
- **nk** Number of observations
- **D** number of samples
- **materncov** logical indicating whether matern covariance is used
- **nu** Smoothness of the matern covariance
- **hyppar** A vector indicating the penalised complexity prior hyperparameters. Default is c(1,1,1)
- **x** observation
- **lambda** scale parameter of the type-2 Gumbel distribution
- **log** logical indicating whether to return log. Default is TRUE
- **rho** length-scale parameter
- **a** amplitude
- **lambda1** first parameter of distribution
- **lambda2** second parameter of distribution
- **inith** initial hyperparameters
- **X** The data
- **niter** Number of MH iterations
- **hyperMean** A vector indicating the log-normal means. Default is c(0,0,0).
- **hyperSd** A vector indicating the log-normal standard deviations. Default is c(1,1,1)
- **propSd** The proposal standard deviation. Default is c(0.3,0.1,0.1). Do not change unless you know what you are doing.

Value

- Returns gp gradient
- Returns gp gradient
- Returns the gradient of the posterior
- return the gradient of the log prior, length-scale, amplitude and noise
kldirpg

Returns gp negative log likelihood
Returns gp negative log likelihood
Returns the negative log posterior of the GP
Returns the likelihood of the type-2 Gumbel distribution
Returns the likelihood of the bivariate penalised complexity prior
Returns new hyperparameters and the acceptance rate
Returns the likelihood of the type-2 Gumbel distribution
Returns the likelihood of the bivariate penalised complexity prior

Examples

Gumbel(3, lambda = 1)

---

kldirpg

*Computes the Kullback-Leibler divergence between Polya-Gamma and Dirichlet priors*

**Description**
Computes the Kullback-Leibler divergence between Polya-Gamma and Dirichlet priors

Compute the KL divergence between two Dirichlet distributions

A function to compute the prior predictive distribution of the Dirichlet prior.

A function to compute the prior predictive distribution of the Polya-Gamma prior.

**Usage**

kldirpg(sigma = diag(1, 1, 1), mu = c(0, 0, 0), alpha = c(1))

kldir(alpha, beta)

prior_pred_dir(object, fcol = "markers", iter = 5000, dirPrior = NULL, q = 15)

prior_pred_pg(
  objectCond1,
  objectCond2,
  fcol = "markers",
  tau = 0.2,
  lambda = 0.01,
  mu_prior = NULL,
  iter = 10000,
  q = 15
)
Arguments

sigma the sigma parameter of the Polya-Gamma prior. A positive-definite symmetric matrix.
mu the mu parameter of the Polya-Gamma prior. A vector of means
alpha The concentration parameter of the first Dirichlet distribution
beta The concentration parameter of the second Dirichlet distribution
object An instance of class MSnSet
fcol The feature column indicating the markers. Default is "markers"
iter Number of sample to use from prior predictive distribution. Default is 10000
dirPrior The Dirichlet prior used. If NULL (default) will generate a default Dirichlet prior. This should be a matrix with the same dimensions as the number of subcellular niches. The diagonal terms correspond to the prior probability of not differentially localising. The (i,j) term corresponds to prior probability of differentially localising between niche i and j.
q The upper tail value. That is the prior probability of having more than q differential localisations. Default is 15.
objectCond1 An instance of class MSnSet, usually the control dataset
objectCond2 An instance of class MSnSet, usually the treatment dataset
tau The tau parameter of the Polya-Gamma prior. Default is 0.2.
lambda The lambda ridge parameter used for numerical stability. Default is 0.01
mu_prior The mean of the Polya-Gamma prior. Default is NULL which generates a default Polya-Gamma prior.

Value

returns a numeric indicating the KL divergence
a numeric indicating the KL divergence
A list contain the prior predictive distribution of differential localisations, the mean number of differential localised proteins and the probability than more than q are differentially localised
A list contain the prior predictive distribution of differential localisations, the mean number of differential localised proteins and the probability than more than q are differentially localised

Examples

kldirpg(sigma = diag(c(1,1,1)), mu = c(0,0,0), alpha = 1)
kldir(c(1,1), c(3,1))
library(pRolocdata)
data("tan2009r1")
out <- prior_pred_dir(object = tan2009r1)
library(pRolocdata)
mcmc_plot_probs

Generate a violin plot showing the probability of protein localisation to different organelles

Description
These functions implement plotting functions for bandle objects

Usage

mcmc_plot_probs(
  params,
  fname,
  cond = 1,
  n = 1,
  bw = 0.05,
  scale = "width",
  trim = TRUE
)

Arguments

params An instance of class bandleParams
fname The name of the protein to plot
cond Which conditions do we want to plot. Must be 1 or 2. Default is 1
n The chain from which we plot the probability distribution. Default is 1.
bw The bandwidth use in probability distribution smoothing of geom_violin Default is 0.05.
scale Scaling of geom_violin. Defaults to width.
trim trim parameter of geom_violin. Defaults to true.

Value

returns a named vector of differential localisation probabilities
meanOrganelle

*Computes Organelle means and variances using markers*

**Description**

Computes Organelle means and variances using markers.

**Usage**

```r
meanOrganelle(object, fcol = "markers")
```

**Arguments**

- `object` a instance of class `MSnset`
- `fcol` a feature column indicating which feature define the markers

**Value**

returns a list of means and variances for each

**Examples**

```r
library(pRolocdata)
data("tan2009r1")
meanOrganelle(object = tan2009r1)
```
plotConvergence  Generates a histogram of ranks (a rank plot) for convergence

Description

Produces a rank plot to analyse convergence of MCMC algorithm

Usage

plotConvergence(params)

Arguments

params                An instance of class bandleParams

Value

Returns the ranks of the number of outliers in each chain. The side effect returns rank plots. Number of rank plots is equal to the number of chains

Examples

## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
            numRep = 4L,
            numDyn = 100L)
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]

## fit GP params
gpParams <- lapply(tansim$lopitrep, function(x)
            fitGPMaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))

## run bandle
res <- bandle(objectCond1 = control,
              objectCond2 = treatment,
              gpParams = gpParams,
              fcol = "markers",
              numIter = 5L,
              burnin = 1L,
              thin = 2L,
              numChains = 2,
              BPPARAM = SerialParam(RNGseed = 1),
              seed = 1)

## Process bandle results
bandleres <- bandleProcess(res)

## Convergence plots
par(mfrow = c(1, 2))
plotConvergence(bandleres)

---

**plotTable**

Generates a table for visualising changes in localisation between two conditions/datasets

### Description

Produces a table summarising differential localisation results

### Usage

plotTable(params, all = FALSE, fcol)

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>An instance of class bandleParams or an instance of class MSnSetList of length 2.</td>
</tr>
<tr>
<td>all</td>
<td>A logical specifying whether to count all proteins or only show those that have changed in location between conditions. Default is FALSE.</td>
</tr>
<tr>
<td>fcol</td>
<td>If params is a list of MSnSets. Then fcol must be defined. This is a character vector of length 2 to set different labels for each dataset. If only one label is specified, and the character is of length 1 then this single label will be used to identify the annotation column in both datasets.</td>
</tr>
</tbody>
</table>

### Value

Returns a summary table of translocations of proteins between conditions.

### Examples

```r
## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
                      numRep = 4L,
                      numDyn = 100L)
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]

## fit GP params
gpParams <- lapply(tansim$lopitrep, function(x)
  fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))
```
## run bandle

```
res <- bandle(objectCond1 = control,
               objectCond2 = treatment,
               gpParams = gpParams,
               fcol = "markers",
               numIter = 5L,
               burnin = 1L,
               thin = 2L,
               numChains = 2,
               BPPARAM = SerialParam(RNGseed = 1),
               seed = 1)
```

## Process bandle results

```
bandleres <- bandleProcess(res)
```

## Tabulate results

```
plotTable(bandleres)
```

---

**plotTranslocations**  
*Generates a chord diagram or alluvial plot for visualising changes in localisation between two conditions/datasets*

### Description

Produces a chord diagram (circos plot) or an alluvial plot (also known as a Sankey diagram) to show changes in location between two conditions or datasets.

### Usage

```
plotTranslocations(
  params,
  type = "alluvial",
  all = FALSE,
  fcol,
  col,
  labels = TRUE,
  labels.par = "adj",
  cex = 1,
  spacer = 4,
  ...
)
```

### Arguments

- **params**: An instance of class `bandleParams` or an instance of class `MSnSetList` of length 2.
- **type**: A character specifying the type of visualisation to plot. One of "alluvial" (default) or "chord".
all  
A logical specifying whether to count all proteins or only show those that have changed in location between conditions. Default is FALSE.

col  
A list of colours to define the classes in the data. If not defined then the default pRoloc colours in getStockCol() are used.

fcoll  
If params is a list of MSnSets. Then fcol must be defined. This is a character vector of length 2 to set different labels for each dataset. If only one label is specified, and the character is of length 1 then this single label will be used to identify the annotation column in both datasets.

labels  
Logical indicating whether to display class/organelle labels for the chord segments or alluvial stratum. Default is TRUE.

labels.par  
If type is "alluvial". Label style can be specified as one of "adj", "repel". Default is "adj".

cex  
Text size. Default is 1.

spacerr  
A numeric. Default is 4. Controls the white space around the circos plotting region.

...  
Additional arguments passed to the chordDiagram function.

Value

Returns a directional circos/chord diagram showing the translocation of proteins between conditions. If type = "alluvial" output is a ggplot object.

Examples

```r
## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 4L,
  numDyn = 100L)
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]

## fit GP params
gpParams <- lapply(tansim$lopitrep, function(x)
  fitGPmaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))

## run bandle
res <- bundle(objectCond1 = control,
  objectCond2 = treatment,
  gpParams = gpParams,
  fcol = "markers",
  numIter = 5L,
  burnin = 1L,
  thin = 2L,
  numChains = 1,
  BPPARAM = SerialParam(RNGseed = 1),
```

```r
```
proteinAllocation

```
seed = 1)

## Process the results
bandleres <- bandleProcess(res)

## plot the results
plotTranslocations(bandleres)
plotTranslocations(bandleres, type = "chord")
```

---

**proteinAllocation**  
*sample allocations, probabilities and compute log likelihoods*

---

**Description**

Internal sampling function, not for outside use documented for completeness

**Usage**

```r
proteinAllocation(loglikelihoods, currentweights, alloctemp, cond)
```

```r
outlierAllocationProbs(
  outlierlikelihood,
  loglikelihoods,
  epsilon,
  alloctemp,
  cond
)
```

```r
sampleOutlier(allocoutlierprob)
```

```r
covOrganelle(object, fcol = "markers")
```

```r
pg_prior(object_cond1, object_cond2, K, pgPrior = NULL, fcol = "markers")
```

```r
sample_weights_pg(nk_mat, pgPrior, w, K, tau = 0.2)
```

```r
sample_weights_dir(nk_mat, dirPrior)
```

**Arguments**

- `loglikelihoods`: the log likelihoods
- `currentweights`: the current allocations weights
- `alloctemp`: the current protein allocations
- `cond`: the control = 1, treatment = 2
- `outlierlikelihood`: the outlier log likelihoods
epsilon          the outlier component weight
allocoutlierprob the outlier probabilities
object           An instance of class MSnSet
fcol             The feature column containing the markers.
object_cond1     A list of instance of class MSnSets usually control
object_cond2     A list of instance of class MSnSets usually treatment
K                The number of organelle classes
pgPrior          The Polya-Gamma prior
nk_mat           The summary matrix of allocations
w                The Polya-Gamma auxiliary variable
tau              The empirical bayes parameter for the Polya-Gamma variable. Defaults to 0.2.
dirPrior         The Dirichlet prior

Value

returns samples for protein allocations, log likelihoods and probabilities
returns outlier probabilities
returns outlier allocations
returns covariance of organelles using marker proteins
returns the Polya-Gamma prior
returns A sample of the weights using Polya-Gamma priors.
returns A sample of the weights using Dirichlet prior.

Examples

library(pRolocdata)
data("tan2009r1")
covOrganelle(object = tan2009r1)

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
                      numRep = 6L,
                      numDyn = 100L)
d1 <- tansim$blopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
out <- pg_prior(object_cond1 = control1, object_cond2 = treatment1, K = 11)
robustMahalanobis

robust Mahalanobis distance

Description

These function implement the MR method of Itzhak et al.

Usage

robustMahalanobis(delta)

reprodScore(x, y, method = c("pearson"))

mrMethod(objectCond1, objectCond2, method = "2017")

Arguments

delta The difference profile to compute the squared mahalanobis distance
x Numeric vector to compute reproducibility score
y Numeric vector to compute reproducibility score
method Correlation method. Default is Pearson
objectCond1 A list of MSnbase::MSnSets where each is an experimental replicate for the first condition, usually a control
objectCond2 A list of MSnbase::MSnSets where each is an experimental replicate for the second condition, usually a treatment

Value

The squared Mahalanobis distance
The R score
The MR score of the Ithzak et al. 2016/2017

Examples

## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 4L,
  numDyn = 100L)
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]

## compute delta matrix
deltaMatrix <- exprs(control[[1]]) - exprs(treatment[[1]])
res <- bandle:::robustMahalanobis(deltaMatrix)
## @examples
## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 4L,
  numDyn = 100L)
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]
## compute delta matrix
deltaMatrix1 <- exprs(control[[1]]) - exprs(treatment[[1]])
deltaMatrix2 <- exprs(control[[2]]) - exprs(treatment[[2]])
mr_score <- bandle:::reprodScore(deltaMatrix1, deltaMatrix2)
library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 6L,
  numDyn = 100L)
d1 <- tansim$lopitrep
control1 <- d1[1:3]
treatment1 <- d1[4:6]
mr1 <- mrMethod(objectCond1 = control1, objectCond2 = treatment1)
plot(mr1$Mscore, mr1$Rscore, pch = 21,
  xlab = "MScore", ylab = "RScore")

---

**sim_dynamic**

*Generate a dynamic spatial proteomics experiment*

**Description**

A function to simulate dynamic spatial proteomics data using a bootstrap method

**Usage**

```r
sim_dynamic(
  object,
  subsample = NULL,
  knn_par = 10L,
  fcol = "markers",
  numRep = 6L,
  method = "wild",
  batch = FALSE,
  frac_perm = FALSE,
)```

spatial2D

nu = 2, numDyn = 20L

Arguments

object A instance of class MSnSet from which to generate a spatial proteomics dataset.

subsample how many proteins to subsample to speed up analysis. Default is NULL.

knn_par the number of nearest neighbours to use in KNN classification to simulate dataset. Default is 10

fcol feature column to indicate markers. Default is "markers". Proteins with unknown localisations must be encoded as "unknown".

numRep The total number of datasets to generate. Default is 6. An integer must be provided

method The bootstrap method to use to simulate dataset. Default is "wild". refer to BANDLE paper for more details.

batch Whether or not to include batch effects. Default is FALSE.

frac_perm whether or not to permute the fractions. Default is FALSE

nu parameter to generate residual inflated noise. Default is 2. See BANDLE paper for more details

numDyn An integer number of protein to simulate dynamic transitions. Default is 20

Value

returns simulate dynamic lopit datasets and the name of the relocalated protein.

Examples

library(pRolocdata)
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1, numRep = 6L, numDyn = 100L)

spatial2D

Generate a PCA plot with smoothed probability contours

Description

Generate a PCA plot with smoothed probability contours
Usage

```r
spatial2D(
  object,
  params,
  fcol = "markers",
  dims = c(1, 2),
  cov.function = NULL,
  theta = 2,
  derivative = 2,
  k = 1,
  cond = 1,
  n = 1,
  breaks = c(0.99, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7),
  aspect = 0.5
)
```

Arguments

- **object**: An instance of class `MSnSet` to provide the pca coordinates
- **params**: An instance of class `bandleParams`
- **fcol**: Feature columns that defines the markers. Defaults to "markers".
- **dims**: The PCA dimensions to plot. Defaults to `c(1,2)`
- **cov.function**: The covariance function for the smoothing kernel. Defaults to `wendland.cov`
- **theta**: The theta parameter of the `wendland.cov`. Defaults to 2.
- **derivative**: The derivative paramter of the `wendland.cov`. Defaults to 2.
- **k**: The k parameter of the `wendland.cov`
- **cond**: Which conditions do we want to plot. Must be 1 or 2. Default is 1
- **n**: The chain from which we plot the probability distribution. Default is 1.
- **breaks**: The levels at which to plot the contours. Defaults to `c(0.99, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7)`
- **aspect**: The aspect ratio of the pca plots. Defaults to 0.5.

Value

returns a named vector of differential localisation probabilities

Examples

```r
## Not run:
## Generate some example data
library("pRolocdata")
data("tan2009r1")
set.seed(1)
tansim <- sim_dynamic(object = tan2009r1,
  numRep = 4L,
  numDyn = 100L)
```
data <- tansim$lopitrep
control <- data[1:2]
treatment <- data[3:4]

## fit GP params
gpParams <- lapply(tansim$lopitrep, function(x)
                   fitGPhaternPC(x, hyppar = matrix(c(0.5, 1, 100), nrow = 1)))

## run bandle
res <- bandle(objectCond1 = control,
               objectCond2 = treatment,
               gpParams = gpParams,
               fcol = "markers",
               numIter = 5L,
               burnin = 1L,
               thin = 2L,
               numChains = 1,
               BPPARAM = SerialParam(RNGseed = 1),
               seed = 1)

## Process the results
bandleres <- bandleProcess(res)

## plot the results
spatial2D(control[[1]], bandleres)

## End(Not run)

---

StatStratum inherits StatSratum

**Description**

inherits StatSratum

**Usage**

StatStratum

**Format**

An object of class StatStratum (inherits from Stat, ggproto, gg) of length 5.
Index

* datasets
  StatStratum, 37
* package
  bundle-package, 2
  .bundleChain (bundleChains-class), 6
  .bundleChains (bundleChains-class), 6
  .bundleParams (bundleChains-class), 6
  .bundleSummaries (bundleChains-class), 6
  .bundleSummary (bundleChains-class), 6
  gpParams (gpParams-class), 20
  .nicheParam (bundleChains-class), 6
  .nicheParams (bundleChains-class), 6
  [,bundleChains,ANY,ANY,ANY-method (bundleChains-class), 6
  [,bundleParams,ANY,ANY,ANY-method (bundleChains-class), 6
  [,bundleSummaries,ANY,ANY,ANY-method (bundleChains-class), 6
  [,nicheParams,ANY,ANY,ANY-method (bundleChains-class), 6
  [[,bundleChains,ANY,ANY-method (bundleChains-class), 6
  [[,bundleParams,ANY,ANY-method (bundleChains-class), 6
  [[,bundleSummaries,ANY,ANY-method (bundleChains-class), 6
  [[,nicheParams,ANY,ANY-method (bundleChains-class), 6
  bundle, 3
  bundle(), 10
  bundle-package, 2
  bundleChain-class (bundleChains-class), 6
  bundleChains-class, 6
  bundleJoint (bundleChains-class), 6
  bundleJoint, bundleSummary-method (bundleChains-class), 6
  bundleParams-class (bundleChains-class), 6
  bundlePredict, 10
  bundleProcess, 11
  bundleSummaries-class (bundleChains-class), 6
  bundleSummary-class (bundleChains-class), 6
  besselK (besselK_boost), 12
  besselK_boost, 12
  binomialDiffLocProb (diffLocalisationProb), 15
  bootstrapdiffLocprob (diffLocalisationProb), 15
  centeredData (besselK_boost), 12
  centeredDatamatern (besselK_boost), 12
  chains (bundleChains-class), 6
  comploglike (besselK_boost), 12
  comploglikelist (besselK_boost), 12
  componentloglike (besselK_boost), 12
  covOrganelle (proteinAllocation), 31
  diffLoc (bundle), 3
  diffLocalisationProb, 15
  dmvtCpp (besselK_boost), 12
  dmvtInt (besselK_boost), 12
  EFDR, 17
  fitGP, 18
  fitGPmatern (fitGP), 18
  fitGPmaternPC (fitGP), 18
  gradientamatern (besselK_boost), 12
  gradientGP, 21
  gradientGPcpp (besselK_boost), 12
  gradientGPcppmatern (besselK_boost), 12
  gradientGPmatern (gradientGP), 21
  gradientlogprior (gradientGP), 21
  gradientrhomatern (besselK_boost), 12
  Gumbel (gradientGP), 21