Package 'sppmix'

October 14, 2020

Title Modeling Spatial Poisson and Related Point Processes

Version 1.0.2.1

Author Jiaxun Chen <chenjiaxun9@hotmail.com>
and Athanasios (Sakis) Christou Micheas <micheasa@missouri.edu>.
Significant contributions on the package skeleton creation, plotting functions and other code by Yuchen Wang <ycwang0712@gmail.com>

Maintainer Sakis Micheas <micheasa@missouri.edu>

Description
Implements classes and methods for modeling spatial point patterns using inhomogeneous Poisson point processes, where the intensity surface is assumed to be analogous to a finite additive mixture of normal components and the number of components is a finite, fixed or random integer. Extensions to the marked inhomogeneous Poisson point processes case are also presented. We provide an extensive suite of R functions that can be used to simulate, visualize and model point patterns, estimate the parameters of the models, assess convergence of the algorithms and perform model selection and checking in the proposed modeling context.

Depends R (>= 3.2.1), stats, graphics, grDevices, utils

License GPL (>= 2)

LazyData true

Imports Rcpp, spatstat, rgl, fields, mvtnorm, ggplot2 (>= 2.1.0)

LinkingTo Rcpp, RcppArmadillo, mvtnorm

URL http://faculty.missouri.edu/~micheasa/sppmix/index.html

RoxygenNote 6.0.1

Suggests knitr,rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-14 16:34:52 UTC

R topics documented:

add_title . . . 3
approx_normmix . . . 4
check_labels . . . 5
CompareSurfs . . . 7
Count_pts . . . 8
Datasets . . . 10
demo_mix . . . 12
Demo_sppmix . . . 13
dnormmix . . . 14
drop_realization . . . 15
est_intensity_np . . . 16
est_MIPPP_cond_loc . . . 17
est_MIPPP_cond_mark . . . 21
est_mix_damcmc . . . 24
FixLS_da . . . 28
GetBDCompfit . . . 30
GetBDTable . . . 31
GetBMA . . . 32
GetDensityValues . . . 33
GetIPPPLikValue . . . 34
GetKLEst . . . 35
GetMAPEst . . . 37
GetMAPLabels . . . 38
GetPMEst . . . 39
GetStats . . . 40
Get_Rdiag . . . 41
kstest2d . . . 42
kstest2dsurf . . . 44
MaternCov . . . 45
mc_gof . . . 46
normmix . . . 47
openwin_sppmix . . . 49
plot.bdmcmc_res . . . 50
plot.damcmc_res . . . 52
plot.intensity_surface . . . 53
plot.normmix . . . 54
plot.sppmix . . . 55
plot2dPP . . . 57
plotmix_2d . . . 58
plotmix_3d . . . 59
plotstring . . . 61
Plots_off . . . 61
PlotUSAStates . . . 62
plot_autocorr . . . 65
plot_avgsurf . . . 67
plot_chains . . . 68

plot_CompDist 70
plot_convdiags 71
plot_density 72
plot_ind 73
plot_MPP_fields 74
plot_MPP_probs 75
plot_runmean 76
plot_true_labels 77
rGRF 78
rMIPPP_cond_loc 79
rMIPPP_cond_mark 83
rmixsurf 85
rnormmix 86
rsppmix 88
Save_AllOpenRglGraphs 90
selectMix 91
sppmix 93
summary.bdmcmc_res 96
summary.damcmc_res 97
to_int_surf 98

Index **100**

`add_title` *Add a title to an existing ggplot2 plot*

Description

The function adds extra titles to a ggplot2 plot.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#add_title

Usage

```
add_title(title, lambda = "", m = "", n = "", t = "", L = "",
  nmarks = "", mu = "", theta = "", nu = "", gamma = "", sigma = "",
  df = "")
```

Arguments

`title`	The title to use for the plot.
`lambda, m, n, t, L, nmarks, mu, theta, nu, gamma, sigma, df`	Optional info to display on the second row of the title: average number of points, number of components in the mixture, number of points in the point pattern, time frame, number of iterations, total number of marks, a mean value, parameters for a Matern covariance model, a parameter gamma, and degrees of freedom, respectively.

Author(s)

Sakis Micheas, Yuchen Wang

See Also

rnormmix, dnormmix, MaternCov

Examples

```
truemix = rnormmix(m = 5, sig0 = .1, df = 5,xlim= c(0, 3), ylim = c(0, 3))
intsurf=to_int_surf(truemix,lambda = 100,win=spatstat::owin( c(0, 5),c(0, 5)))
#plot the intensity surface
plotmix_2d(intsurf)+add_title("A pretty projection of the 3d surface to 2 dimensions")
```

---

approx_normmix *Approximate the masses of bivariate normal mixture components*

Description

Calculates the mass of the density of each component of a normal mixture over a given window.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#approx_normmix

Usage

```
approx_normmix(mix, xlim = c(0, 1), ylim = c(0, 1))
```

Arguments

mix	An object of class normmix
xlim, ylim	Vectors defining the x-y integration limits. A mixture component mass is estimated within this window.

Value

A numerical vector with elements corresponding to the mass of each component within the window. These values are required when we use truncation over an observation window in order to handle edge effects.

Author(s)

Jiaxun Chen, Yuchen Wang

See Also

normmix

Examples

```
truemix = normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2),
 c(.8, .8)), sigmas =list(.01*diag(2),.01*diag(2)))
approx_normmix(truemix,xlim= c(-2, 2), ylim = c(-2, 2))
```

`check_labels` *Check for label switching*

Description

Checks if there is label switching present in the posterior realizations using the chains for mu. The algorithm is heuristic and works as follows; for each chain of mu for a given component, we look for sharp changes in the generated values that lead to dramatically different variability from the variability observed in the past history of the chain. The lag history is 5% of the total number of realizations, excluding the burnin realizations.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#check_labels

Usage

```
check_labels(fit, burnin = floor(fit$L/10), lagnum = floor(0.05 * (fit$L -
  burnin) + 1), showmessages = TRUE)
```

Arguments

`fit`	Object of class `damcmc_res` or `bdmcmc_res`.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
`lagnum`	Number of past realizations to use for the detection algorithm. Default is .05 of the total realizations after burnin realizations are removed.
`showmessages`	Logical variable requesting to show informative messages (TRUE) or suppress them (FALSE). Default is TRUE.

Details

To avoid the label switching problem one can plot the average of the intensities of the posterior realizations, i.e., the average of the surfaces instead of the surface of the posterior averages. However, by doing so we lose the ability to perform mixture deconvolution, namely, work with the posterior means of the ps, mus and sigmas of the mixture intensity. In general, the average of posterior surfaces for each realization of the ps, mus and sigmas, and the surface based on the posterior means of the ps, mus and sigmas, need not be the same.

For a DAMCMC fit, avoiding label switching can be accomplished using function `plot_avgsurf`, which plots the average posterior surface. The surface of posterior means is plotted using the function `plot.damcmc_res` on the returned value of `GetPMEst`.

When working with a BDMCMC fit, it is recommended that you use GetBDCompfit to retrieve the realizations corresponding to a specific number of components and then fix the labels. Of course the best estimate in this case, is the Bayesian model average intensity obtained via GetBMA, but it can be very slow to compute. The BMA is an average on surfaces based on each of the posterior realizations, and as such does not suffer from the label switching problem.

Value

Logical value indicating if label switching was detected.

Author(s)

Jiaxun Chen, Sakis Micheas

References

Jasra, A., Holmes, C.C. and Stephens, D. A. (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixtures. Statistical Science, 20, 50-67.

See Also

normmix, rsppmix, est_mix_damcmc, plot_chains, FixLS_da

Examples

```
# generate data
mix1 = normmix(ps=c(.4, .2,.4), mus=list(c(0.3, 0.3), c(.5,.5),c(0.7, 0.7)),
 sigmas = list(.02*diag(2),.05*diag(2), .02*diag(2)),lambda = 100,
 win = spatstat::square(1))
plot(mix1,main="True Poisson intensity surface (mixture of normal components)")
pp1 = rsppmix(mix1)
# Run Data augmentation MCMC and get posterior realizations
postfit = est_mix_damcmc(pp1,m=5)
#plot the chains
plot_chains(postfit)
#check labels
check_labels(postfit)
#plot the average posterior surface
plot(GetPMEst(postfit))
#plot the surface of posterior means, can be slow for large LL
avgsurf=plot_avgsurf(postfit, LL = 50, burnin = 1000)
# Fix label switching, start with approx=TRUE
post_fixed = FixLS_da(postfit, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)
#this one works better in theory
post_fixed = FixLS_da(postfit,approx=FALSE, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)
```

CompareSurfs *Quantify the difference between two surfaces*

Description

This function can be used to compare two intensity surfaces. In particular, if we have the true surface and an estimator of the truth, then we can assess how well the estimate fits the surface, i.e., how close are the two surfaces. Now if we have two estimates of the true surface then the estimate with the smallest measure fits the truth better. We can also compare two estimating surfaces this way.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#CompareSurfs

Usage

```
CompareSurfs(surf1, surf2, LL = 100, truncate = FALSE)
```

Arguments

surf1, surf2	Either IPPP intensity surfaces (objects of type intensity_surface) or images (objects of type im) that represent intensity surfaces.
LL	Length of the side of the square grid. The intensities are calculated on an L * L grid. The larger this value is, the slower the calculation, but the better the approximation to the difference between the two surfaces.
truncate	Requests to truncate the components of the mixture intensities to have all their mass within the observation window.

Details

Since the two surfaces passed to the function can be represented as a 2d intensity surface, any measure between two images can be used for comparison purposes, provided that the window is the same.

If the two windows are different the function will choose the largest one and compare the two surfaces in there.

Value

Returns a list containing all distances computed and the window of comparison, an object of class owin.

Author(s)

Sakis Micheas

See Also

rmixsurf, owin, rsppmix, est_mix_bdmcmc, drop_realization, plotmix_3d, plot_avgsurf, GetBMA,

Examples

```
#compare two surfaces first
mixsurf1 = rmixsurf(m = 5, sig0 = .1, df = 5,xlim= c(-1,4), ylim = c(-2,1), rand_m = FALSE)
mixsurf2 = rmixsurf(m = 8, sig0 = .1, df = 5,xlim= c(-4,3), ylim = c(-1,2), rand_m = FALSE)
comp=CompareSurfs(mixsurf1,mixsurf2)
plot(mixsurf1,main = "First IPPP",win=comp$win)
plot(mixsurf2,main = "Second IPPP",win=comp$win)
# now we compare intensity surfaces and image objects that represent intensity surfaces
truemixsurf = rmixsurf(m = 5,xlim= c(-2,2), ylim = c(-2,2))
plot(truemixsurf,main="True IPPP surface")
#get a point pattern
genpp = rsppmix(truemixsurf,truncate=FALSE)
# Run BDMCMC and get posterior realizations
postfit=est_mix_bdmcmc(genpp,m=10,L=30000)
postfit=drop_realization(postfit,.1*postfit$L) #apply burnin
BMA=GetBMA(postfit,burnin=0)
title1 = paste("Bayesian model average of",postfit$L,"posterior realizations")
plotmix_3d(BMA,title1=title1)
comp=CompareSurfs(truemixsurf,BMA,LL=100)
#We compare the average surface and the truth for many cases below. If we pass images, we
#make sure it has the same dimensions or we force it to the same value by setting LL=100.
#We retrieve the average surfaces corresponding to MAP-1, MAP and MAP+1 components and
#compare them against the truth.
#First retrieve the frequency table and MAP estimate for number of components
BDtab=GetBDTable(postfit)
BDtab
MAPm=BDtab$MAPcomp
BDfitMAPcomp_minus1=GetBDCompfit(postfit,MAPm-1,burnin=0)
avgsurfMAPcomp_minus1=plot_avgsurf(BDfitMAPcomp_minus1$BDgens, LL = 100,burnin=0)
comp=CompareSurfs(truemixsurf,avgsurfMAPcomp_minus1,LL=100)
BDfitMAPcomp=GetBDCompfit(postfit,MAPm,burnin=0)
avgsurfMAPcomp=plot_avgsurf(BDfitMAPcomp$BDgens, LL = 100,burnin=0)
comp=CompareSurfs(truemixsurf,avgsurfMAPcomp,LL=100)
BDfitMAPcomp_plus1=GetBDCompfit(postfit,MAPm+1,burnin=0)
avgsurfMAPcomp_plus1=plot_avgsurf(BDfitMAPcomp_plus1$BDgens, LL = 100,burnin=0)
comp=CompareSurfs(truemixsurf,avgsurfMAPcomp_plus1,LL=100)
```

Count_pts *Counts points in a window*

Description

This function counts the number of points from a point pattern within a specified window.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Count_pts

Usage

```
Count_pts(pp, win)
```

Arguments

pp	Object of class sppmix or ppp.
win	The window of observation as an object of class owin, defining the window of observation.

Value

An integer representing the number of points from pp within the window win.

Author(s)

Sakis Micheas

See Also

rsppmix, rmixsurf, plotmix_2d, square, owin

Examples

```
truemix_surf=rmixsurf(m = 3,lambda=100, xlim=c(-5,5),ylim=c(-5,5))
genPP=rsppmix(truemix_surf)
plotmix_2d(truemix_surf,genPP)
Count_pts(genPP,spatstat::square(1))
Count_pts(genPP,spatstat::square(2))
Count_pts(genPP,spatstat::square(3))
Count_pts(genPP,spatstat::square(4))
Count_pts(genPP,spatstat::square(5))
Count_pts(genPP,spatstat::owin(c(-5,5),c(-5,5)))
```

Datasets *Datasets*

Description

Several datasets have been included in the sppmix package. They are all open source datasets that have been processed into ppp objects. Many examples and tutorials use these datasets.

For basic examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Datasets

ChicagoCrime2015

The 2015 Chicago crime dataset (http://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2) contains the reported locations of homicide cases in Chicago during the year 2015. The sppmix package object ChicagoCrime2015 is a marked point pattern containing the crime locations for two types of crimes; kidnappings coded as 0, and homicides coded as 1. In particular, there are 654 reported incident locations, 188 kidnappings, and 466 homicides.

ChicagoArea

The ChicagoArea object contains the coordinates for the different neighborhoods of the city of Chicago.

CAQuakes2014

The Southern California Earthquake Data Center (SCEDC, Southern California Earthquake Center, Caltech, Dataset: doi:10.7909/C3WD3xH1), operates at the Seismological Laboratory at Caltech and is the primary archive of seismological data for southern California, recording the seismological activity from 1932 to present. Here we concentrate on the year 2014, and all this data is contained in a marked point process object named CAQuakes2014. There are two continuous variables containing marks for the events, including variable mag which represents the magnitude of the earthquake and variable depth representing the depth.

CAQuakes2014.RichterOver3.0

In many examples we further restrict to events with magnitudes over 3.0 in the Richter scale. The latter marked point pattern is aplty named CAQuakes2014.RichterOver3.0.

TornadoesAll

The National Oceanic and Atmospheric Administration (NOAA, http://www.noaa.gov/) is a U.S.A. agency tasked with the dissemination of daily weather forecasts and severe storm warnings, as well as, conducting climate monitoring among many other important tasks. The Storm Prediction Center of NOAA contains important information on tornado occurrences throughout the U.S., starting from 1950 all the way to the present. All this information is contained in the data.frame object TornadoesAll. The variables (columns) included are as follows: 1="RecordNumber", 2="Year", 3="Month", 4="Day", 5="Date:yyyy-mm-dd", 6="Time:HH:MM:SS", 7="State", 8="Fscale", 9="Injuries" 10="Fatalities", 11="Estimated property loss", 12="Estimated crop loss", 13="Starting latitude", 14="Starting longitude", 15="Length in miles", and 16="Width in yards".

Note that each event (row) is marked using one of 6 levels (variable Fscale), each denoting the strength of a tornado in the Fujita scale (Enhanced Fujita scale after January, 2007), with 0 denoting minimal damage and 5 indicating complete destruction.

`Tornadoes2011MO`

In many examples we use the marked point pattern for year 2011 (a single time snap-shot), contained in the object `Tornadoes2011MO`, in order to study the behavior of the events of that year; this year is of particular interest since on Sunday, May 22, 2011, Joplin Missouri, USA, was struck by a destructive tornado resulting in over 150 deaths and $2.8 billion in damages.

`ContinentalUSA_state_names`

In several examples we use the names of the USA states. This data is contained in the object `ContinentalUSA_state_names`.

`MOAggIncomeLevelsPerCounty`

In some examples we use the aggregate income levels per county for the state of Missouri, USA. This data is contained in the object `MOAggIncomeLevelsPerCounty`.

`USAStatesCounties2016`

In many examples we use the boundaries of the states and counties of the USA. The Cartographic Boundary Shapefiles (boundary data) is provided by the USA Census Bureau at https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html. This is a list containing `StateNames` (the state and territory names, 56 total), `StatePolygons` (a list of size 56, containing a list of matrices describing the boundaries of the states/territories), and `CountiesbyState` (a list of size 56, with each list element a list containing element `CountyName` and `CountyPolies`, describing the county name of the specific state and polygons used). For example, USAStatesCounties2016$StateNames[1] is Alabama, and USAStatesCounties2016$CountiesbyState[[1]]$CountyName[1] corresponds to Escambia county which can be plotted using the boundary coordinates in USAStatesCounties2016$CountiesbyState[[1]]$CountyPolies[[1]].

Usage

```
Datasets()

ChicagoCrime2015

ChicagoArea

CAQuakes2014

CAQuakes2014.RichterOver3.0

TornadoesAll

Tornadoes2011MO

ContinentalUSA_state_names

MOAggIncomeLevelsPerCounty

USAStatesCounties2016
```

Format

All datasets are objects of type ppp and sppmix, except for object USAStatesCounties2016.

Author(s)

Sakis Micheas

Examples

```
ChicagoCrime2015
summary(ChicagoCrime2015)
plot(ChicagoCrime2015)+add_title("Chicago Crime, 2015")
CAQuakes2014.RichterOver3.0
summary(CAQuakes2014.RichterOver3.0)
plot(CAQuakes2014.RichterOver3.0)+add_title("Earthquakes in California, 2014")
Tornadoes2011MO
summary(Tornadoes2011MO)
plot(Tornadoes2011MO)+add_title("Tornado events about Missouri, 2011")
```

---

demo_mix *Demo objects*

Description

Demo objects (mixture, surface and generated pattern) using the classes provided by the sppmix package.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#demo_mix

Usage

```
demo_mix

demo_intsurf

demo_genPPP

demo_truemix3

demo_truemix3comp

demo_truesurf3

demo_intsurf3comp
```

Format

An object of class normmix of length 5.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

Examples

```
demo_mix <- normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2), c(.8, .8)),
 sigmas = list(.01*diag(2), .01*diag(2)))
demo_intsurf <- normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2),
 c(.8, .8)),sigmas = list(.01*diag(2), .01*diag(2)), lambda = 100,
 win = spatstat::square(1))
demo_genPPP<-rsppmix(demo_truesurf3, truncate=FALSE)
```

Demo_sppmix *Run an sppmix package demo or vignette*

Description

The function starts the different tutorials of the sppmix package.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Demo_sppmix

Usage

```
Demo_sppmix(whichdemo = NULL)
```

Arguments

whichdemo	A number indicating which demo to run. Do not pass any number in order to see all the availabe choices.

Author(s)

Sakis Micheas

Examples

```
Demo_sppmix()#shows all available demos and opens the vignettes page
Demo_sppmix(1)#demo on sppmix objects
```

dnormmix *Calculate the density or intensity of a normal mixture over a fine grid*

Description

When a normmix object is given, this function calculates the mixture density over a fine grid for the given window. When an intensity_surface object is given, the function multiplies the density with the intensity surface parameter lambda, and returns the Poisson mixture intensity function over the grid. Used for plotting.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#dnormmix

Usage

```
dnormmix(mix, xlim = c(0, 1), ylim = c(0, 1), L = 128, truncate = TRUE)
```

Arguments

mix	An object of class normmix or intensity_surface
xlim, ylim	Vectors defining the x-y integration limits. A mixture component mass is estimated within this window.
L	Length of the side of the square grid. The density or intensity is calculated on an L * L grid. The larger this value is, the slower the calculation, but the better the approximation.
truncate	Requests to truncate the components of the mixture intensity to have all their mass within the given x-y limits.

Value

An object of class im. This is a pixel image on a grid with values corresponding to the density (or intensity surface) at that location.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, rnormmix, plotmix_2d, plot_density, to_int_surf

Examples

```
truemix <- rnormmix(m = 3, sig0 = .1, df = 5,xlim= c(0, 5),
 ylim = c(0, 5))
normdens=dnormmix(truemix,xlim= c(0, 5), ylim = c(0, 5))
#2d plots
plot_density(as.data.frame(normdens))+ ggplot2::ggtitle(
 "2d mixture density plot\nWindow=[0,5]x[0,5]")
plot_density(as.data.frame(normdens),TRUE)+ ggplot2::ggtitle(
 "2d mixture contour plot\nWindow=[0,5]x[0,5]")
#3d plot
plotmix_3d(normdens)
#Now build an intensity surface based on the normal mixture
intsurf=to_int_surf(truemix,lambda = 100, win =
 spatstat::owin( c(0, 5),c(0, 5)))
intsurfdens=dnormmix(intsurf,xlim= c(0, 5), ylim = c(0, 5))
plot_density(as.data.frame(intsurfdens))+ ggplot2::ggtitle(
 "2d mixture intensity plot\nWindow=[0,5]x[0,5]")
plot_density(as.data.frame(intsurfdens),TRUE)+ ggplot2::ggtitle(
 "2d mixture intensity contour plot\nWindow=[0,5]x[0,5]")
plotmix_3d(normdens)#3d plot
#For an intensity surface object we use these functions instead
plotmix_2d(intsurf)
plot(intsurf)
```

`drop_realization` *Drop MCMC realizations*

Description

The function drops realizations from a DAMCMC or BDMCMC fit and returns the resulting fit object.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#drop_realization

Usage

```
drop_realization(fit, drop = 0.1 * fit$L)
```

Arguments

`fit`	Object of class `damcmc_res` or `bdmcmc_res`.
`drop`	If one integer is provided, the function will drop the first 1:drop realizations. If an integer vector is provided, it will drop these iterations. If a logical vector is provided (with the same length as the chain length of `fit`), it will be used for subsetting directly.

Author(s)

Sakis Micheas, Yuchen Wang

See Also

est_mix_bdmcmc

Examples

```
fit <- est_mix_bdmcmc(spatstat::redwood, m = 5)
fit
drop_realization(fit, 500)
drop_realization(fit, fit$numcomp != 5)
```

---

est_intensity_np *Estimate the intensity surface using a non-parametric method*

Description

Using an Epanechnikov kernel we calculate an estimate of intensity surface while accounting for edge effects.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#est_intensity_np

Usage

```
est_intensity_np(pattern, win, h, L = 10, kernel = c("Epanechnikov"),
  edgecorrect = TRUE, truncate = TRUE)
```

Arguments

pattern	A two-dimesional spatial point pattern in the form of a ppp object.
win	Object of class owin.
h	Kernel bandwidth. h should be a positive number.
L	Length of the side of the square grid.
kernel	Kernel used to estimate the intensity surface. Currently, only supports the Epanechnikov kernel.
edgecorrect	Logical flag indicating whether or not to use edge-correction in the estimating intensity surface. The default is TRUE.
truncate	Logical flag indicating whether or not to use points only within the window. The default is TRUE.

Value

An object of class im.

Author(s)

Jiaxun Chen, Sakis Micheas

See Also

rnormmix, to_int_surf, rsppmix, plotmix_3d

Examples

```
mix1 <- rnormmix(5, sig0 = .01, df = 5, xlim=c(0, 5),ylim=c(0, 5))
intsurf1=to_int_surf(mix1,lambda = 40, win =spatstat::owin( c(0, 5),c(0, 5)))
plot(intsurf1)
pp1 <- rsppmix(intsurf1)
# estimate and plot the estimated intensity surface
surfNP1 <- est_intensity_np(pp1, win=spatstat::owin( c(0, 5),c(0, 5)), h=0.05,
 L=100,truncate = FALSE)
plotmix_3d(surfNP1,title1="Non parametric estimator of the intensity surface")
#truncate components to have all their mass in the window
surfNP2 <- est_intensity_np(pp1, win=spatstat::owin( c(0, 5),c(0, 5)), h=0.5, L=100)
plotmix_3d(surfNP2,title1="(Truncated) Non parametric estimator of the intensity surface")
```

---

est_MIPPP_cond_loc *Fit a MIPPP conditionally on location*

Description

This function fits a Marked IPPP (MIPPP) on a marked point pattern by modeling the (joint) intensity surface of the locations and the marks using an IPPP for the locations (independent of the mark values) and for discrete marks a Gibbs model for the mark distribution which is conditionally defined on all the locations. NOTE: The estimation procedure for continuous marks (random fields) will be implemented in future versions of the sppmix package.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#est_MIPPP_cond_loc

Usage

```
est_MIPPP_cond_loc(pp, r, hyper = 1, L = 10000, burnin = floor(L/10),
  m = 3, fit_groundIPPP = FALSE, truncate = FALSE, grayscale = FALSE,
  startgamma, discrete_mark = TRUE, LL = 150, open_new_window = FALSE,
  show_plots = TRUE)
```

```
## S3 method for class 'MIPPP_fit'
plot(x, surf, open_new_window = FALSE,
  grayscale = FALSE,
  main = "Locations and ground intensity surface of a marked IPPP", ...)

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

Arguments

pp	Marked point pattern of class ppp.
r	Radius used to define the neighborhood system. Any two locations within this distance are considered neighbors.
hyper	Hyperparameter for the proposal distribution of gamma. This is currently the standard deviation for the random walk Metropolis-Hastings steps (one step for each gamma). Use a small value.
L	Number of iterations for the MCMC; default is 10000.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
m	The number of components to fit for the ground process when fit_groundIPPP=TRUE.
fit_groundIPPP	Logical variable requesting to fit and return the DAMCMC results of the ground process.
truncate	Logical variable indicating whether or not we we only work with events within the window defined in the point pattern pp.
grayscale	Logical to request plots in grayscale.
startgamma	Initial value for the gamma vector. If missing the zero vector is used.
discrete_mark	Logical flag indicating whether the mark is a discrete (numerical value) or not. For continuous marks set this to FALSE.
LL	Length of the side of the square grid.
open_new_window	Open a new window for a plot.
show_plots	Logical variable requesting to produce the probability field plots for each mark.
x	An object of class MIPPP_fit (the result of a request from est_MIPPP_cond_loc).
surf	An object of type intensity_surface representing the IPPP surface for the ground process (conditioning on location only). This can be the surface of posterior means or the MAP from a damcmc_res object or the MAP number of components surface from a bdmcmc_res object.
main	Main title for the plot.
...	Additional arguments for the S3 method.
object	An object of class MIPPP_fit (the result of a request from est_MIPPP_cond_loc).

Details

We assume that the joint distribution of a marked point pattern N=[s,m(s)] with n events is of the form:

p(N)=lambda^n*exp(-lambda)/(n!)*f(all s|theta1)*g(all m|theta2(s),all s)

where s denotes a location and m=m(s) a mark value at that location, lambda a parameter with the interpretation as the average number of points over the window of observation, and f, g are proper densities.

The location (or ground process) f(all s|theta1) can be fit using any method for unmarked point patterns (for us it is modeled using an IPPP with a mixture of normals intensity surface). The function fits the parameters of the second part of this model by default. However, setting fit_groundIPPP=TRUE will fit a mixture intensity surface for the ground process and return it for future processing. Alternatively, simply retrieve the marked point process returned and fit the ground process using est_mix_damcmc or est_mix_bdmcmc (the marks will be ignored and only the locations will be used).

Since s is observed over some window and the marks are conditioned on knowing the locations, then g is a random field for each value of m.

The neighborhood system is controlled by r and is crucial in this case. Small values tend to produce probability fields with concentrated masses about observed events of the process, whereas, large neighborhoods allow us to borrow strength across locations and result in much smoother probability fields. This parameter is currently NOT estimated by the sppmix package, but will be implemented in future releases.

See Micheas (2014) for more details on special cases (2 marks) of these Marked IPPP models via conditioning arguments.

Value

An object of class MIPPP_fit, which is simply a list containing the following components:

gen_gammas	Posterior realizations of the gammas.
prob_fields	Probability fields of marks.
discrete_mark	Same logical flag as the input argument.
r	Same as the input argument.
pp	Same as the input argument.
ground_fit	An object of type damcmc_res which contains the results of a DAMCMC fit to the ground process. If fit_groundIPPP=FALSE this is NULL.
condition_on_loc	Logical variable indicating the type of conditioning used in order to produce this MIPPP fit. For this function it is set to TRUE.

Author(s)

Sakis Micheas, Jiaxun Chen

References

Hierarchical Bayesian Modeling of Marked Non-Homogeneous Poisson Processes with finite mixtures and inclusion of covariate information. Micheas, A.C. (2014). Journal of Applied Statistics, 41, 12, 2596-2615, DOI: 10.1080/02664763.2014.922167.

See Also

GetStats, rMIPPP_cond_loc

Examples

```
# Create a marked point pattern
x <- runif(100)
y <- runif(100)
#mark distribution is discrete uniform
m <- sample(1:2, 100, replace=TRUE)
m <- factor(m, levels=1:2)
pp <- spatstat::ppp(x, y, c(0,1), c(0,1), marks=m)
# estimate the probability fields for each mark; since we have a discrete
# uniform for the mark distribution we should see probabilities about .5
# for both marks, as well as, the gamma credible sets should include 0,
# meaning that the marks are independent of location (probability .5 for
# each of the two mark values)
mpp_est <- est_MIPPP_cond_loc(pp, 0.1, hyper=0.2)
GetStats(mpp_est$gen_gammas[,1])$CredibleSet
GetStats(mpp_est$gen_gammas[,2])$CredibleSet
mpp_est <-est_MIPPP_cond_loc(pp, 0.3, hyper=0.2)
GetStats(mpp_est$gen_gammas[,1])$CredibleSet
GetStats(mpp_est$gen_gammas[,2])$CredibleSet
mpp_est <- est_MIPPP_cond_loc(pp, 0.5, hyper=0.2)
GetStats(mpp_est$gen_gammas[,1])$CredibleSet
GetStats(mpp_est$gen_gammas[,2])$CredibleSet
#Visualize the Tornado data about MO. We request to fit both the mark
#and ground process.
#plot the states, the tornado locations and the marks (strength of a tornado)
ret=PlotUSAStates(states=c('Iowa','Arkansas', 'Missouri','Illinois','Indiana',
 'Kentucky','Tennessee', 'Kansas','Nebraska','Texas','Oklahoma', 'Mississippi',
 'Alabama','Louisiana'), showcentroids=FALSE,shownames=TRUE, plotlevels = FALSE,
 main= "Tornadoes about MO, 2011")
#check out the mark values and their frequency
table(Tornadoes2011MO$marks)
#plot each point with a different shape according to its marks
ret$p+ggplot2::geom_point(data=as.data.frame( Tornadoes2011MO),ggplot2::aes(x=x,
 y=y,shape= as.factor(marks)))+ggplot2::guides(shape = ggplot2::guide_legend(
 title="Tornado power", ncol=2,byrow=TRUE))
#plot each point with a different color according to its marks
ret$p+ggplot2::geom_point(data=as.data.frame( Tornadoes2011MO),ggplot2::aes(x=x,
 y=y,color= as.factor(marks)))+ggplot2::guides(color = ggplot2::guide_legend(
 title="Tornado power", ncol=2,byrow=TRUE))
#plot each point with a different circle size according to its marks
ret$p+ggplot2::geom_point(data=as.data.frame( Tornadoes2011MO),ggplot2::aes(x=x,
```

```
 y=y,size=marks),shape=21)+ ggplot2::scale_size_continuous(breaks=sort(unique(
 Tornadoes2011MO$marks))) + ggplot2::guides(size =ggplot2::guide_legend(title=
 "Tornado power", ncol=2,byrow=TRUE))
# the marks must start from 1, recode the original
Tornadoes2011MO1=Tornadoes2011MO
Tornadoes2011MO1$marks=Tornadoes2011MO1$marks+1
mpp_est=est_MIPPP_cond_loc(Tornadoes2011MO1,r=1.5,hyper=0.01,
 startgamma = c(.1,.2,.3,.4,.5,.6),fit_groundIPPP=TRUE)
#Now generate an MIPPP with 3 marks
newMPP=rMIPPP_cond_loc(gammas=c(.1,.2,.5))
summary(newMPP)
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
true_gammas=newMPP$gammas
genMPP=newMPP$genMPP
newMPP$r
mpp_est=est_MIPPP_cond_loc(genMPP,newMPP$r, hyper=0.2)
GetStats(mpp_est$gen_gammas[,1])$Mean
GetStats(mpp_est$gen_gammas[,2])$Mean
GetStats(mpp_est$gen_gammas[,3])$Mean
GetStats(mpp_est$gen_gammas[,1])$CredibleSet
GetStats(mpp_est$gen_gammas[,2])$CredibleSet
GetStats(mpp_est$gen_gammas[,3])$CredibleSet
summary(mpp_est)
plot(mpp_est)
plot(mpp_est,newMPP$surf)
```

`est_MIPPP_cond_mark` *Fit a MIPPP conditionally on mark*

Description

This function fits a Marked IPPP (MIPPP) on a marked point pattern by modeling the (joint) intensity surface of the locations and the marks using an IPPP for the marks (independent of the locations) and an IPPP with mixture intensity for the corresponding ground process, where the mixture parameters depend on the mark value. NOTE: The estimation procedure for continuous marks will be implemented in future versions of the `sppmix` package.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#est_MIPPP_cond_mark

Usage

```
est_MIPPP_cond_mark(pp, m = 10, L = 50000, burnin = floor(L/10), hyper_da,
  hyper, fit_markdist = TRUE, truncate = FALSE, grayscale = FALSE,
  discrete_mark = TRUE, LL = 256, open_new_window = FALSE,
  show_plots = TRUE, compute_surfaces = TRUE)
```

Arguments

pp	Marked point pattern of class ppp.
m	A vector representing the number of components to fit for the ground process corresponding to each mark. Since in real applications we don't know these numbers we can specify an integer so that the routine will fit a BDMCMC with this m as the maximum number of components. Then we use the MAP number of components for each ground process with a mixture intensity function of this many components. If not supplied the default is m=10.
L	Number of iterations for the MCMC; default is 50000.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
hyper_da	A list of hyperparameters for est_mix_damcmc. Each element of this list should contain 3 values (hyperparameters) and the number of elements should be the same as the number of marks. If this parameter is omitted the default hyperparameters of est_mix_damcmc will be used.
hyper	Hyperparameter for the mark distribution. Must be a vector of positive real numbers. If omitted the vector of one's is used.
fit_markdist	Logical variable requesting to fit and return the parameter estimates of the mark distribution.
truncate	Logical variable indicating whether or not we we only work with events within the window defined in the point pattern pp.
grayscale	Logical to request plots in grayscale.
discrete_mark	Logical flag indicating whether the mark is discrete or not. For continuous marks set this to FALSE.
LL	Length of the side of the square grid.
open_new_window	Open a new window for a plot.
show_plots	Logical variable requesting to produce the ground fits and probability field plots for each mark. If label switching is present, the MAPE surface is computed and returned, otherwise the PME.
compute_surfaces	Logical to request computation of the Average of Surfaces (if m is a vector) or the Bayesian Model Average (if m is an integer or missing). Default is TRUE. This is a SLOW operation.

Value

An object of class MIPPP_fit, which is simply a list containing the following components:

gen_mark_ps	The posterior realizations of the discrete mark distribution probabilities.
mark_dist	The posterior means of the discrete mark distribution probabilities.
discrete_mark	Same logical flag as the input argument.
pp	Same as the input argument.

ground_fits	A List of objects of type damcmc_res which contain the results of the DAMCMC (or the BDMCMC for MAP number of components) fits to the ground process for each discrete mark value.
ground_fitsAoS	A List of objects of type im which contain the AoS (average of surfaces) surface based on the DAMCMC (or the BMA from BDMCMC) fits to the ground process for each discrete mark value.
post_surf	A List of intensity_surface objects, one for each mark, representing the surface of posterior means, after fixing label switching using SEL permutation.
condition_on_loc	Logical variable indicating the type of conditioning used in order to produce this MIPPP fit. For this function it is set to FALSE.
fit_DAMCMC	Logical variable indicating whether or not a DAMCMC or BDMCMC fit was requested.
m	Same as input.

Author(s)

Sakis Micheas, Jiaxun Chen

See Also

rMIPPP_cond_mark, GetStats

Examples

```
#Create a marked point pattern; use randomization and 3 discrete marks
newMPP=rMIPPP_cond_mark( params=c(.2,.5,.3),bigwin = spatstat::owin(c(-10,10),c(-10,10)))
newMPP$params
#supply the true number of components for each ground process
m=c(newMPP$groundsurfs[[1]]$m, newMPP$groundsurfs[[2]]$m, newMPP$groundsurfs[[3]]$m)
MIPPPfit=est_MIPPP_cond_mark(newMPP$genMPP,m=m,compute_surfaces=FALSE)
#check out the mark distribution parameters
#posterior means
MIPPPfit$mark_dist
#credible sets
GetStats(MIPPPfit$gen_mark_ps[,1])$CredibleSet#should contain .2
GetStats(MIPPPfit$gen_mark_ps[,2])$CredibleSet#should contain .5
GetStats(MIPPPfit$gen_mark_ps[,3])$CredibleSet#should contain .3
#now pretend we do not know the truth as is usually the case. Supply an integer
#for m so that the routine will fit a BDMCMC with this as the max number of
#components and use the MAP number of components
MIPPPfit=est_MIPPP_cond_mark(newMPP$genMPP,m=7,compute_surfaces=FALSE)
#check out the mark distribution parameters
MIPPPfit$mark_dist
GetStats(MIPPPfit$gen_mark_ps[,1])$CredibleSet#should contain .2
GetStats(MIPPPfit$gen_mark_ps[,2])$CredibleSet#should contain .5
GetStats(MIPPPfit$gen_mark_ps[,3])$CredibleSet#should contain .3
```

`est_mix_damcmc` *Estimate a mixture model parameters using MCMC*

Description

These functions fit a Poisson point process with a mixture intensity, in a Bayesian framework, using the Data Augmentation MCMC (DAMCMC) method for a fixed number of components or the Birth-Death MCMC (BDMCMC) method for a random, finite number of components. Estimation of the parameters of the models is accomplished via calculation of the posterior means, based on posterior realizations obtained by these computational methods.

For DAMCMC examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#est_mix_damcmc

For BDMCMC examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#est_mix_bdmcmc

Usage

```
est_mix_damcmc(pp, m, truncate = FALSE, L = 10000, hyper_da = c(3, 1, 1),
  useKmeans = FALSE)

est_mix_bdmcmc(pp, m, truncate = FALSE, lambda1 = 1, lambda2 = 10,
  hyper = c(m/2, 1/36, 3, 2, 1, 1), L = 30000)
```

Arguments

`pp`	Point pattern object of class ppp.
`m`	Either the number of components to fit in DAMCMC or the maximum number of components requested for a BDMCMC fit.
`truncate`	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the point pattern pp.
`L`	Number of iterations for the DAMCMC (default is 10000) or BDMCMC (default is 30000). If the value passed is less than the default, then it is set to the default value.
`hyper_da`	Hyperparameters for DAMCMC, default is (3, 1, 1).
`useKmeans`	Logical variable. If TRUE use a kmeans clustering method to obtain the starting values for the component means, otherwise, randomly sample a point from the pattern and use it as a component mean.
`lambda1`	Parameter for the truncated Poisson prior; `lambda1` = 1 by default.
`lambda2`	Birth rate; `lambda2` = 10 by default.
`hyper`	Hyperparameters for the hierarchical prior. See 'Details' for more information.

Details

The DAMCMC follows the sampling scheme of Diebolt and Robert (1994).

The BDMCMC uses the sampling scheme of Stephens (2000). The definitions of the hyperparameters can be found in equations (21)-(24). The number of components entertained is from 1 to `m`, and the moves for the BDMCMC chain for the number of components, are either one up (add a component) or one down (remove a component). Make sure you plot the BDMCMC fit to obtain additional information on the fit.

Value

An object of type `damcmc_res`, containing MCMC realizations.

`allgens_List`	A list of size `L` containing posterior realizations, with elements that are lists of size `m`, with each element being the posterior realization of the parameters of a component, i.e., `p`, `mu` and `sigma`.
`genps`	An `Lxm` matrix containing the L posterior realizations of the m component probabilities of the normal mixture.
`genmus`	An `mx2xL` array containing the L posterior realizations of m component mean vectors of the normal mixture.
`gensigmas`	An `Lxm` list, with elements that are `2x2` matrices, the posterior realizations of the covariance matrices of the components of the normal mixture.
`genzs`	An `Lxn` matrix containing the membership indicators of the `n` points of the point pattern `pp`, over all iterations of the MCMCM.
`genlambdas`	An `Lx1` vector containing the posterior realizations of the of the lambda parameter (the average number of points over the window).
`genlambdas`	An `Lx1` vector containing the posterior realizations of the of the lambda parameter (the average number of points over the window).
`ApproxCompMass`	An `Lxm` matrix containing the approximate mass of the m mixture components for each iteration.
`data`	The original point pattern.
`L`	Same as input.
`m`	Same as input.

Additional return values from the BDMCMC fit (this is a `bdmcmc_res` object).

`numcomp`	An `Lx1` vector containing the number of components the BDMCMC chooses to fit at each iteration.
`maxnumcomp`	Same as input m.
`Badgen`	An `Lx1` vector with 0-1 values, where 1 indicates that the realization should be dropped, or 0 if the realization should be kept. The BDMCMC can produce "bad" births and degenerate realizations (zero component mass) that end up in the chain for a specific number of components. Although these realizations do not affect averages of surfaces (no label switching problem and the corresponding component probability is zero), they have a detrimental effect when working for mixture deconvolution within the chain of a specific number of components,

i.e., computing the posterior averages of the mixture parameters corresponding to the mixture with MAP number of components. If this parameter is 1 for some realizations, then dropping these degenerate realizations allows us to use the label switching algorithms efficiently and achieve mixture deconvolution. Obviously, the number of posterior realizations can drop significantly in number when we first apply burnin and then drop the bad realizations, so it is good practice to run the BDMCMC for at least 20000 iterations.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

References

Diebolt, J., and Robert, C. P. (1994). Estimation of Finite Mixture Distributions through Bayesian Sampling. Journal of the Royal Statistical Society B, 56, 2, 363-375.

Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. The Annals of Statistics, 28, 1, 40-74.

See Also

PlotUSAStates, plotmix_2d, GetPMEst, plot_chains, check_labels, GetBDTable, GetBDCompfit, plot.intensity_surface, plot.bdmcmc_res, to_int_surf, owin, plot_CompDist, drop_realization, plot_chains, plot_ind, FixLS_da, rnormmix

Examples

```
fit <- est_mix_damcmc(spatstat::redwood, m = 3)
fit
plot(fit)
#We work with the California Earthquake data. We fit an IPPP with intensity surface modeled
#by a mixture with 5 normal components.
CAfit=est_mix_damcmc(CAQuakes2014.RichterOver3.0, m=5, L = 20000)
#Now retrieve the surface of Maximum a Posteriori (MAP) estimates of the mixture parameter.
#Note that the resulting surface is not affected by label switching.
MAPsurf=GetMAPEst(CAfit)
#Plot the states and the earthquake locations along with the fitted MAP IPPP intensity
#surface.
ret=PlotUSAStates(states=c('California','Nevada','Arizona'), showcentroids=FALSE,
 shownames=TRUE,main= "Earthquakes in CA, 2014",pp=CAQuakes2014.RichterOver3.0, surf=MAPsurf,
 boundarycolor="white",namescolor="white")
plot(CAfit)
#check labels
check_labels(CAfit)
# Fix label switching, start with approx=TRUE
post_fixed = FixLS_da(CAfit, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)
#this one works generally better but it is slow for large m
post_fixed = FixLS_da(CAfit,approx=FALSE, plot_result = TRUE)
```

```
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)


fitBD <- est_mix_bdmcmc(spatstat::redwood, m = 5)
fitBD
plotsBDredwood=plot(fitBD)
#Earthquakes example
CAfitBD=est_mix_bdmcmc(pp = CAQuakes2014.RichterOver3.0, m = 5)
BDtab=GetBDTable(CAfitBD)#retrieve frequency table and MAP estimate for number of components
BDtab
MAPm=BDtab$MAPcomp
plotsCAfitBD=plot(CAfitBD)
#get the surface of posterior means with MAP components and plot it
plotmix_2d(GetPMEst(CAfitBD,MAPm),CAQuakes2014.RichterOver3.0)
#retrieve all BDMCMC realizations corresponding to a mixture with MAP components
BDfitMAPcomp=GetBDCompfit(CAfitBD,MAPm)
BDfitMAPcomp
plot(BDfitMAPcomp$BDsurf,main=paste("Mixture intensity surface with",MAPm, "components"))
#Example of Dropping bad realizations and working with the MAP surface
open_new_plot=FALSE
truncate=FALSE
truemix4=rnormmix(m = 4, sig0 = .1, df = 5,xlim= c(-2,2), ylim = c(-2,2))
plot(truemix4,xlim= c(-2,2), ylim = c(-2,2),whichplots=0, open_new_window=
 open_new_plot)+add_title("True mixture of normals density")
trueintsurfmix4=to_int_surf(truemix4,lambda = 150,win =spatstat::owin( c(-2,2),c(-2,2)))
#not truncating so let us use a larger window
bigwin=spatstat::owin(c(-4,4),c(-4,4))
ppmix4 <- rsppmix(intsurf = trueintsurfmix4,truncate = truncate,win=bigwin)# draw points
print(plotmix_2d(trueintsurfmix4,ppmix4, open_new_window=open_new_plot,
 win=spatstat::owin(c(-4,4),c(-4,4)))+add_title(
 "True Poisson intensity surface along with the point pattern, W=[-4,4]x[-4,4]",
 lambda =trueintsurfmix4$lambda,m=trueintsurfmix4$m,n=ppmix4$n))
BDMCMCfit=est_mix_bdmcmc(pp = ppmix4, m = 5,L=30000,truncate = truncate)
#check the original distribution of the number of components
plot_CompDist(BDMCMCfit, open_new_window=open_new_plot)
#get the realizations corresponding to the MAP number of components
BDtab=GetBDTable(BDMCMCfit,FALSE)#retrieve frequency table and MAP estimate for the
#number of components
MAPm=BDtab$MAPcomp
BDMCMCfitMAPcomp=GetBDCompfit(BDMCMCfit,MAPm)
BDMCMCfitMAPcompgens=BDMCMCfitMAPcomp$BDgens
#look at the range of the means with the degenerate realizations included
print(range(BDMCMCfitMAPcompgens$genmus[,1,]))
print(range(BDMCMCfitMAPcompgens$genmus[,2,]))
#use the original output of the BDMCMC and apply 10% burnin (default)
BDMCMCfit=drop_realization(BDMCMCfit)
#now we drop the bad realizations
BDMCMCfit=drop_realization(BDMCMCfit, (BDMCMCfit$Badgen==1))
#we see how many realizations are left
plot_CompDist(BDMCMCfit, open_new_window=open_new_plot)
#get the realizations for the MAP only
BDMCMCfitMAPcomp=GetBDCompfit(BDMCMCfit,MAPm)
```

```
BDMCMCfitMAPcompgens=BDMCMCfitMAPcomp$BDgens
#check again the range of values for the x-y coords of the component means; they
#should be within the window
print(range(BDMCMCfitMAPcompgens$genmus[,1,]))
print(range(BDMCMCfitMAPcompgens$genmus[,2,]))
#check the MAP surface
plotmix_2d(BDMCMCfitMAPcomp$BDsurf,ppmix4, open_new_window=open_new_plot,
 win=bigwin) +add_title("MAP Poisson intensity surface along with the point pattern",
 lambda =BDMCMCfitMAPcomp$BDsurf$lambda, m=BDMCMCfitMAPcomp$BDsurf$m, n=ppmix4$n,
 L=BDMCMCfitMAPcomp$BDgens$L)
plot_chains(BDMCMCfitMAPcompgens, open_new_window=open_new_plot, separate = FALSE)
#burnin has been applied, set to zero and check for label switching
labelswitch=check_labels(BDMCMCfitMAPcompgens,burnin=0)
#use the identifiability constraint approach first
post_fixedBDMCMCfitIC = FixLS_da(BDMCMCfitMAPcompgens,burnin=0)
plot_chains(post_fixedBDMCMCfitIC, open_new_window=open_new_plot, separate = FALSE)
print(plot_ind(post_fixedBDMCMCfitIC, burnin=0, open_new_window=
 open_new_plot)+add_title("Posterior means of the membership indicators (IC permuted labels)",
 m = post_fixedBDMCMCfitIC$m, n = post_fixedBDMCMCfitIC$data$n))
permSurfaceofPostMeansIC=GetPMEst(post_fixedBDMCMCfitIC, burnin=0)
print(plotmix_2d(permSurfaceofPostMeansIC,ppmix4, open_new_window=open_new_plot,
 win=bigwin)+add_title("Poisson surface of posterior means (IC)",
 lambda=permSurfaceofPostMeansIC$lambda, m=permSurfaceofPostMeansIC$m, n=ppmix4$n,
 L=post_fixedBDMCMCfitIC$L))
#use the decision theoretic approach via SEL to find the best permutation; this one should
#work much better
post_fixedBDMCMCfitSEL = FixLS_da(BDMCMCfitMAPcompgens,approx=FALSE, burnin=0)
plot_chains(post_fixedBDMCMCfitSEL, open_new_window=open_new_plot, separate = FALSE)
print(plot_ind(post_fixedBDMCMCfitSEL, burnin=0, open_new_window=open_new_plot)+add_title(
 "Posterior means of the membership indicators (best permutation)",
 m=post_fixedBDMCMCfitSEL$m,n = post_fixedBDMCMCfitSEL$data$n))
permSurfaceofPostMeansSEL=GetPMEst(post_fixedBDMCMCfitSEL, burnin=0)
print(plotmix_2d(permSurfaceofPostMeansSEL,ppmix4, open_new_window=open_new_plot,
 win=bigwin)+ add_title("Poisson surface of posterior means (best permutation)",
 lambda=permSurfaceofPostMeansSEL$lambda, m=permSurfaceofPostMeansSEL$m,
 n=ppmix4$n,L=post_fixedBDMCMCfitSEL$L))
```

FixLS_da *Fix Label Switching*

Description

Permutes the posterior realizations in order to fix the labels by either applying an identifiability constraint or by minimizing the squared error loss to find the best permutation.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#FixLS_da

Usage

```
FixLS_da(fit, burnin = floor(fit$L/10), xlab = "x", ylab = "y",
  approx = TRUE, plot_result = FALSE, run_silent = FALSE)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
xlab	The label for the x-axis.
ylab	The label for the y-axis.
approx	Logical flag to request use of the identifiability constraint to permute all realizations. If FALSE, minimizing the loss function can be very slow for moderate to large number of components (m>10), since the algorithm goes through all m! permutations for each posterior realization.
plot_result	Logical flag for requesting plots of the point pattern and intensity surface based on the permuted realizations. The default is FALSE.
run_silent	Logical flag to hide progress messages. Default is FALSE.

Author(s)

Jiaxun Chen, Sakis Micheas

References

Jasra, A., Holmes, C.C. and Stephens, D. A. (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixtures. Statistical Science, 20, 50-67.

See Also

normmix, rsppmix, est_mix_damcmc, plot_chains, check_labels

Examples

```
# generate data
mix1 <- normmix(ps=c(.4, .2,.4), mus=list(c(0.3, 0.3), c(.5,.5),c(0.7, 0.7)),
 sigmas = list(.02*diag(2),.05*diag(2), .02*diag(2)),lambda = 100, win = spatstat::square(1))
#plot the true mixture
plot(mix1,main = "True Poisson intensity surface (mixture of normal components)")
pp1 <- rsppmix(mix1)
# Run Data augmentation MCMC and get posterior realizations
postfit = est_mix_damcmc(pp1, m=3, truncate=TRUE)
#plot the chains
plot_chains(postfit)
plot_chains(postfit,separate = FALSE)
# get the intensity of posterior means
post_mean = GetPMEst(postfit)
```

```
# plot the estimated intensity surface
plot(post_mean)
#check labels
check_labels(postfit)
# Fix label switching, start with approx=TRUE
post_fixed = FixLS_da(postfit, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed, separate = FALSE)
#this one works better in theory
post_fixed = FixLS_da(postfit, approx=FALSE, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed, separate = FALSE)
```

`GetBDCompfit` *Retrieve parts of a BDMCMC fit*

Description

The function can be used to obtain the realizations and the corresponding surface of posterior means, for a specific number of components. Use `GetPMEst` if you want just the surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetBDCompfit

Usage

```
GetBDCompfit(BDfit, num_comp, burnin = floor(BDfit$L/10))
```

Arguments

`BDfit`	Object of class `damcmc_res`.
`num_comp`	Number of components requested. Only the posterior realizations that have this many components will be returned. The function fails if the BDMCMC chain never visited this number of components.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.

Value

A list containing the following:

`BDgens`	Realizations corresponging to this many mixture components. This is a `damcmc_res` object (same as the result of a `est_mix_damcmc` call). All realizations for the requested number of components are returned, that is, burnin is not applied to this object.
`BDsurf`	For the requested `num_comp`, this is the Poisson intensity surface based on the corresponding posterior means (label switching might be present).
`BDnormmix`	For the requested `num_comp`, this is a `normmix` object containing the corresponding ps, mus and sigmas (label switching might be present).

Author(s)

Sakis Micheas

See Also

est_mix_bdmcmc, GetBDTable, plot.damcmc_res, plot.normmix

Examples

```
fit <- est_mix_bdmcmc(pp = spatstat::redwood, m = 7)
GetBDTable(fit)
#retrieve all BDMCMC realizations corresponding to a mixture with 5 components
BDfit5comp=GetBDCompfit(fit,5)
plot(BDfit5comp$BDsurf,main="Mixture intensity surface with 5 components")
#plot with the correct window
plot(BDfit5comp$BDnormmix,xlim =BDfit5comp$BDsurf$window$xrange,ylim =
 BDfit5comp$BDsurf$window$yrange )
plot(BDfit5comp$BDgens)
```

GetBDTable *Retrieve the MAP and distribution of the number of components*

Description

The function can be used to obtain the MAP estimate (mode of the posterior) along with the frequency table for the number of components, based on a BDMCMC fit from est_mix_bdmcmc.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetBDTable

Usage

```
GetBDTable(BDfit, showtable = TRUE)
```

Arguments

BDfit	A BDMCMC fit obtain from est_mix_bdmcmc.
showtable	Logical variable requesting to display the frequency table. Default is TRUE.

Value

A list containing the following:

MAPcomp	The MAP number of mixture components.
FreqTab	Frequency table for the number of components.
MeanComp	The posterior mean for the number of components.

Author(s)

Sakis Micheas

See Also

est_mix_bdmcmc

Examples

```
fit <- est_mix_bdmcmc(pp = spatstat::redwood, m = 7)
GetBDTable(fit)
```

GetBMA *Compute the Bayesian Model average*

Description

This function uses the posterior realizations from a est_mix_bdmcmc call, to compute the Bayesian Model Average across different number of components and returns the fitted Poisson point process with mixture of normals intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetBMA

Usage

```
GetBMA(fit, win = fit$data$window, burnin = fit$L/10, LL = 100,
  zlims = c(0, 0))
```

Arguments

fit	Object of class bdmcmc_res.
win	An object of class owin.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
LL	Length of the side of the square grid. The density or intensity is calculated on an L * L grid. The larger this value is, the slower the calculation, but the better the approximation as well as the smoother the resulting plots.
zlims	The limits of the z axis. Defaults to [0,max(z)].

Value

An image as an object of class im.object.

Author(s)

Sakis Micheas

See Also

est_mix_bdmcmc, plotmix_3d, plot_density

Examples

```
fit=est_mix_bdmcmc(pp = spatstat::redwood, m = 5)
BMA=GetBMA(fit)
burnin=.1*fit$L
title1 = paste("Bayesian model average of",fit$L-burnin,"posterior realizations")
plotmix_3d(BMA,title1=title1)
plot_density(as.data.frame(BMA))+ggplot2::ggtitle("Bayesian model average intensity surface")
plot_density(as.data.frame(BMA),TRUE)+ggplot2::ggtitle(
 "Contours of the Bayesian model average intensity surface")
```

GetDensityValues *Retrieve density values*

Description

This function operates on the point pattern and the realizations of a DAMCMC or BDMCMC fit (object damcmc_res or bdmcmc_res) and returns a plethora of information about the fit. When a bdmcmc_res is passed, only the realizations corresponding to the MAP number of components are used for calculations.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetDensityValues

Usage

```
GetDensityValues(fit)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.

Value

A list containing the following components:

Marginal	the value of the Marginal (approximately)
LogLikelihood	the value of the LogLikelihood

`CompDensityAtXi` the value of the component densities across all realizations and for each data point

`DensityAtXi` the value of the mixture density across all realizations and for each data point

`EntropyMAP` an approximation of the entropy of the distribution of the component indicators

`Density` the joint density at each posterior realization, i.e., for each iteration of `ps`, `mus` and `sigmas`.

Author(s)

Sakis Micheas

See Also

`normmix`, `est_mix_damcmc`, `est_mix_bdmcmc`, `rsppmix`

Examples

```
# create the true mixture intensity surface
truesurf =normmix(ps=c(.2, .6,.2), mus=list(c(0.3, 0.3), c(0.7, 0.7),
 c(0.5, 0.5)),sigmas=list(.01*diag(2), .01*diag(2), .01*diag(2)),
 lambda=100,win=spatstat::square(1))
plot(truesurf)
# generate the point pattern, truncate=TRUE by default
genPP=rsppmix(truesurf,truncate=FALSE)
fit=est_mix_damcmc(pp = genPP, m = 3)
allvals=GetDensityValues(fit)
MAPest=GetMAPEst(fit,vals=allvals)
plot(MAPest,main="IPPP intensity surface of MAP estimates")
```

GetIPPPLikValue *Retrieve the IPPP likelihood value*

Description

Given a point pattern this function calculates the IPPP likelihood value.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetIPPPLikValue

Usage

```
GetIPPPLikValue(pp, surf, truncate = FALSE)
```

Arguments

pp	Point pattern object of class ppp.
surf	IPPP intensity surface object of class intensity_surface.
truncate	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the point pattern pp.

Author(s)

Sakis Micheas

See Also

est_mix_damcmc, rmixsurf, rsppmix, GetPMEst

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100,xlim = c(-3,3),ylim = c(-3,3))
plot(truemix_surf,main="True IPPP intensity surface")
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
fit <- est_mix_damcmc(genPPP, m = 3)
MAPest=GetMAPEst(fit)
GetIPPPLikValue(genPPP,MAPest)
GetIPPPLikValue(genPPP,GetPMEst(fit))
```

GetKLEst *Retrieve the surface of Kullback-Leibler (KL) estimators*

Description

This function calculates the Kullback-Leibler estimators of the parameters of the components of the mixture intensity, based on a DAMCMC or BDMCMC fit. This is a decision theoretic estimator of the parameters, meaning that, we compute the Posterior Expected Loss (PEL) using the KL loss function and then find the parameter values that minimize the PEL.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetKLEst

Usage

```
GetKLEst(fit, burnin = floor(fit$L/10), fixLS = FALSE, approx = FALSE,
  segment = 50)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
fixLS	Logical requesting to check and fix label switching (if present). Default is FALSE.
approx	Logical flag to request use of the identifiability constraint to permute all realizations. Same parameter as in function FixLS_da.
segment	Number of segments to split the posterior realizations into. Each portion of posterior realizations is used to calculate a single Kullback-Leibler realization. The KL estimator is the average of all the KL realizations. Default is 50.

Value

An object of class intensity_surface.

Author(s)

Jiaxun Chen, Sakis Micheas

See Also

rmixsurf, rsppmix, est_mix_damcmc, GetPMEst, GetMAPEst, CompareSurfs

Examples

```
#generate a surface
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf,main="True IPPP intensity surface")
#generate a point pattern
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
#fit the IPPP model using DAMCMC
fit = est_mix_damcmc(genPPP, m = 3,L=20000)
#get the surfaces of posterior means, MAP and KL estimates
Meansest=GetPMEst(fit)
MAPest=GetMAPEst(fit)
KLest=GetKLEst(fit)
#plot all fitted surfaces
plot(Meansest,main="IPPP intensity surface of posterior means")
plot(MAPest,main="IPPP intensity surface of MAP estimates")
plot(KLest,main="IPPP intensity surface of KL estimates")
#fix labels (if label switching is detected)
KLestLSFixed=GetKLEst(fit,fixLS=TRUE,approx=FALSE)
plot(KLestLSFixed,main="IPPP intensity surface of KL estimates (LS fixed)")
#compare the four estimates against the truth
CompareSurfs(truemix_surf, Meansest, LL = 100, truncate = FALSE)
CompareSurfs(truemix_surf, MAPest, LL = 100, truncate = FALSE)
CompareSurfs(truemix_surf, KLest, LL = 100, truncate = FALSE)
CompareSurfs(truemix_surf, KLestLSFixed, LL = 100, truncate = FALSE)
```

GetMAPEst *Retrieve the surface of MAP estimators*

Description

The function calculates the Maximum A Posteriori (MAP) estimate of the IPPP mixture intensity surface parameters. Use function GetPMEst if you want the surface of posterior means.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetMAPEst

Usage

```
GetMAPEst(fit, burnin = floor(fit$L/10), vals, truncate = FALSE,
  priortype = 1, d, mu0, Sigma0, df0, sig0)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
vals	Contains the density values over the point pattern and realizations in the fit object. This can be obtained via a call to GetDensityValues. If this argument is missing then the density values are computed herein before computing the MAP estimates.
truncate	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the point pattern pp.
priortype	For different types of priors, ignored right now.
d, mu0, Sigma0, df0, sig0	Optional parameters for the prior distributions used: d are the weights of the Dirichlet prior on the component probabilities. mu0 and Sigma0 are the mean and covariance matrix of a bivariate normal that yields the component means. df0 and sig0 are the degrees of freedom and sig0^2*Identity the parameter matrix for the Inverse Wishart prior that yields the component matrices. If omitted they are set to the following values, which are the default values used in est_mix_damcmc: Sigma0=cov(cbind(ppx,ppy)), mu0=c(mean(pp$x),mean(pp$y)), sig0=1, df0=10, and d=rep(1,m).

Value

An object of type intensity_surface.

Author(s)

Sakis Micheas

See Also

est_mix_damcmc, rmixsurf, rsppmix, GetPMEst

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf,main="True IPPP intensity surface")
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
#the larger the number of realizations the better
fit <- est_mix_damcmc(genPPP, m = 3,L=100000)
MAPest=GetMAPEst(fit)
plot(GetPMEst(fit),main="IPPP intensity surface of posterior means")
plot(MAPest,main="IPPP intensity surface of MAP estimates")
fitBD <- est_mix_bdmcmc(pp = genPPP, m = 5)
MAPest=GetMAPEst(fitBD)
plot(MAPest,main="IPPP intensity surface of MAP estimates for MAP m")
```

GetMAPLabels *Retrieve the MAP estimates for the component labels*

Description

The function returns the Maximum A Posteriori (MAP) estimates of the component labels (membership indicator variables) based on a damcmc_res object (output from est_mix_damcmc) or a bdmcmc_res object (output from est_mix_bdmcmc) for the chain corresponding to MAP number of components.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetMAPLabels

Usage

```
GetMAPLabels(fit)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.

Value

A vector with size equal to the number of points, containing the MAP estimators of the component labels (or membership indicator variables). This the most likely component we would classify a point in.

Author(s)

Jiaxun Chen

See Also

normmix, to_int_surf, rsppmix, est_mix_damcmc

Examples

```
truemix <- normmix(ps=c(.4, .2,.4), mus=list(c(0.3, 0.3), c(.5,.5),c(0.7, 0.7)),
 sigmas = list(.02*diag(2), .05*diag(2), .01*diag(2)))
intsurf=to_int_surf(truemix,lambda = 100, win = spatstat::square(1))
pp1 <- rsppmix(intsurf)
plot(pp1)
plot(pp1, mus = intsurf$mus)#plot the mixture means as well
#plot the points with different colors depending on the true component label
plot(pp1, colors = TRUE)
#plot the points with different colors depending on the estimated component label
fit <- est_mix_damcmc(pp1, 3)
est_comp <- GetMAPLabels(fit)
plot(pp1, estcomp = est_comp, colors = TRUE)
fitBD <- est_mix_bdmcmc(pp1, 5)
est_compBD <- GetMAPLabels(fitBD)
plot(pp1, estcomp = est_compBD, colors = TRUE)
```

GetPMEst *Retrieve the Surface of Posterior Means*

Description

The function first calculates the posterior means of the parameters of the components of the mixture intensity, based on a DAMCMC or BDMCMC fit. Then the surface of posterior means is calculated using the posterior means of the parameters. For a BDMCMC fit, the number of components should be specified, and all realizations with that number of components are gathered to calculate the posterior intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetPMEst

Usage

```
GetPMEst(fit, num_comp = 1, burnin = floor(fit$L/10))
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
num_comp	Number of components requested. The posterior will be calculated only based on the posterior realizations that have this many mixture components. If missing the realizations corresponding to the MAP number of components are returned. This parameter is ignored if fit is of class damcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.

Value

An object of class intensity_surface.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

est_mix_damcmc,est_mix_bdmcmc

Examples

```
fit <- est_mix_damcmc(pp = spatstat::redwood, m = 3)
post_intsurf <- GetPMEst(fit, burnin = 1000)
plot(post_intsurf)
fit <- est_mix_bdmcmc(pp = spatstat::redwood, m = 5)
post_intsurf <- GetPMEst(fit, num_comp = 4, burnin = 1000)
plot(post_intsurf)
post_fixed = FixLS_da(fit,approx=FALSE, plot_result = TRUE)
plot(GetPMEst(post_fixed))
```

GetStats *Retrieves basic Bayesian estimates from a generated chain*

Description

The function returns the posterior mean and Credible Set for a parameter based on a chain of posterior realizations.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#GetStats

Usage

```
GetStats(chain, alpha = 0.05)
```

Arguments

chain	A Markov Chain (a vector) containing the posterior realizations of the parameter.
alpha	Level to use for the credible set.

Value

A list containing the min, max, mean, Credible Set and CredibleSetConfidence level.

Author(s)

Sakis Micheas

See Also

normmix, to_int_surf, rsppmix, est_mix_damcmc

Examples

```
truemix <- normmix(ps=c(.4, .2,.4), mus=list(c(0.3, 0.3), c(.5,.5),c(0.7, 0.7)),
 sigmas = list(.02*diag(2), .05*diag(2), .01*diag(2)))
intsurf=to_int_surf(truemix,lambda = 100, win = spatstat::square(1))
pp1 <- rsppmix(intsurf)
fit <- est_mix_damcmc(pp1, 3)
p1=GetStats(fit$genps[,1])
p1$Mean
p1$CredibleSet
p2=GetStats(fit$genps[,2])
p2$Mean
p2$CredibleSet
p3=GetStats(fit$genps[,3])
p3$Mean
p3$CredibleSet
```

Get_Rdiag *Checking convergence: diagnostics*

Description

This function reports the Gelman-Rubin convergence diagnostic R (also known as the potential scale reduction), by producing k DAMCMC fits and computing the within-chain and between-chain variances. Values approximately equal to 1 indicate convergence, otherwise we need to run the chain for a longer number of iterations to get convergence.

For DAMCMC examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Get_Rdiag

Usage

```
Get_Rdiag(pp, m, truncate = FALSE, L = 20000, numofchains = 2,
  permute = 0)
```

Arguments

pp	Point pattern object of class ppp.
m	The number of components to fit.
truncate	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the point pattern pp.
L	Number of iterations to use for each chain created; default is 20000. Note that half of them will be dropped so use a large number.
numofchains	Number of chains to create; default is 2.
permute	Request to generate chains that are unpermuted (permute=0), identifiability constraint (IC) permuted (permute=1), or minimum squared error loss (SEL) permuted (permute=2).

Author(s)

Sakis Micheas

See Also

est_mix_damcmc, rmixsurf, rsppmix

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf)
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
Get_Rdiag(pp = genPPP, m = 3)
```

---

kstest2d *Nonparametric Goodness-of-fit test between two point patterns*

Description

This function performs a two-dimensional Kolmogorov-Smirnov goodness-of-fit test on two point patterns.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#kstest2d

Usage

```
kstest2d(x1, x2, showinfo = TRUE)
```

Arguments

x1, x2	Objects of class ppp.
showinfo	Logical variable. Requests to display the test conclusion based on the value of the p-value. Default is TRUE.

Value

A list with class "htest" containing the following components:

statistic	Value of the KS statistic
p.value	The p-value of the test
alternative	A character string describing the alternative hypothesis

Author(s)

Jiaxun Chen, Sakis Micheas

References

Peacock, J.A. (1983). Two-dimensional goodness-of-fit testing in astronomy. Monthly Notices Royal Astronomy Society, 202, 615-627.

Adapted from Matlab code by Dylan Muir.

See Also

rnormmix,to_int_surf,owin

Examples

```
# generate two point patterns
mix1 <- rnormmix(3, sig0 = .01, df = 5, xlim=c(0, 5), ylim=c(0, 5))
intsurf1=to_int_surf(mix1,lambda = 40, win =spatstat::owin( c(0, 5),c(0, 5)))
mix2 <- rnormmix(8, sig0 = .01, df = 10, xlim=c(0, 5),ylim=c(0, 5))
intsurf2=to_int_surf(mix2,lambda = 50, win =spatstat::owin( c(0, 5),c(0, 5)))
#generate patterns from the two different models
pp1 <- rsppmix(intsurf1)
pp2 <- rsppmix(intsurf2)
pp3 <- rsppmix(intsurf2)#pp3 is from the same model as pp2
# Test for goodness of fit, p-value should be small
kstest2d(pp1, pp2)
# Test for goodness of fit, p-value should be large
kstest2d(pp2, pp3)
```

kstest2dsurf *Nonparametric Goodness-of-fit test for a point pattern against a surface*

Description

This function performs a two-dimensional Kolmogorov-Smirnov goodness-of-fit test for a point pattern against a given intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#kstest2dsurf

Usage

```
kstest2dsurf(pp, intsurf, truncate = FALSE, iters = 500)
```

Arguments

pp	Object of class ppp.
intsurf	Object of class intensity_surface.
truncate	Requests to truncate the generated point patterns to be within the window of the intensity object intsurf. Default is FALSE.
iters	Number of point patterns to generate and compare against pp. The larger this value is, the more test performed, and thus the more reliable the result.

Author(s)

Sakis Micheas

See Also

rmixsurf, kstest2d, rsppmix, plotmix_2d

Examples

```
# generate two intensity surfaces; assume the same window [-3,3]x[-3,3]
mixsurf1 <- rmixsurf(m = 3, lambda=100,xlim=c(-3,3),ylim=c(-3,3))
plot(mixsurf1)
mixsurf2 <- rmixsurf(m = 5, lambda=200,xlim=c(-3,3),ylim=c(-3,3))
plot(mixsurf2)
#generate point patterns from the two different models
pp1 <- rsppmix(mixsurf1, truncate=FALSE)
plotmix_2d(mixsurf1,pp1,colors=TRUE)
pp2 <- rsppmix(mixsurf2, truncate=FALSE)
plotmix_2d(mixsurf2,pp2,colors=TRUE)
# Test for goodness of fit, p-value should be small
kstest2d(pp1, pp2)
```

```
# Test each pattern for gof against both Poisson models
kstest2dsurf(pp1, mixsurf1)#correct model for pp1
kstest2dsurf(pp1, mixsurf2)#wrong model for pp1
kstest2dsurf(pp2, mixsurf2)#correct model for pp2
kstest2dsurf(pp2, mixsurf1)#wrong model for pp2
```

`MaternCov` *Matern covariance function*

Description

Computes the Matern covariance function. Used in the creation of stationary and isotropic Gaussian Random Fields (GRFs).

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#MaternCov

Usage

```
MaternCov(grid, nu = 0.5, theta = 1, sig = 1)
```

Arguments

`grid`	An `nx2` matrix of locations over which to compute the covariance matrix or an `nxn` matrix representing the distances of the `n` planar points.
`nu, theta, sig`	Matern model parameters. See details.

Details

The Matern covariance model for two points with Euclidean distance r, is given by

C(r) = sig^2 2^(1-nu) gamma(nu)^(-1) (sqrt(2nu) r/theta)^nu B_nu(sqrt(2nu) r/theta)

where sig, theta, nu>0, and B_nu is the modified Bessel function of second kind. Note that the Matern for nu=.5 reduces to the exponential.

Value

A matrix representing the covariance matrix for a random field (typically a GRF).

Author(s)

Sakis Micheas

See Also

`rGRF`

Examples

```
grid=cbind(seq(0,1,length=10), seq(0,1,length=10))
MaternCov(grid)
```

`mc_gof` *Monte Carlo goodness of fit test*

Description

Performs a Monte Carlo test of goodness-of-fit for a given point pattern. The entertained model is a Poisson with mixture of normals intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#mc_gof

Usage

```
mc_gof(pp, intsurf, alpha = 0.5, L = 20000, burnin = floor(0.1 * L),
  truncate = FALSE)
```

Arguments

`pp`	Point pattern object of class ppp.
`intsurf`	Object of class `intensity_surface`.
`alpha`	Significance level for the goodness-of-fit test.
`L`	Number of iterations requested; default is 20000.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
`truncate`	Requests to truncate the components of the mixture intensity to have all their mass within the window of the intensity object intsurf. Default is FALSE.

Details

The test statistic is the average of the average distances between the points assigned to the jth mixture component from the mean of the component. The Monte Carlo test utilizes realizations from the posterior predictive distribution to obtain the critical point, i.e., the ath percentile of the distribution of the test statistic. Make sure that L is large in order to get accurate results.

Author(s)

Jiaxun Chen, Sakis Micheas

See Also

`normmix`, `rsppmix`

Examples

```
# Create the intensity surface
intsurf1 <- normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2), c(.8, .8)), sigmas =
 list(.01*diag(2), .01*diag(2)), lambda = 100, win = spatstat::square(1))
# Generate a point pattern
pp1 <- rsppmix(intsurf1)
# Assess goodness-of-fit. Since this is the right model, we should get gof. Make
# sure L is large for more accurate results
mc_gof(pp1, intsurf1, 0.05)
# Create another intensity surface
intsurf2 <- normmix(ps = c(.5, .5), mus = list(c(0.2, 0.8), c(.8, .2)), sigmas =
 list(.01*diag(2), .01*diag(2)), lambda = 100, win = spatstat::square(1))
# Assess goodness-of-fit against this Poisson. Since this is the wrong model,
# we should NOT get gof
mc_gof(pp1, intsurf2, 0.05)
```

`normmix` *Create a 2d mixture with normal components*

Description

Constructor function for the `normmix` class. Creates a mixture in two dimensions with bivariate normal components. If the parameters lambda and window are set, this function will create an intensity surface object of class `intensity_surface`.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#normmix

The `print` function can be used on a `normmix` or `intensity_surface` object in order to display basic information.

The `summary` function can be used on a `normmix` or `intensity_surface` object in order to display additional information.

Usage

```
normmix(ps, mus, sigmas, lambda = NULL, win = NULL, estimated = FALSE)

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

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

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

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

Arguments

ps	Vector of component probabilities.
mus	A list where every element is a vector of length 2, defining the center of each component.
sigmas	A list where every element is a 2 by 2 covariance matrix, defining the covariance for each component.
lambda	Optional parameter denoting the average number of points over the window. If set along with the win parameter, the returned object will be an intensity surface.
win	Optional parameter for the window of observation, an object of type owin. Must be set together with lambda in order to create an intensity surface.
estimated	Logical variable to indicate that this is an estimated mixture not the true mixture surface. By default it is set to FALSE, but when using the function to define an mixture based on estimates of the parameters it should be set to TRUE.
x	An object of class normmix or intensity_surface.
...	Additional arguments for the S3 method.
object	An object of class normmix or intensity_surface.

Value

An object of class "normmix" containing the following components:

m	Number of components.
ps	Vector of component probabilities.
mus	List of mean vectors of the components.
sigmas	List of covariance matrices of the components.
lambda	Returned only if lambda is provided when calling normmix.
window	Returned only win is provided when calling normmix. This is an object of class owin.
estimated	Whether the normal mixture is estimated.

Author(s)

Yuchen Wang, Sakis Micheas

See Also

rnormmix for generating a mixture with random parameters.

Examples

```
mix1 <- normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2), c(.8, .8)),
 sigmas = list(.01*diag(2), .01*diag(2)))
mix1
summary(mix1)
```

```
intsurf1 <- normmix(ps = c(.3, .7), mus = list(c(0.2, 0.2), c(.8, .8)),
 sigmas = list(.01*diag(2), .01*diag(2)), lambda = 100, win = spatstat::square(1))
intsurf1
summary(intsurf1)
```

---

openwin_sppmix *Opens a new graphics window*

Description

This function is independent of the OS present, and is useful when working outside of RStudio. The latter GUI places all plots under the plots tab, but if working in the R GUI the plots will be overwritten if you don't open a new device.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#openwin_sppmix

Usage

```
openwin_sppmix(check2open = FALSE)
```

Arguments

check2open	Logical: TRUE to open a newplot, FALSE do not open.

Details

This function is used by almost all plotting functions of the sppmix package.

Author(s)

Sakis Micheas

Examples

```
openwin_sppmix(TRUE)
```

plot.bdmcmc_res *Plot results from a BDMCMC fit*

Description

This function uses the posterior realizations from a est_mix_bdmcmc call, to produce a plethora of plots about the fitted Poisson point process with mixture intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot.bdmcmc_res

Usage

```
## S3 method for class 'bdmcmc_res'
plot(x, win = fit$data$window,
  burnin = floor(fit$L/10), LL = 100, zlims = c(0, 0), ...)
```

Arguments

x	Object of class bdmcmc_res.
win	An object of class owin.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
LL	Length of the side of the square grid. The density or intensity is calculated on an L * L grid. The larger this value is, the slower the calculation, but the better the approximation as well as the smoother the resulting plots.
zlims	The limits of the z axis. Defaults to [0,max(z)].
...	Additional arguments for the S3 method.

Details

Unlike the corresponding output from DAMCMC (fixed number of components), the BDMCMC algorithm allows us to obtain a distribution for the number of components which can be thought of as a statistical inference procedure for model selection. In particular, the Bayesian model average of all the realizations is perhaps the best possible estimator of the Poisson intensity surface, however, it can be very slow to compute for moderate number of iterations and maximum number of components allowed.

Value

A list with the following objects

FreqTab	Frequency table for the number of components.
Mapsurf	The Maximum A Posteriori (MAP) Poisson intensity surface based on the corresponding posterior means (label switching might be present). The MAP is the mode of the distribution of the number of components.

BMA — The Bayesian Model Average is returned only if we answer "Y" to request it. Alternatively, use function GetBMA to compute the BMA. This is an image, i.e., an object of class im.object.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

est_mix_bdmcmc, PlotUSAStates, plotmix_3d, plot_density, check_labels, FixLS_da, plot_chains, GetBMA GetBDTable,

Examples

```
fit <- est_mix_bdmcmc(pp = spatstat::redwood, m = 10)
plot(fit)
#Tornadoes
ret=PlotUSAStates(states=c('Iowa','Arkansas', 'Missouri','Illinois','Indiana','Kentucky',
 'Tennessee', 'Kansas','Nebraska','Texas','Oklahoma','Mississippi', 'Alabama','Louisiana'),
 showcentroids=FALSE,shownames=TRUE, plotlevels = FALSE, main="Tornadoes about MO, 2011",
 pp=Tornadoes2011MO)
print(ret)
Tornfit=est_mix_bdmcmc(pp = Tornadoes2011MO, m = 7)
TornResults=plot(Tornfit)#if we plot the Bayesian model average return it
TornResults
if(!is.null(TornResults$BMA)){
  BMA_image=TornResults$BMA#must answer yes above or compute it using GetBMA
  burnin=.1*Tornfit$L
  title1 = paste("Bayesian model average of", Tornfit$L-burnin,"posterior realizations")
  plotmix_3d(BMA_image,title1=title1)
  plot_density(as.data.frame(BMA_image))+ggplot2::ggtitle(
   "Bayesian Model Average Intensity")
  plot_density(as.data.frame(BMA_image),TRUE)+ggplot2::ggtitle(
   "Contours of the Bayesian Model Average Intensity")}
# Work with the MAP intensity
Mapsurf=TornResults$Mapsurf
plot(Mapsurf)
#retrieve realizations for the MAP number of components only
tab=GetBDTable(Tornfit)
MAPm=tab$MAPcomp
BDfitMAPcomp=GetBDCompfit(Tornfit,MAPm)
summary(BDfitMAPcomp)
summary(BDfitMAPcomp$BDgens)
plot(BDfitMAPcomp$BDsurf,main=paste(
 "Poisson Mixture intensity surface, MAP # of components=",MAPm))
#check labels
check_labels(BDfitMAPcomp$BDgens)
# If present then fix label switching, start with approx=TRUE
post_fixed = FixLS_da(BDfitMAPcomp$BDgens, plot_result = TRUE)
plot_chains(post_fixed)
```

```
plot_chains(post_fixed,separate = FALSE)
#this one works better
post_fixed = FixLS_da(BDfitMAPcomp$BDgens,approx=FALSE, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)
```

`plot.damcmc_res` *Plot results from a DAMCMC fit*

Description

This function uses the posterior realizations from a `est_mix_damcmc` call, to produce a plethora of plots about the fitted Poisson point process with mixture intensity surface.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot.damcmc_res

Usage

```
## S3 method for class 'damcmc_res'
plot(x, burnin = floor(length(fit$allgens)/10), ...)
```

Arguments

`x`	Object of class `damcmc_res`.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
`...`	Additional arguments for the S3 method.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

`est_mix_damcmc`, `PlotUSAStates`, `GetPMEst`, `check_labels`, `FixLS_da`, `plot_chains`

Examples

```
fit <- est_mix_damcmc(pp = spatstat::redwood, m = 10)
plot(fit)
#Tornadoes
Tornfit=est_mix_damcmc(Tornadoes2011MO, m=5, L = 20000)
MAPsurf=GetMAPEst(Tornfit)
ret=PlotUSAStates(states=c('Iowa','Arkansas', 'Missouri','Illinois','Indiana',
 'Kentucky','Tennessee', 'Kansas','Nebraska','Texas','Oklahoma','Mississippi',
```

```
 'Alabama','Louisiana'),showcentroids=FALSE, shownames=TRUE, plotlevels = FALSE,
 main="Tornadoes about MO, 2011", pp=Tornadoes2011MO, surf=MAPsurf,
 boundarycolor="white", namescolor="white")
print(ret)
plot(Tornfit)
# get the intensity of posterior means
post_mean = GetPMEst(Tornfit)
# plot the estimated intensity surface
plot(post_mean)
#check labels
check_labels(Tornfit)
# If present then fix label switching, start with approx=TRUE
post_fixed = FixLS_da(Tornfit, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed,separate = FALSE)
```

`plot.intensity_surface` *Plots a normal mixture intensity in 3d*

Description

Plot the 3d intensity surface of a Poisson point process with mixture intensity of normal components.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot.intensity_surface

Usage

```
## S3 method for class 'intensity_surface'
plot(x, truncate = TRUE, L = 256, zlims = c(0,
  0), main = "Poisson intensity surface (mixture of normal components)",
  grayscale = FALSE, ...)
```

Arguments

`x`	Object of class `intensity_surface` or `normmix`.
`truncate`	Requests to truncate the components of the mixture intensity to have all their mass within the window of the intensity object intsurf. Default is TRUE.
`L`	Length of the side of the square grid. The intensity is calculated on an L * L grid. The larger this value is, the better the picture resolution.
`zlims`	The limits of the z axis. Defaults to [0,1.1*max(intensity)].
`main`	Title for the plot.
`grayscale`	Logical flag to request a gray scale plot.
`...`	Additional parameters passed to `to_int_surf()`.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, to_int_surf

Examples

```
truemix <- rnormmix(m = 5, sig0 = .1, df = 5, xlim= c(-1, 5), ylim =c(2, 5))
intsurf=to_int_surf(truemix, lambda = 200, win =spatstat::owin( c(-1, 5),c(2, 5)))
plot(intsurf,main = "True Poisson intensity surface (mixture of normal components)")
#use the demo intensity surface
demo_intsurf
summary(demo_intsurf)
#3d plot of the intensity surface
plot(demo_intsurf,main = "True Poisson intensity surface (mixture of normal components)")
```

---

plot.normmix *Plot a mixture of normal components*

Description

Create a 3d plot and 2d image or contour plots of the density of a mixture of normal components.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot.normmix

Usage

```
## S3 method for class 'normmix'
plot(x, xlim, ylim, contour = FALSE, truncate = FALSE,
  open_new_window = FALSE, grayscale = FALSE, L = 256,
  title1 = "Mixture with normal components", whichplots = 2, ...)
```

Arguments

x	Object of class normmix.
xlim, ylim	The observation window.
contour	Logical flag requesting the countour plot only.
truncate	Logical flag requesting that the components are truncated within the window.
open_new_window	Open a new window for the plot.
grayscale	Plot in gray scale. Default is FALSE (use colors).

L	Length of the side of the square grid. The intensity is calculated on an L * L grid. The larger this value is, the better the picture resolution.
title1	Optional title for the 3d plot.
whichplots	Requests plots of the normal mixture density (surface). To get only the 2d plot set whichplots=0, only the 3d plot set whichplots=1, or for both the 2d and 3d plots set whichplots=2. Default action is to produce both plots.
...	Additional arguments for the S3 method.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, to_int_surf, owin, rsppmix

Examples

```
# plot normmix density
truemix<- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(-1, 2), ylim = c(-1, 2))
summary(truemix)
#plot the normal mixture
plot(truemix, xlim= c(-1, 2), ylim = c(-1,2),
 title1="True mixture density in 3d")+add_title(
 "True mixture of normals density")
plot(truemix,xlim= c(-1, 2), ylim = c(-1, 2),contour = TRUE)+add_title(
 "Contour plot of the true mixture of normals density")
#build a mixture intensity surface for the Poisson point process
trueintsurf=to_int_surf(truemix, lambda = 100, win=
 spatstat::owin( c(-1, 2),c(-1, 2)))
plot(trueintsurf)#plot the surface, it is lambda*normmix
```

---

plot.sppmix *Plot a spatial point pattern*

Description

Plot a spatial point pattern generated from a Poisson with mixture intensity surface. Alternatively, the function can plot a spatstat ppp object.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot.sppmix

Usage

```
## S3 method for class 'sppmix'
plot(x, mus, estcomp, open_new_window = FALSE,
  colors = FALSE, showmarks = TRUE, whichmark = 1, ...)
```

Arguments

x	A point pattern of class sppmix or ppp.
mus	An optional list of the theoretical means of the mixture components.
estcomp	The estimated component label should be a vector whose length should be the same as number of points. If estcomp is not missing, the function will plot the points using different colors according to estcomp. See the example on how to calculate estcomp from a DAMCMC fit. If this variable is missing and we pass a point pattern generated using rsppmix, then the true component labels will be used, otherwise, the function will not plot the points with different colors to indicate the different components.
open_new_window	Open a new window for the plot.
colors	Logical flag requesting to use different colors for the points based on which component they belong to.
showmarks	Logical flag requesting to plot each point with a different circle size according to its mark value. If the mark is a data.frame object (multivariate marks), the first column (mark) is used as the the marks and displayed.
whichmark	If multivariate marks, choose to display this one.
...	Additional parameters to the add_title function. Valid choices are m, n and L. To add a different title than the default, use add_title after the plot call (see examples below).

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, to_int_surf, owin, rsppmix, est_mix_damcmc, GetMAPLabels, rMIPPP_cond_mark

Examples

```
mix1 <- rnormmix(5, sig0 = .01, df = 5, xlim=c(0, 5), ylim=c(0, 5))
intsurf1=to_int_surf(mix1, lambda = 40, win =spatstat::owin( c(0, 5),c(0, 5)))
pp1 <- rsppmix(intsurf1)
plot(pp1)
plot(pp1, mus=intsurf1$mus)
plot(pp1,mus=intsurf1$mus)+add_title(
 "Poisson point pattern along with the true component means", m=intsurf1$m,n=pp1$n)
plot(pp1, mus = intsurf1$mus, lambda = intsurf1$lambda)
plot(pp1, mus = intsurf1$mus)+ add_title(
 "Poisson point pattern along with the true component means", lambda = intsurf1$lambda,
 m=intsurf1$m,n=pp1$n)
#use the demo intensity surface
demo_intsurf
pp2 <- rsppmix(demo_intsurf,marks = 1:3)
plot(pp2)
```

```
plot(pp2, mus = demo_intsurf$mus)#plot the mixture means as well
#plot the points with different colors depending on the true component label
plot(pp2, colors = TRUE)
#plot the points with different colors depending on the estimated component label
fit <- est_mix_damcmc(pp2, 2)
est_comp <- GetMAPLabels(fit)
plot(pp2, estcomp = est_comp, colors = TRUE)
#generate and plot a marked point pattern
newMPP=rMIPPP_cond_mark()
plot(newMPP$genMPP, showmarks=TRUE)
```

---

plot2dPP *Plot a spatial point pattern*

Description

Standard 2d plot for a spatial point pattern.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot2dPP

Usage

```
plot2dPP(pp, mus, add2plot = FALSE, title1 = "Spatial point pattern",
  open_new_window = FALSE)
```

Arguments

pp	A point pattern of class sppmix or ppp.
mus	An optional list of the theoretical means of the mixture components.
add2plot	Logical variable to indicate if the function should add the points to an existing plot.
title1	Title for the plot.
open_new_window	Open a new window for the plot.

Author(s)

Sakis Micheas

See Also

normmix, to_int_surf, owin, rsppmix

Examples

```
mix1 <- rnormmix(5, sig0 = .01, df = 5, xlim=c(0, 5), ylim=c(0, 5))
intsurf1=to_int_surf(mix1, lambda = 40, win =spatstat::owin( c(0, 5),c(0, 5)))
pp1 <- rsppmix(intsurf1)
plot2dPP(pp1)
plot2dPP(pp1, mus = intsurf1$mus)
```

`plotmix_2d` *2d exploratory plots for mixture intensity surfaces*

Description

Create a 2d image or contour plot of the intensity surface, with the option to display a point pattern.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plotmix_2d

Usage

```
plotmix_2d(intsurf, pattern, estcomp, contour = FALSE, truncate = TRUE,
  L = 256, open_new_window = FALSE, grayscale = FALSE, colors = FALSE,
  ...)
```

Arguments

`intsurf`	Object of class `intensity_surface` or `normmix`.
`pattern`	Optional spatial point pattern to add to the plot. This is an object of class ppp.
`estcomp`	The estimated component label should be a vector whose length should be the same as number of points. If `estcomp` is not missing, the function will plot the points using different colors according to `estcomp`. See the example on how to calculate `estcomp` from a DAMCMC fit. If this variable is missing and we pass a point pattern generated using `rsppmix`, then the true component labels will be used, otherwise, the function will not plot the points with different colors to indicate the different components.
`contour`	Logical flag requesting the countour plot only.
`truncate`	Logical variable indicating that the points should be within the window of observation. Default is TRUE.
`L`	Length of the side of the square grid. The intensity is calculated on an L * L grid. The larger this value is, the better the picture resolution.
`open_new_window`	Open a new window for the plot.
`grayscale`	Plot in gray scale. Default is FALSE (use colors).
`colors`	Logical flag requesting to use different colors for the points based on which component they belong to.
`...`	Additional parameters passed to `to_int_surf()`.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, to_int_surf, owin, rsppmix, GetMAPLabels, est_mix_damcmc, PlotUSAStates

Examples

```
# plot normmix density
truemix<- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(0, 5), ylim = c(0, 5))
summary(truemix)
intsurf=to_int_surf(truemix, lambda = 100, win =spatstat::owin( c(0, 5),c(0, 5)))
#plot the intensity surface
plotmix_2d(intsurf)
plotmix_2d(intsurf,contour = TRUE)
pp1 <- rsppmix(intsurf = intsurf)# draw points
plotmix_2d(intsurf, pp1)
plotmix_2d(intsurf, pp1,contour = TRUE)
#fit a Poisson with mixture intensity surface
CAgens=est_mix_damcmc(pp = CAQuakes2014.RichterOver3.0, m = 5)
#retrieve the surface of posterior means
CAfit=GetPMEst(CAgens)
#plot the surface and the point pattern
plotmix_2d(CAfit,CAQuakes2014.RichterOver3.0)
#to include the state boundaries use function PlotUSAStates
ret=PlotUSAStates(states=c('California','Nevada','Arizona'), showcentroids=FALSE,
 shownames=TRUE, main="Earthquakes in CA, 2014", pp=CAQuakes2014.RichterOver3.0,
 surf=CAfit, boundarycolor="white", namescolor="white")
#plotting the points with different colors depending on the component they belong to
truemix <- rnormmix(m = 5, sig0 = .1, df = 5, xlim=c(-2,2), ylim=c(-2,2))
intsurf=to_int_surf(truemix, lambda = 100, win = spatstat::owin(c(-2,2),c(-2,2)))
pp1 <- rsppmix(intsurf)
#plot the points with different colors depending on the true component label
plotmix_2d(intsurf,pp1, colors = TRUE)
#plot the points with different colors depending on the estimated component label
fit <- est_mix_damcmc(pp1, 5)
est_comp <- GetMAPLabels(fit)
plotmix_2d(intsurf,pp1, estcomp = est_comp, colors = TRUE)
plotmix_2d(intsurf,pp1, estcomp = est_comp, contour = TRUE,colors = TRUE)
```

---

plotmix_3d *Plot the density or intensity of a normal mixture in 3d over a fine grid*

Description

When a normmix object is given, this function calculates the mixture density over a fine grid for the given window. When an intensity_surface object is given, the function multiplies the density with the surface lambda, and returns the Poisson mixture intensity function over the grid. Used for plotting.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plotmix_3d

Usage

```
plotmix_3d(dens_image, title1 = "3d Surface (Density or Intensity)",
  zlims = NULL, grayscale = FALSE)
```

Arguments

dens_image	An image as an object of class im.
title1	A title for the 3d plot.
zlims	The limits of the z axis. Defaults to [0,1.1*max(dens_image)].
grayscale	Plot in gray scale. Default is FALSE (use colors).

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

rnormmix, dnormmix

Examples

```
truemix <- rnormmix(m = 5, sig0 = .1, df = 5, xlim= c(0, 3), ylim = c(0, 3))
normdens=dnormmix(truemix, xlim= c(0, 3), ylim = c(0, 3))
plotmix_3d(normdens)
plotmix_3d(normdens, title1="Density of a normal mixture")
#use the demo_mix and demo_truemix3comp objects; the windows are found in the
#corresponding demo demo_intsurf and demo_intsurf3comp
demo_intsurf$window
normdens1=dnormmix(demo_mix, xlim= c(0, 1), ylim = c(0, 1))
plotmix_3d(normdens1, title1="Density of a normal mixture, 2 components")
#change the window
normdens1=dnormmix(demo_mix, xlim= c(-1, 1.5), ylim = c(-1, 1.5))
plotmix_3d(normdens1, title1="Density of a normal mixture, 2 components")
demo_intsurf3comp$window
normdens2=dnormmix(demo_truemix3comp, xlim= c(-1, 1), ylim = c(-2, 3))
plotmix_3d(normdens2, title1="Density of a normal mixture, 3 components")
```

plotstring *General Helper Functions*

Description

The plotstring function plots a string in a generic plot device.

Usage

```
plotstring(str = "Hello World")
```

Arguments

str	A string to display.

Examples

```
plotstring()
```

Plots_off *Closes all open plots*

Description

The function closes all Rgl plots, as well as, any graphics devices.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Plots_off

Usage

```
Plots_off()
```

Author(s)

Sakis Micheas

See Also

Save_AllOpenRglGraphs

Examples

```
Plots_off()
```

PlotUSAStates *Visualization of USA states and their counties*

Description

The function plots the requested USA state or county boundaries and additional information if requested or if certain parameters are supplied. We use this function for visualization of geostatistical data, in particular, (Marked) IPPPs.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#PlotUSAStates

Usage

```
PlotUSAStates(showcounties = FALSE, states = "Missouri",
  showcentroids = TRUE, typecentroid = 0, shownames = FALSE,
  showmarks = FALSE, grayscale = FALSE, open_new_window = FALSE,
  main = "States (true levels)", guidemain = "Level",
  discretelevels = TRUE, levels = 1:3, showplot = TRUE,
  plotlevels = TRUE, marks, pp, surf, boundarycolor = "black",
  namescolor = "black", ppsize = 1)
```

Arguments

showcounties	Logical to denote that we want a plot of counties. Default is FALSE. Setting this to TRUE will show all the counties for the states passed in the states parameter.
states	A vector of state names. Set to NULL to request all states or ContinentalUSA_state_names to show only the continental USA states.
showcentroids	Logical requesting to show centroids for each state or county. These centroids are returned in a ppp object. The centroid is chosen so that it is always within the state or county boundaries.
typecentroid	If showcentroids=TRUE we can display either the average of the boundary (typecentroid=0) or the "marker point" of the state or county (typecentroid=1). For convex states or counties, the latter point is the most south-western point of the state or county.
shownames	Logical to display the names of the states for showcounties=FALSE or counties for showcounties=TRUE.
showmarks	Logical to display the mark values given to each state for showcounties=FALSE or county for showcounties=TRUE.
grayscale	Logical to request plots in grayscale.
open_new_window	Logical to request plotting in a new graphics window.
main	A character string to serve as the main title for the plot.
guidemain	A character string to be used as the title for the guide used (legend or colorbar).

discretelevels	Logical indicating that the marks are discrete valued.
levels	When discretelevels=TRUE, the parameter levels contains all the possible discrete levels (marks). This is a vector of integers or strings. Default is 1:3.
showplot	Logical requesting to show the plot. Set to FALSE if you want to simply retrieve the centroids of the states or counties, in which case the plot will not be created.
plotlevels	Logical requesting that the levels (marks) of each state or county are displayed. If marks is not supplied, then for discretelevels=TRUE the mark of each state or county is uniformly generated over the values of levels, otherwise the marks are uniform in (0,1) (probabilities). If marks are given, then they are used to appropriately paint a state or county.
marks	A vector of length equal to the number of states or counties requested, containing the mark values for each state or county. A mark is an integer pointing to an element from the vector levels for discretelevels=TRUE, otherwise a real number.
pp	Optionally, a point pattern as an object of type ppp to be displayed over the created plot. The window of this point pattern will be used as the window of observation (overrides the window in the surf parameter).
surf	Optionally, an intensity surface as an object of type intensity_surface or an image object of class im to be plotted first and then the map will be displayed over this field. Supplying this parameter sets the flag plotlevels=FALSE automatically. The window of this intensity surface will be used as the window of observation.
boundarycolor	A specific color to use for drawing boundaries. Default is "black". Set to NULL if you do not want boundaries drawn.
namescolor	A specific color to use for drawing the state or county names when plotnames=TRUE. Default is "black".
ppsize	Size used in plotting the points. Default is 1.

Details

Note that we use the state and county longitude and latitude boundaries in the USAStatesCounties2016 object.

Value

A list containing the following components:

PPPcent	The centroids of the states or counties requested, returned as a marked point pattern.
PPPMarker	The marker points of the states or counties requested, returned as a marked point pattern.
itemnames	Vector of strings containing all items processed (i.e., either all state names or all county names).
p	The created plot, otherwise NULL.

Author(s)

Sakis Micheas and Jiaxun Chen

See Also

est_MIPPP_cond_loc, est_mix_damcmc, est_mix_bdmcmc, plot_CompDist, drop_realization, GetBDTable, GetBDCompfit, plotmix_2d, GetBMA, plot_MPP_probs, GetMAPEst

Examples

```
#plot the continental USA with uniformly sampled discrete marks from 10 different levels
ret=PlotUSAStates(states=ContinentalUSA_state_names, levels=1:10, grayscale = FALSE,
 shownames=TRUE, plotlevels =TRUE, discretelevels=TRUE, main="Continental USA (generated levels)")
#now use continuous marks
ret=PlotUSAStates(states=ContinentalUSA_state_names, shownames=FALSE, discretelevels=FALSE,
 main="Continental USA (generated probabilities)", guidemain="Probability", showcentroids = FALSE)
#Fit an IPPP to the California Earthquake data
fitDA=est_mix_damcmc(CAQuakes2014.RichterOver3.0, 8, L = 20000)
#get the surface of Maximum a Posteriori estimates
MAPsurf=GetMAPEst(fitDA)
#plot the states and the earthquake points along with the fitted MAP IPPP intensity surface
ret=PlotUSAStates(states=c('California','Nevada','Arizona'), showcentroids=FALSE,
 shownames=TRUE, main="Earthquakes in CA, 2014", pp=CAQuakes2014.RichterOver3.0, surf=MAPsurf,
 boundarycolor="white", namescolor="white")
#Visualize the Tornado data about MO
#plot the states and the tornado points
ret=PlotUSAStates(states=c('Iowa','Arkansas','Missouri','Illinois','Indiana','Kentucky',
 'Tennessee','Kansas','Nebraska','Texas','Oklahoma','Mississippi','Alabama','Louisiana'),
 showcentroids=FALSE, shownames=TRUE, plotlevels = FALSE, main="Tornadoes about MO, 2011",
 pp=Tornadoes2011MO)
#Visualize aggregate income levels in MO by county using data from the American Community
#Survey (ACS)
#plot in the original scale first; here we pass the marks vector which contains the aggregate
#income values of Missourian counties
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'), showcentroids=TRUE, typecentroid=1,
 discretelevels=FALSE, shownames=TRUE, plotlevels=TRUE, marks=MOAggIncomeLevelsPerCounty,
 main="Aggregate Income in MO, 2014", guidemain = "Income level", namescolor="gray",
 boundarycolor="gray")
#plot in the log scale
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'), showcentroids=TRUE, typecentroid=1,
 discretelevels=FALSE, shownames=TRUE, plotlevels=TRUE, marks=log(MOAggIncomeLevelsPerCounty),
 main="Aggregate Income in MO, 2014", guidemain = "Income level\n(log scale)", namescolor="gray",
 boundarycolor="gray")
#plot the marker points, county boundaries and names
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'), showcentroids=TRUE, typecentroid = 1,
 discretelevels=FALSE, shownames=TRUE, plotlevels=FALSE, marks=log(MOAggIncomeLevelsPerCounty),
 main="Marker points for Missouri counties")
#now plot only the marker points, we treat this as a marked IPPP
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'), showcentroids=TRUE, typecentroid = 1,
 discretelevels=FALSE, shownames=FALSE, plotlevels=FALSE, marks=log(MOAggIncomeLevelsPerCounty),
 main="Marker points for Missouri counties", boundarycolor = NULL)
```

```
#let us discretize log(income) to 3 levels; low if <=20, average if >20 and <=23, and high if >23
newmarks=rep(0,length(MOAggIncomeLevelsPerCounty))
newmarks[log(MOAggIncomeLevelsPerCounty)<=20]=1
newmarks[log(MOAggIncomeLevelsPerCounty)>20 & log(MOAggIncomeLevelsPerCounty)<=23]=2
newmarks[log(MOAggIncomeLevelsPerCounty)>23]=3
table(newmarks)
levels=c("low","average","high")
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'), showcentroids=TRUE, typecentroid=1,
 discretelevels=TRUE, shownames=TRUE, plotlevels=TRUE, main="Aggregate Income in MO, 2014",
 marks=newmarks, levels=levels, guidemain = "Income level", namescolor="gray",
 boundarycolor="gray")
#now fit a marked IPPP model, use the PP of marker points
MPP=ret$PPPMarker
mpp_est <- est_MIPPP_cond_loc(MPP,r=1, hyper=0.2)
plot_MPP_probs(mpp_est)
#now obtain a BDMCMC fit for the ground process this way we can cluster the data
BDMCMCfit <- est_mix_bdmcmc(MPP,m=10,L = 50000)
plot_CompDist(BDMCMCfit)
#use the original output of BDMCMC and apply 10% burnin (default)
BDMCMCfit=drop_realization(BDMCMCfit)
#get the realizations corresponding to the MAP number of components
BDtab=GetBDTable(BDMCMCfit,FALSE)#retrieve frequency table and MAP estimate for
#the number of components
MAPm=BDtab$MAPcomp
BDMCMCfitMAPcomp=GetBDCompfit(BDMCMCfit,MAPm)
BDMCMCfitMAPcompgens=BDMCMCfitMAPcomp$BDgens
MAPsurf=GetMAPEst(BDMCMCfitMAPcompgens)
plotmix_2d(MAPsurf,MPP)+add_title(
 "IPPP intensity surface of MAP estimates (MAP number of components)",
 lambda =MAPsurf$lambda, m=MAPsurf$m, n=MPP$n, L=MAPsurf$L)
plot_ind(BDMCMCfitMAPcompgens)
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'),
 showcentroids=TRUE, typecentroid=1, discretelevels=TRUE, shownames=TRUE,
 main="Ground surface of MAP estimates", marks=newmarks, levels=levels,
 guidemain = "Income level", namescolor="gray", boundarycolor="gray",
 pp=MPP, surf=MAPsurf)
#obtain and plot the Bayesian model average; first drop the bad realizations
BDMCMCfit=drop_realization(BDMCMCfit,(BDMCMCfit$Badgen==1))
BMAest=GetBMA(BDMCMCfit)
ret=PlotUSAStates(showcounties=TRUE, states=c('Missouri'),
 showcentroids=TRUE, typecentroid=1, discretelevels=TRUE, shownames=TRUE,
 main="Bayesian model average ground intensity surface", marks=newmarks,
 levels=levels, guidemain = "Income level", namescolor="gray",
 boundarycolor="gray", pp=MPP, surf=BMAest)
```

---

plot_autocorr *Checking convergence: autocorrelation plot*

Description

This function can be used to assess convergence by visualizing the autocorrelations between the draws of the Markov chain `chain`. The `lag k` autocorrelation `rho_k` is the correlation between every draw and its `k`th lag. We would expect the `k`th lag autocorrelation to be smaller as `k` increases (that is, the 100th and 1000th draws should be less correlated than the 100th and 105th draws). For higher values of k we anticipate small autocorrelation values, otherwise the chain is not `mixing` well (in other words we do not explore the parameter space adequately).

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_autocorr

Usage

```
plot_autocorr(chain, open_new_window = FALSE, maxlag = 100)
```

Arguments

`chain`	An `Lx1` vector containing the `L` posterior realizations.
`open_new_window`	Open a new window for the plot.
`maxlag`	The maximum lag value to consider. Default is 100.

Author(s)

Sakis Micheas

See Also

`est_mix_damcmc`, `rmixsurf`, `rsppmix`

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf)
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
fit <- est_mix_damcmc(pp = genPPP, m = 3)
plot_autocorr(fit$genps[,1])
plot_autocorr(fit$genps[,2])
plot_autocorr(fit$genps[,3])
```

plot_avgsurf *Plot the average intensity surface*

Description

This function calculates the intensity surface at each posterior realization and then computes the average for the intensity surface over a fine grid. The result is a much smoother posterior estimator of the intensity surface, which is not necessarily the same as the surface of posterior means, which is obtained by GetPMEst.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_avgsurf

Usage

```
plot_avgsurf(fit, win = fit$data$window, LL = 100,
  burnin = floor(fit$L/10), zlims = c(0, 0), grayscale = FALSE,
  showplot = TRUE)
```

Arguments

fit	An object the contains all posterior realizations, e.g., the return value from est_mix_damcmc or est_mix_bdmcmc.
win	An object of class owin.
LL	Length of the side of the square grid. The density or intensity is calculated on an L * L grid. The larger this value is, the slower the calculation, but the better the approximation.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
zlims	The limits of the z axis. Defaults to [0,1.1*max(intensity)].
grayscale	Logical flag to request a gray scale plot.
showplot	Logical flag to request that the plot will be shown. Set to FALSE if you want to return the im.object, but do not want to produce the 3d plot.

Value

An image as an object of class im.object.

Author(s)

Jiaxun Chen, Sakis Micheas

See Also

rnormmix, to_int_surf, rsppmix, est_mix_damcmc, plot_density, ggtitle, geom_point, plotmix_3d, GetPMEst

Examples

```
truemix <- rnormmix(m = 5, sig0 = .1, df = 5, xlim= c(-1, 1), ylim =c(0, 3))
trueintsurf=to_int_surf(truemix, lambda = 200, win =spatstat::owin( c(-1, 1),c(0, 3)))
plot(trueintsurf, main = "True Poisson intensity surface (mixture of normal components)")
pp1 <- rsppmix(trueintsurf)
# Run Data augmentation MCMC and get posterior realizations
postfit=est_mix_damcmc(pp1,m=5)
# Plot the average of the surfaces of the posterior realizations
avgsurf=plot_avgsurf(postfit, LL = 50)
p<-plot_density(as.data.frame(avgsurf))+ggplot2::ggtitle(
 "Average surface of the posterior realization surfaces\n x denotes a true component mean")
#show the point pattern points
pp_df <- data.frame(pp1$x,pp1$y)
names(pp_df) <- c("x", "y")
p<-p + ggplot2::geom_point(data = pp_df,size=0.8)
#show the true means
mean_df <- data.frame(do.call(rbind, trueintsurf$mus))
names(mean_df) <- c("x", "y")
p + ggplot2::geom_point(data = mean_df, color = "red", shape = "x", size = 5)
#repeat for the contour plot
p<-plot_density(as.data.frame(avgsurf),contour = TRUE)+ggplot2::ggtitle(
 "Average surface of the posterior realization surfaces\n x denotes a true component mean")
#show the point pattern points
pp_df <- data.frame(pp1$x,pp1$y)
names(pp_df) <- c("x", "y")
p<-p + ggplot2::geom_point(data = pp_df,size=0.8)
#show the true means
mean_df <- data.frame(do.call(rbind, trueintsurf$mus))
names(mean_df) <- c("x", "y")
p + ggplot2::geom_point(data = mean_df, color = "red", shape = "x", size = 5)
#plot the 3d surface again based on the returned object
plotmix_3d(avgsurf,title1 = paste("Average of", .9*postfit$L,
 "posterior realizations of the intensity surface"))
```

`plot_chains` *Plot MCMC chains*

Description

Plot the MCMC chains for all component means and probabilities, generated by `est_mix_damcmc` or `est_mix_bdmcmc`.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_chains

Usage

```
plot_chains(fit, burnin = floor(fit$L/10), chain = c("p", "x", "y"),
  ncol = fit$m%%3 + 1, separate = TRUE, open_new_window = FALSE)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
chain	Character vector choosing from c("p", "x", "y"). Multiple choices are supported. This will plot the MCMC chain for the requested variables.
ncol	Number of columns in each plot.
separate	Logical flag to request the the chains should be shown in separate plots or shown in one plot with different colors per component. The latter (separate=FALSE) is useful for spotting label switching visually.
open_new_window	Open a new window for the plot.

Author(s)

Jiaxun Chen, Sakis Micheas

See Also

PlotUSAStates, normmix, rsppmix, est_mix_damcmc, FixLS_da

Examples

```
fit <- est_mix_damcmc(pp = spatstat::redwood, m = 10)
plot(fit)
plot_chains(fit)
#plot the chains in the same plot with different colors
plot_chains(fit, separate = FALSE)
# Only plot the realizations for the component means
plot_chains(fit, chain = c("x", "y"))
#check labels
check_labels(fit)
#fix labels and plot the chains again
post_fixed = FixLS_da(fit, plot_result = TRUE)
plot_chains(post_fixed)
plot_chains(post_fixed, separate = FALSE)
#We work with the California Earthquake data. We fit an IPPP with intensity surface
#modeled by a mixture with 8 normal components.
CAfit=est_mix_damcmc(CAQuakes2014.RichterOver3.0, m=5, L = 20000)
#Now retrieve the surface of Maximum a Posteriori (MAP) estimates of the mixture parameter.
#Note that the resulting surface is not affected by label switching.
MAPsurf=GetMAPEst(CAfit)
#Plot the states and the earthquake locations along with the fitted MAP IPPP intensity surface
ret=PlotUSAStates(states=c('California','Nevada','Arizona'), showcentroids=FALSE,
 shownames=TRUE, main= "Earthquakes in CA, 2014", pp=CAQuakes2014.RichterOver3.0, surf=MAPsurf,
 boundarycolor="white", namescolor="white")
CAfit=est_mix_damcmc(pp = CAQuakes2014.RichterOver3.0, m = 5)
plot(CAfit)
```

```
check_labels(CAfit)
plot_chains(CAfit, separate = FALSE)
#fix labels and plot the chains again
post_fixedCA = FixLS_da(CAfit, plot_result = TRUE)
plot_chains(post_fixedCA, separate = FALSE)
```

`plot_CompDist` *Plots for the number of components*

Description

The function produces two plots: the trace plot for the number of components based on all realizations of a BDMCMC fit, and a barplot describing the distribution of the number of components.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_CompDist

Usage

```
plot_CompDist(fit, open_new_window = FALSE)
```

Arguments

`fit`	Object of class `bdmcmc_res`.
`open_new_window`	Open new windows for the two plots.

Author(s)

Sakis Micheas

See Also

`est_mix_bdmcmc`

Examples

```
fitBD <- est_mix_bdmcmc(spatstat::redwood, m = 10)
plot_CompDist(fitBD)
CAfitBD=est_mix_bdmcmc(pp = CAQuakes2014.RichterOver3.0, m = 10)
plot_CompDist(CAfitBD)
```

plot_convdiags *Checking convergence visually*

Description

Based on a 'damcmc_res' object, this function will produce many graphs to help assess convergence visually, including running mean plots and autocorrelation plots for all the parameters. This function calls plot_runmean and plot_autocorr for all parameters so we do not have to it individually.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_convdiags

Usage

```
plot_convdiags(fit, burnin = floor(fit$L/10), open_new_window = FALSE,
  maxlag = 100)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
open_new_window	Open a new window for the plot.
maxlag	The maximum lag value to consider. Default is 100.

Author(s)

Sakis Micheas

See Also

est_mix_damcmc, rmixsurf, plot_runmean, plot_autocorr, rsppmix

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf)
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
fit = est_mix_damcmc(pp = genPPP, m = 3)
plot_convdiags(fit)
```

plot_density *Plots a density or image*

Description

Create a 2d image or contour plot of the density, intensity or any image object.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_density

Usage

```
plot_density(density_df, contour = FALSE, grayscale = FALSE, pp = NULL,
  surf = NULL, ppsize = 1, main = "2d surface (density or intensity)")
```

Arguments

density_df	A data frame. Typically density_df=as.data.frame(imdens), where imdens an im object.
contour	Logical flag requesting the countour plot only.
grayscale	Plot in gray scale. Default is FALSE (use colors).
pp	Optional point pattern to display (a ppp or sppmix object).
surf	Optional intensity_surface object containing means to be displayed in the plot.
ppsize	Size of the points in the plot.
main	A title for the 2d plot.

Details

This function does not open a new window for the plot.

Author(s)

Sakis Micheas

See Also

dnormmix

Examples

```
# plot a mixture of normals density
truemix <- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(0, 5), ylim = c(0, 5))
summary(truemix)
normdens=dnormmix(truemix, xlim = c(0, 5), ylim = c(0, 5))
#2d plots
```

```
plot_density(normdens, main="2d mixture density plot\nWindow=[0,5]x[0,5]")
#Contour plot
plot_density(normdens, contour=TRUE, main="2d mixture contour plot\nWindow=[0,5]x[0,5]")
```

---

plot_ind *Plot membership indicators*

Description

The function plots the posterior means of the membership indicators (or allocation variables) of each point to one of the mixture components, based on a DAMCMC fit. These are the posterior probabilities of a point belonging to a component.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_ind

Usage

```
plot_ind(fit, burnin = floor(fit$L/10), open_new_window = FALSE)
```

Arguments

fit	Object of class damcmc_res or bdmcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
open_new_window	Open a new window for the plot.

Author(s)

Sakis Micheas, Yuchen Wang

See Also

est_mix_damcmc

Examples

```
fit <- est_mix_damcmc(pp = spatstat::redwood, m = 10)
plot_ind(fit)
```

plot_MPP_fields *Plot the mark probability fields*

Description

The function displays the mark probability fields for each location of a marked point pattern. These fields are simply the probabilities of observing the corresponding mark value at that location.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_MPP_fields

Usage

```
plot_MPP_fields(MPP, gammas, r, discrete_mark = TRUE, grayscale = FALSE,
  truncate = FALSE, open_new_window = FALSE, LL = 128)
```

Arguments

MPP	A marked point pattern as an object of class ppp.
gammas	For discrete marks (discrete_mark=TRUE), this is a vector of length equal to the number of marks. These parameters should typically be non-negative and they represent weights affecting the probability fields of each mark. For values close to 0, we get higher probabilities of observing this mark. Large positive values lead to small probabilities of observing the corresponding mark.
r	Radius used to define the neighborhood system. Any two locations within this distance are considered neighbors.
discrete_mark	Logical flag indicating whether the mark is discrete or not. Default is TRUE. For continuous marks set this to FALSE.
grayscale	Logical to request plots in grayscale.
truncate	Logical variable indicating to discard points if they are not within the window of observation. Default is FALSE.
open_new_window	Logical requesting a new window for the plot(s).
LL	Length of the side of the square grid. The larger this value is, the better the picture resolution.

Author(s)

Sakis Micheas

See Also

rMIPPP_cond_loc

Examples

```
newMPP=rMIPPP_cond_loc(gammas=c(.1,.2,.5), r=.5)
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
plot_MPP_fields(newMPP$genMPP,newMPP$gammas,newMPP$r)
plot_MPP_fields(newMPP$genMPP,newMPP$gammas,1)
plot_MPP_fields(newMPP$genMPP,newMPP$gammas,1.5)
plot_MPP_fields(newMPP$genMPP,newMPP$gammas,2)
```

---

plot_MPP_probs *Plot the mark probabilities of a marked point pattern*

Description

For discrete marks only, the function displays for each location of a marked point pattern, the probabilities of observing each of the discrete marks at that location.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_MPP_probs

Usage

```
plot_MPP_probs(MPPfit, truncate = FALSE, open_new_window = FALSE)
```

Arguments

MPPfit	Object of class MIPPP_fit.
truncate	Logical variable indicating to discard points if they are not within the window of observation. Default is FALSE.
open_new_window	Open a new window for the plot.

Author(s)

Sakis Micheas

See Also

rMIPPP_cond_loc, est_MIPPP_cond_loc

Examples

```
newMPP=rMIPPP_cond_loc(gammas=c(.1,.2,.5))
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
genMPP=newMPP$genMPP
newMPP$r
mpp_est <- est_MIPPP_cond_loc(genMPP,newMPP$r, hyper=0.2)
plot_MPP_probs(mpp_est)
```

`plot_runmean` *Checking convergence: running means plot*

Description

This function produces a running mean plot, that is, a plot of the iterations against the mean of the draws up to each iteration. If the plot is not a near constant line then convergence has not been achieved (e.g., label switching is present).

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_runmean

Usage

```
plot_runmean(chain, open_new_window = FALSE)
```

Arguments

`chain`	An `Lx1` vector containing the `L` posterior realizations.
`open_new_window`	Open a new window for the plot.

Author(s)

Sakis Micheas

See Also

`est_mix_damcmc`, `rmixsurf`, `rsppmix`

Examples

```
truemix_surf <- rmixsurf(m = 3, lambda=100, xlim = c(-3,3), ylim = c(-3,3))
plot(truemix_surf)
genPPP=rsppmix(intsurf = truemix_surf, truncate = FALSE)
fit <- est_mix_damcmc(pp = genPPP, m = 3)
plot_runmean(fit$genps[,1])
plot_runmean(fit$genps[,2])
```

```
plot_runmean(fit$genps[,3])
```

`plot_true_labels` *Plot the true membership indicators*

Description

The function plots the true membership indicators (or allocation variables) of each point to one of the mixture components, based on a generated `sppmix` object. These are the true probabilities of a point belonging to a component.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#plot_true_labels

Usage

```
plot_true_labels(pattern, open_new_window = FALSE)
```

Arguments

`pattern`	Object of class `sppmix`.
`open_new_window`	Open a new window for the plot.

Author(s)

Sakis Micheas

See Also

`rnormmix`, `to_int_surf`, `owin`, `rsppmix`

Examples

```
truemix <- rnormmix(m = 5, sig0 = .1, df = 5, xlim= c(-3, 3), ylim = c(-3, 3))
intsurf=to_int_surf(truemix, lambda = 100, win =spatstat::owin( c(-3, 3),c(-3, 3)))
pp1 <- rsppmix(intsurf,FALSE)
plot_true_labels(pp1)
```

rGRF *Generate a Gaussian Random Field*

Description

Generates Gaussian random fields (GRFs) and related fields via transformations. The spatial covariances are modeled using Matern's model.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rGRF

Usage

```
rGRF(mu = 0, gentype = 0, xlims = c(-5, 5), ylims = c(-5, 5),
  LL = 128, df = 10, nu = 0.5, theta = 1, sig = 1, pattern)
```

Arguments

mu	Mean of the stationary GRF.
gentype	Set to 0 for Gaussian, 1 for Chi-square. Default is gentype=0.
xlims, ylims	Vectors defining the grid limits of the x-y locations over which to compute the covariance matrix.
LL	Length of the side of the square grid.
df	Degrees of freedom (an integer) for the chi-square random field when gentype=1.
nu, theta, sig	Matern model parameters. See MaternCov for details.
pattern	Optionally, a point pattern as an object of type ppp containing locations within the window. The values of the generated GRF over these locations are returned as the marks of the point pattern pattern.

Details

The code of the rGRF function uses a modification of the functions sim.rf and matern.image.cov from the fields package, by Douglas Nychka, Reinhard Furrer, John Paige, and Stephan Sain.

Depending on the choice of the Matern model parameters we might end up having trouble with the FFT giving negative values. The code accounts for this event and adjusts the range of values via an increasing variable incr. If it still takes a long time to generate the fields try increasing the domain of observation using wider xlims and ylims.

Value

An image as an object of class im.object, containing the realization of the field over the grid. If argument pattern was supplied, the return value is now a list contaning the realization of the field as an image, augmented by the marked point pattern with locations in pattern and marks the field values over these locations. This capability is illustrated for realizations of marked point processes conditioning on continuous marks. See function rMIPPP_cond_loc for more details.

Author(s)

Sakis Micheas

See Also

MaternCov, plot_density, ggtitle, add_title

Examples

```
#Gaussian random field as an image
GRF1=rGRF()
p<-plot_density(as.data.frame(GRF1))
p_title<-expression( paste("GRF with Matern covariances, ", theta,"=1,",mu,"=0,",nu, "=.5,",
 sigma,"=1"))
p+ggplot2::ggtitle(p_title)
#or simply use the add_title function
p+add_title("GRF with Matern model covariances", mu=0,theta=1,nu=.5,sigma=1)
#Chi-Square random field as an image
ChiSqRF=rGRF(gentype=1,df=10)
p<-plot_density(as.data.frame(ChiSqRF))
p+add_title(paste(chi^{2}," random fields with Matern model covariances for the GRFs"),
mu=0,theta=1,nu=.5,sigma=1,df=10)
#Log-Gaussian random field as an image
GRF2=rGRF()
LogGRF=exp(rGRF())
p<-plot_density(as.data.frame(LogGRF))
p+add_title("Log-Gaussian random field with Matern model covariances", mu=0,theta=1,
 nu=.5,sigma=1)
```

---

rMIPPP_cond_loc *Generate a Marked Poisson point process (conditional on location)*

Description

This function generates realizations (point patterns) from a given Marked IPPP or a generated one. See details for the choice of models for the mark distribution. The location (ground) process is a standard IPPP (unmarked) with mixture intensity surface, and is responsible for the number of events in the point pattern.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rMIPPP_cond_loc

Usage

```
rMIPPP_cond_loc(surf, locPP, gammas, r, hyper = 0.01, truncate = FALSE,
  win = owin(c(-3, 3), c(-3, 3)), bigwin, discrete_mark = TRUE,
  open_new_window = FALSE, grayscale = FALSE, show_plots = TRUE,
  LL = 128, L = 50000, mark_distr_choice = 0, GRFmu = 0, df = 10,
  nu = 0.5, theta = 1, sig = 1)
```

Arguments

surf	An object of type intensity_surface representing the IPPP surface for the ground process. Omit this argument to create a surface randomly.
locPP	The ground IPPP (locations of the events). If missing then these are generated using a call to rsppmix. Note that if surf is not supplied, then it will be generated which may lead to completely inappropriate locations of the events, if the supplied locPP was created with a completely different surface. It is safer to supply both the surface and ground locations at the same time or none of the two, so that both will be generated.
gammas	For discrete marks (discrete_mark=TRUE), this is a vector of length equal to the number of marks. These parameters should typically be non-negative and they represent weights affecting the probability fields of each mark. For values close to 0, we get higher probabilities of observing this mark. Large positive values lead to small probabilities of observing the corresponding mark. Negative values are allowed, but they can lead to a mark not being present in the generated pattern. If the vector gammas is not supplied, then we randomly generate the number of marks from 1:10 and the values of the vector gammas from a gamma distribution.
r	Radius used to define the neighborhood system. Any two locations within this distance are considered neighbors. If missing, we randomly select the radius using the generated (ground) point pattern over the window parameter win.
hyper	Hyperparameter for the distribution of gamma.
truncate	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the window win. This affects the mixture model for the intensity surface of the ground process.
win	Object of type owin defining the window of observation.
bigwin	Object of type owin. If supplied, this will be the window of observation, even if the pattern is generated over win. Useful if we do not truncate (truncate=FALSE) and we want better presentation of the generated MIPPP.
discrete_mark	Logical flag indicating whether the mark is discrete or not. Default is TRUE. For continuous marks set this to FALSE.
open_new_window	Open a new window for a plot.
grayscale	Logical to request plots in grayscale.
show_plots	Logical variable requesting to produce exploratory plots of the Marked IPPP intensity surface and generated point pattern.

LL — Length of the side of the square grid. The larger this value is, the better the picture resolution.

L — Number of iterations. Required when sampling from the mark model conditional on locations.

mark_distr_choice — A number indicating which mark distribution to use. Currently we have only one choice in the discrete mark case, which is essentialy a Markov random field (MRF) over the window. See details for more on the mark model currently used. For continuous marks, we have two choices, Gaussian random field (GRF) for mark_distr_choice=0 or Chi-Square random field for mark_distr_choice=1.

GRFmu — This is the mean of the Gaussian random field. Only stationarity is currently supported (i.e., GRFmu does not depend on location). Used only if discrete_mark=FALSE.

df — Degrees of freedom (an integer) for the chi-square random field when mark_distr_choice=1. Default is df=10. Used only if discrete_mark=FALSE.

nu, theta, sig — Additional arguments passed to the MaternCov function in order to create the spatial covariance field. Default values are nu=.5, theta=1, and sig=1. See MaternCov for details. Used only if discrete_mark=FALSE.

Details

We assume that the joint distribution of a marked point pattern N=[s,m(s)] with n events is of the form:

p(N)=lambda^n*exp(-lambda)/(n!)*f(all s|theta1)*g(all m|theta2(s),all s)

where s denotes a location and m=m(s) a mark value at that location, lambda a parameter with the interpretation as the average number of points over the window of observation, and f, g are proper densities.

In order to simulate from this Marked IPPP we first simulate the number of events and their locations from an IPPP with mixture intensity surface lambda*f(s|theta1) (e.g., using rsppmix), and then generate the mark at that location s.

In the discrete mark case, the mark is modeled using a mixture distribution of Dirac measures on the marks with the probability q(m,s) of observing a specific mark value m depending on the current location s and the marks of its neighbors. Since we have a window of observation, any point in there can potentially be marked, which leads to q(m,s) being a field. In particular, the probability q(m,s) is analogous to

exp(-gammas_(j)*(sum over all neighbors of s of their marks minus m squared))

and when we fit the MIPPP model, our goal is to estimate the parameters gammas.

Note that if all gammas are zero then we fall back to a discrete uniform mark distribution.

The neighborhood system is controlled by r and is crucial in this case. Small values tend to produce probability fields with concentrated masses about observed events of the process, whereas, large neighborhoods allow us to borrow strength across locations and result in much smoother probability fields.

In the continuous case the mark is generated from a (typically stationary) Gaussian process or chi-squared random process, e.g., using function rGRF.

See Micheas (2014) for more details on Marked IPPP models via conditioning arguments.

Value

A list containing the following components:

surf	The generated or supplied intensity surface object surf for the ground process.
gammas	The generated or supplied parameters gammas. Returned only if discrete_mark=TRUE.
genMPP	The generated point pattern as an object of class ppp and sppmix. The member $marks contains the marks at each of the generated locations. If the ground PP locPP was supplied, this is also the ground process for the MIPPP and only the marks are generated (at those locations).
r	The generated or supplied parameter r. Returned only if discrete_mark=TRUE.
prob_fields	In the continuous mark case this is the realization of the random field (as an image im object). For discrete marks, this is a list of size equal to the number of marks containing the probability fields for each mark value.
prob_field_params	A list of the parameters used to create the continuous valued mark fields. Returned only if discrete_mark=FALSE.

Author(s)

Sakis Micheas

References

Hierarchical Bayesian Modeling of Marked Non-Homogeneous Poisson Processes with finite mixtures and inclusion of covariate information. Micheas, A.C. (2014). Journal of Applied Statistics, 41, 12, 2596-2615, DOI: 10.1080/02664763.2014.922167.

See Also

plot_MPP_fields

Examples

```
# Create a marked point pattern; use randomization and discrete marks (default values)
newMPP=rMIPPP_cond_loc()
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
newMPP$gammas
newMPP$genMPP
newMPP$r
print(table(newMPP$genMPP$marks))
#we can reproduce the random field plots anytime using the following call
plot_MPP_fields(newMPP$genMPP,newMPP$gammas,newMPP$r)
#Now generate continuous marks according to a Gaussian process
newMPP=rMIPPP_cond_loc(discrete_mark = FALSE)
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
#now the marks are taken from a chi-square field
newMPP=rMIPPP_cond_loc(mark_distr_choice=1, discrete_mark = FALSE)
plot(newMPP$surf,main="True IPPP intensity surface for the locations")
```

rMIPPP_cond_mark *Generate a Marked Poisson point process (conditional on mark)*

Description

This function generates realizations (point patterns) from a given Marked IPPP via conditioning of the joint intensity surface on its marked component. See details for the choice of models for the mark distribution. For each mark value we obtain a ground process. There processes are standard IPPP (unmarked) with mixture intensity surfaces. The mark distribution is responsible for the number of events in the point pattern.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rMIPPP_cond_mark

Usage

```
rMIPPP_cond_mark(lambda = 500, params = c(0.5, 0.5),
  mark_distr_choice = 0, truncate = FALSE, discrete_mark = TRUE,
  win = owin(c(-3, 3), c(-3, 3)), bigwin, open_new_window = FALSE,
  grayscale = FALSE, show_plots = TRUE)
```

Arguments

lambda	Average number of mark values observed over the window. This is the total number of points observed (on the average).
params	Parameters for the mark distribution. The value depends on the mark_distr_choice parameter, e.g., params is a vector of probabilities if the mark distribution is discrete (mark_distr_choice=0).
mark_distr_choice	A number indicating which mark distribution to use. In the discrete mark case, the mark distribution is discrete over the marks 1:length(params) with corresponding probabilities in params. The continuous mark case has not been implemented yet.
truncate	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the window win. This affects the mixture model for the intensity surface of the ground process.
discrete_mark	Logical flag indicating whether the mark is discrete or not. Default is TRUE. For continuous marks set this to FALSE.
win	Object of type owin defining the window of observation.
bigwin	Object of type owin. If supplied, this will be the window of observation, even if the pattern is generated over win. Useful if we do not truncate (truncate=FALSE) and we want better presentation of the generated MIPPP.
open_new_window	Open a new window for a plot.

grayscale	Logical to request plots in grayscale.
show_plots	Logical variable requesting to produce exploratory plots of the Marked IPPP intensity surface and generated point pattern for each mark.

Details

For discrete marks, we assume that the joint intensity function of a marked point pattern N=[s,m] with n events is of the form:

intensity(s,m)=lambda*M(m|theta1)*g(s(m)|theta2(m))

where m denotes a mark and s=s(m) a location with mark m, lambda a parameter with the interpretation as the average number of events over the window of observation, and M the mark distribution and g the ground intensity are proper densities.

In order to simulate from this Marked IPPP we first simulate the number of events and their marks from an IPPP with intensity lambda*M(m|theta1), and then generate the ground intensities for each mark. Marks are assumed to be independnet of each other and the mixture parameters describing each ground process are also assumed to be independent of each other.

The continuous mark case will be implemented in future releases.

See Micheas (2014) for more details on Marked IPPP models via conditioning arguments.

Value

A list containing the following components:

groundsurfs	A list of intensity_surface objects containing the surfaces of the ground processes (one for each discrete mark value).
groundPPs	A list of ppp objects containing the locations of the ground processes (one for each discrete mark value).
genMPP	The generated point pattern as an object of class ppp and sppmix. The member $marks contains the marks at each of the generated locations.
mark_distr_choice	The choice of mark distribution. Same as the supplied parameter.
params	The default or supplied parameter params.

Author(s)

Sakis Micheas

References

Hierarchical Bayesian Modeling of Marked Non-Homogeneous Poisson Processes with finite mixtures and inclusion of covariate information. Micheas, A.C. (2014). Journal of Applied Statistics, 41, 12, 2596-2615, DOI: 10.1080/02664763.2014.922167.

See Also

plotmix_2d

Examples

```
# Create a marked point pattern; use randomization and 2 discrete uniform
# marks (default values)
newMPP=rMIPPP_cond_mark(bigwin = spatstat::owin(c(-10,10),c(-10,10)))
newMPP$params
plot(newMPP$genMPP, showmarks=TRUE)+add_title("Marked Poisson point pattern",
 n=newMPP$genMPP$n, nmarks=2)
plotmix_2d(newMPP$groundsurfs[[1]], newMPP$groundPPs[[1]])+ add_title(
 "Poisson point pattern for mark 1", n=newMPP$genMPP$n, m=newMPP$groundsurfs[[1]]$m)
plotmix_2d(newMPP$groundsurfs[[2]], newMPP$groundPPs[[2]])+ add_title(
 "Poisson point pattern for mark 2", n=newMPP$genMPP$n, m=newMPP$groundsurfs[[2]]$m)
```

`rmixsurf` *Generate a Poisson process surface object*

Description

This function creates a Poisson point process intensity surface modeled as a mixture of normal components, on the given 2d window. The means, covariances and component probabilities are chosen randomly based on parameters passed to the function. The number of components can be either fixed or random.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rmixsurf

Usage

```
rmixsurf(m, lambda, sig0, df, rand_m = FALSE, xlim, ylim, dvec, mu0, Sigma0)
```

Arguments

`m`	Number of components of the mixture. If omitted, m is uniformly selected from 1 up to 10.
`lambda`	Average number of points over the window. If omitted lambda is generated from a Gamma with shape~Unif(1,10) and scale~Unif(50,100).
`sig0`	Tuning parameter for generating a random matrix from an Inverse Wishart distribution.
`df`	Degrees of freedom for generating a random matrix from an Inverse Wishart distribution.
`rand_m`	Request a random number of components. When `rand_m = TRUE`, the function will randomly choose a number of components from `1:m`.
`xlim, ylim`	Vectors defining the observation window. The component means are sampled uniformly over this window.

dvec	A vector of weights used in the Dirichlet distribution used to sample the mixture probabilities. If the dimension of dvec is not the same as the number of components, then dvec is either truncated to the same dimension or repeated to have dimension m. If missing, a vector of ones is used.
mu0, Sigma0	Mean and covariance matrix for a multivariate normal distribution, used to generate all component means. If mu0 is missing the center of the window is used. If Sigma0 is missing it is set to the identity matrix. If both mu0 and Sigma0 are missing, the component means are generated uniformly over the window of observation.

Value

Object of class intensity_surface.

Author(s)

Sakis Micheas

See Also

plotmix_2d, summary.intensity_surface, plot.intensity_surface

Examples

```
mixsurf1 <- rmixsurf(m = 3, lambda=100)
summary(mixsurf1)
plot(mixsurf1)
plotmix_2d(mixsurf1)
mixsurf2 <- rmixsurf(m = 5, lambda=200, rand_m = TRUE, ylim = c(-3, 3))
summary(mixsurf2)
plot(mixsurf2)
plotmix_2d(mixsurf2)
mixsurf3 <- rmixsurf(m = 5, lambda=200, rand_m = TRUE, Sigma0=.01*diag(2))
summary(mixsurf3)
plot(mixsurf3)
plotmix_2d(mixsurf3)
```

---

rnormmix *Generate a mixture with normal components*

Description

Generates a mixture on a 2d window where the means, covariances and component probabilities are chosen randomly. The number of components can be either fixed or random.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rnormmix

Usage

```
rnormmix(m, sig0, df = 10, rand_m = FALSE, xlim = c(0, 1), ylim = c(0,
  1), dvec, mu0, Sigma0)
```

Arguments

m	Number of components of the mixture.
sig0	Tuning parameter for generating a random matrix from an Inverse Wishart distribution. If this argument is missing it is set to .1 of the minimum width/height of the window.
df	Degrees of freedom for generating a random matrix from an Inverse Wishart distribution. Default is 10.
rand_m	Request a random number of components. When rand_m = TRUE, the function will randomly choose a number of components from 1:m.
xlim, ylim	Vectors defining the observation window. The component means are sampled uniformly over this window.
dvec	A vector of weights used in the Dirichlet distribution used to sample the mixture probabilities. If the dimension of dvec is not the same as the number of components, then dvec is either truncated to the same dimension or repeated to have dimension m. If missing, a vector of ones is used.
mu0, Sigma0	Mean and covariance matrix for a multivariate normal distribution, used to generate all component means. If mu0 is missing the center of the window is used. If Sigma0 is missing it is set to the identity matrix. If both mu0 and Sigma0 are missing, the component means are generated uniformly over the window of observation.

Value

Object of class normmix.

Author(s)

Sakis Micheas, Yuchen Wang

Examples

```
mix1 <- rnormmix(m = 3, sig0 = .1, df = 5)
summary(mix1)
mix2 <- rnormmix(m = 5, sig0 = .1, df = 5, rand_m = TRUE, ylim = c(0, 5))
summary(mix2)
```

rsppmix *Generate a point pattern from a Poisson process*

Description

This function generates a point pattern from a Poisson point process with intensity surface modeled by a mixture of normal components.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#rsppmix

Usage

```
rsppmix(intsurf, truncate = TRUE, marks = NULL, ...)

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

Arguments

intsurf	Object of class intensity_surface or normmix.
truncate	Logical variable indicating that the points should be within the window of observation. Default is TRUE.
marks	An optional vector defining the mark space. A mark value is randomly selected and attached to the generated locations. Default is NULL, so that we create an unmarked point pattern.
...	Further parameters passed to to_int_surf().
object	A point pattern object of class sppmix.

Details

If an intensity surface is passed to intsurf, the function generates a pattern from the Poisson directly. We can also pass a normal mixture of class normmix and specify the observation window win and the parameter lambda, which is interpreted as the average number of points over the window, as additional parameters.

Even if we pass an intensity surface to rsppmix(), we can still overwrite the lambda and win by passing them as additional parameters. See the examples for specific calls to the function.

For a given window W, the number of points N(W) in W follows a Poisson distribution, with intensity measure Lambda(W). The intensity surface of the Poisson process is the Radon-Nikodym derivative of the measure Lambda with respect to the Lebesgue measure.

The intensity surface is modeled using a multiple of a mixture of normal distributions, i.e., the intensity is given by

f(x|lambda,thetas)=lambda * sum(p_i*f_i(x|mu_i,sigma_i)),

where the parameters thetas consist of the mixture probabilities ps, normal component means mus, and covariances sigmas, with sum(p_i)=1.

When the masses of all the components f_i are within the window, then the mixture sum(p_i*f_i(x|mu_i,sigma_i)) integrates to 1 over W, so that the average number of points is given by E(N(W))=lambda.

If truncate = TRUE, we generate points from the unbounded Poisson with the given mixture intensity function until there are exactly n points in the window (rejection method). If truncate = FALSE, the function will not check if the points are inside the window.

Value

A point pattern of class c("sppmix","ppp"). The object has all the traits of the ppp object and in addition, a comp member indicating from which mixture component the event comes from. The object members include:

x : a vector of x coordinates of the events,

y : a vector of y coordinates of the events,

n : the number of events,

window : the window of observation (an object of class owin),

marks : optional vector of marks,

comp : vector of true allocation variables.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

normmix, rmixsurf, square, rsppmix, plot.sppmix, plotmix_2d, plot2dPP, plot.normmix, rnormmix, plotmix_3d

Examples

```
# create the true mixture
truemix_surf <- normmix(ps=c(.2, .6,.2), mus=list(c(0.3, 0.3), c(0.7, 0.7), c(0.5, 0.5)),
 sigmas = list(.01*diag(2), .03*diag(2), .02*diag(2)), lambda=100, win=spatstat::square(1))
plot(truemix_surf)
# generate the point pattern
genPPP1=rsppmix(truemix_surf)
summary(genPPP1)
plot2dPP(genPPP1)
plot2dPP(genPPP1,truemix_surf$mus)
plotmix_2d(truemix_surf,genPPP1)
# overwrite lambda or win
genPPP2=rsppmix(truemix_surf, lambda = 200)
plotmix_2d(truemix_surf,genPPP2)
genPPP3=rsppmix(truemix_surf, win = spatstat::square(2))
truemix_surf$window
plotmix_2d(truemix_surf,genPPP3)#will not see the points outside the surface window
plotmix_2d(truemix_surf,genPPP3, win = spatstat::square(2)) #have to pass the new window
#to see the points
#use normmix with additional parameters
```

```
truemix<- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(0, 3), ylim = c(0, 3))
plot(truemix)
normdens=dnormmix(truemix, xlim= c(0, 3), ylim = c(0, 3))
plotmix_3d(normdens)
genPPP4=rsppmix(truemix, lambda = 100, win = spatstat::square(3))
# turn off truncation
genPPP5=rsppmix(intsurf = truemix_surf, truncate = FALSE)
plotmix_2d(truemix_surf,genPPP5)
plotmix_2d(truemix_surf,genPPP5, win = spatstat::square(2))
plotmix_2d(truemix_surf,genPPP5,contour=TRUE)
intsurf6=rmixsurf(m=5,lambda=rgamma(1,shape=10,scale=5),
 df=5,sig0=1,rand_m=TRUE,mu0 = c(.5,.5),Sigma0 = 0.001*diag(2))
genPPP6=rsppmix(intsurf6,marks=1:3,truncate = FALSE)
plotmix_2d(intsurf6,genPPP6)
plot(genPPP6,showmarks=TRUE)
```

`Save_AllOpenRglGraphs` *Saves RGL plots*

Description

The function saves all open RGL plots (3d plots) to the specified directory and using a template name.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#Save_AllOpenRglGraphs

Usage

```
Save_AllOpenRglGraphs(dir1, filename1 = "RglGraph")
```

Arguments

`dir1`	Directory to save the plots.
`filename1`	Template filename. Each open plot is saved as filename1.png, filename2.png, and so forth.

Author(s)

Sakis Micheas

See Also

`Plots_off`

Examples

```
# use a temporary directory to save the plots
Save_AllOpenRglGraphs(dir1=tempdir())
```

`selectMix` *Mixture Model Selection*

Description

This function suggests the best number of components by computing model selection criteria, including AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), ICLC (Integrated Classification Likelihood Criterion).

Since the only parameter of interest is the number of components of the mixture, we consider several fixed numbers of components defined in the vector Ms, and we entertain mixture models with their other parameters approximated via the MAP estimators of DAMCMC runs.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#selectMix

Usage

```
selectMix(pp, Ms, L = 30000, burnin = 0.1 * L, truncate = FALSE,
  runallperms = 0)
```

Arguments

`pp`	Point pattern object of class `ppp`.
`Ms`	A vector of integers, representing different numbers of components to assess for the mixture model for the intensity function.
`L`	Number of iterations for the DAMCMC we run for each number of components in `Ms`; default is 30000.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
`truncate`	Logical variable indicating whether or not we normalize the densities of the mixture components to have all their mass within the window defined in the point pattern `pp`.
`runallperms`	Set to 0 to use an approximation to the Likelihood and Entropy within the MCMC (not affected by label switching). Set to 1 to use an identifiability constraint to permute the labels and use the posterior means of the parameters to compute the criteria. Set to 2 to use the decision theoretic approach (minimize Squared Error Loss) in order to permute the labels. The latter setting can take a long time to run for m>7.

Details

For each integer in the vector `Ms`, we fit a mixture with that many components using DAMCMC. Then the criteria are computed and presented at the end of the calculations.

Note that the AIC and BIC do not account for constraints in the parameter space of the mixture model parameters. The ICLC uses the estimated entropy of the distribution of the membership indicators and therefore should be trusted more in identifying the true number of components, instead of AIC and BIC.

In addition, we run Stephens' BDMCMC and present the posterior distribution for the number of components.

All these methods should serve us in making an informed choice about the true number of components and then proceed to fit the DAMCMC with the chosen number of components and take care of label switching (if present), in order to achieve mixture deconvolution. If we simply want the surface, then the Bayesian model average from the BDMCMC fit is the best solution.

Value

A list containing the following components:

`AIC`	the values of the AIC criterion
`BIC`	the values of the BIC criterion
`ICLC`	the values of the ICLC criterion
`Marginal`	the values of the marginal density
`LogLikelihood`	the values of the LogLikelihood

Author(s)

Jiaxun Chen, Sakis Micheas

References

Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components: an alternative to reversible jump methods. The Annals of Statistics, 28, 1, 40-74.

McLachlan, G., and Peel, D. (2000). Finite Mixture Models. Wiley-Interscience.

Jasra, A., Holmes, C.C. and Stephens, D. A. (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture. Statistical Science, 20, 50-67.

See Also

`normmix`, `square`, `est_mix_damcmc`, `est_mix_bdmcmc`, `GetBMA`, `FixLS_da`, `rsppmix`

Examples

```
# create the true mixture intensity surface
truesurf <- normmix(ps=c(.2, .6,.2), mus=list(c(0.3, 0.3), c(0.7, 0.7), c(0.5, 0.5)),
 sigmas = list(.01*diag(2), .01*diag(2), .01*diag(2)), lambda=100, win=spatstat::square(1))
plot(truesurf)
# generate the point pattern, truncate=TRUE by default
pp <- rsppmix(truesurf,truncate=FALSE)
plot(pp,mus=truesurf$mus)
# compute model selection criteria via an approximation that is not affected by label
```

```
# switching and will typically work well for large L
ModelSel=selectMix(pp,1:5,truncate=FALSE)
# show info
ModelSel
#generate the intensity surface randomly
truesurf <- rmixsurf(5,100,xlim = c(-3,3), ylim = c(-3,3), rand_m = TRUE)
truesurf
pp <- rsppmix(truesurf,truncate=FALSE)
ModelSel0=selectMix(pp,1:5,runallperms = 0, truncate=FALSE)
ModelSel1=selectMix(pp,1:5,runallperms = 1, truncate=FALSE)
ModelSel2=selectMix(pp,1:5,runallperms = 2, truncate=FALSE)
```

---

sppmix *sppmix: Poisson point process modeling using normal mixture models for the intensity surface*

Description

This help page contains a summary of the features of the R package sppmix.

The main page for the package is at

http://faculty.missouri.edu/~micheasa/sppmix/index.html

including the package vignettes and R code with examples for each function of the package.

Details

The sppmix package implements classes and methods for modeling spatial point process data using Poisson point processes, where the intensity surface is assumed to be a multiple of a finite additive mixture with normal components.

Comprehensive accounts on modeling spatial point processes can be found in the books by Illian et al. (2008), Gelfand et al. (2010), Diggle (2014), and Baddeley, Rubak and Turner (2015).

The idea of using mixtures of normals for the intensity function is not entirely new (e.g., Thomas processes or Poisson cluster processes). However, the Bayesian framework (DAMCMC) for general mixtures for a fixed number of components were recently presented by Chakraborty and Gelfand (2010), and for a fixed or random number of components in the context of marked point processes, by Micheas (2014) (both DAMCMC and BDMCMC). In Zhou et al. (2015), the authors entertained space-time models based on Poisson point processes with mixture intensity surfaces.

The addition of marks and the introduction of time to a point pattern, leads to general Marked Space-Time Poisson point processes, and such models will be investigated in future versions of the sppmix package.

Getting Started

We recommend that you run the basic demos in order to get a first taste of the capabilities of sppmix. Simply run

```
Demo_sppmix(#)
```

in order to view the available demos and tutorials (vignettes), by choosing an appropriate number #. A call `Demo_sppmix()` will bring up a browser with all the available vignetters of the package. The following demos are available:

1) Defining, generating, working with and basic plotting of sppmix objects.

2) Plotting in the sppmix package.

3-6) How to perform model fitting for a Poisson point process with intensity surface assumed to be a mixture of normals. There are 4 cases discussed: fixed or random number of components and edge effects are considered or not considered.

7) Model checking in the sppmix package.

8) Perform model fitting for a (discrete) marked Poisson point process with ground (locations process) intensity surface assumed to be a mixture of normals.

Bayesian computational methods

Data Augmentation MCMC (DAMCMC by Diebolt and Robert, 1994, function `est_mix_damcmc`) and Birth-Death MCMC (BDMCMC by Stephens, 2000, function `est_mix_bdmcmc`) are the two main MCMC methods we have implemented for estimating the parameters of the Poisson intensity surface, in a Bayesian framework. The intensity surface consists of a parameter `lambda`, interpreted as the average number of points over the window of observation and the mixture component parameters, including the component probabilities `ps`, the component means `mus`, and the component covariances `sigmas`. See the details section of the function `rsppmix`. The latter function is used for sampling a point pattern from the Poisson point process.

Objects in `sppmix`

We introduce an object of class `normmix` for handling 2d mixtures of bivariate normal components. This helps us build the Poisson point process intensity surface as an object of class `intensity_surface`.

The DAMCMC and BDMCMC functions, return objects `damcmc_res` and `bdmcmc_res`, respectively, representing the fitted model parameters, as well as, a plethora of information. The functions `plot` and `summary` can then be applied to these objects to obtain additional information. See `plot.damcmc_res`, `plot.bdmcmc_res`, `summary.damcmc_res`, and `summary.bdmcmc_res`, for more details.

External dependencies

We link and require several R packages, including `spatstat`, `rgl`, `ggplot2`, `Rcpp`, `RcppArmadillo`, `fields`, and `mvtnorm`.

In particular, we utilize the `ppp` and `owin` classes from the `spatstat` package in order to describe a point pattern.

The MCMC algorithms are implemented in C++ using `Rcpp` and `RcppArmadillo`, and were optimized after extensive testing, meaning that this approach is significantly faster than some other implementations of 2d mixture models.

Plotting is accomplished using the `rgl` package in order to create 3d plots of the intensity surfaces. In addition, the `fields` and `ggplot2` packages were used for 2d plots.

Use the help pages and please report errors!

There are many examples in the help pages of each and every function in the sppmix package. You will find all the answers you need there, in the demos and in the vignettes.

But just in case we missed something, or if you find a typo, or a mistake or have any other suggestions feel free to contact the package maintainer Sakis Micheas.

License Information

Furthermore, a lot of our examples utilize the PlotUSAStates function which requires the Cartographic Boundary Shapefiles (boundary data) provided by the USA Census Bureau at https://www.census.gov/geo/maps-data/data/tiger-cart-boundary.html. To our knowledge there is no special license required to use this data.

If you use the sppmix package in your research, we would appreciate a citation.

Citation

In order to cite the sppmix package in publications please use:

Micheas, A., and Chen, J. (2017): sppmix-Modeling spatial Poisson and related point processes using normal mixture models. R package. Package website: http://faculty.missouri.edu/~micheasa/sppmix/index.html.

Author(s)

Jiaxun Chen (Author and Creator) <chenjiaxun9@hotmail.com>.

Athanasios (Sakis) Christou Micheas (Author and Maintainer) <micheasa@missouri.edu>

Significant contributions on the package skeleton creation, plotting functions and other code by Yuchen Wang <ycwang0712@gmail.com>

References

Diebolt, J., and Robert, C. P. (1994). Estimation of Finite Mixture Distributions through Bayesian Sampling. Journal of the Royal Statistical Society B, 56, 2, 363-375.

Stephens, M. (2000). Bayesian analysis of mixture models with an unknown number of components-an alternative to reversible jump methods. The Annals of Statistics, 28, 1, 40-74.

McLachlan, G., and Peel, D. (2000). Finite Mixture Models. Wiley-Interscience.

Jasra, A., Holmes, C.C. and Stephens, D. A. (2005). Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixtures. Statistical Science, 20, 50-67.

Illian, J., Penttinen, A., Stoyan, H. and Stoyan, D. (2008) Statistical Analysis and Modelling of Spatial Point Patterns. Wiley.

Chakraborty, A., and Gelfand, A.E. (2010). Measurement error in spatial point patterns. Bayesian Analysis, 5, 97-122.

Gelfand, A.E., Diggle, P.J., Fuentes, M. and Guttorp, P., editors (2010) Handbook of Spatial Statistics. CRC Press.

Diggle, P.J. (2013) Statistical Analysis of Spatial and Spatio-Temporal Point Patterns, Third edition. Chapman and Hall/CRC.

Micheas, A. C. (2014). Hierarchical Bayesian Modeling of Marked Non-Homogeneous Poisson Processes with finite mixtures and inclusion of covariate information. Journal of Applied Statistics, 41, 12, 2596-2615.

Zhou, Z., Matteson, D.S., Woodard, D.B., Henderson, S.G., and Micheas, A.C. (2015). A Spatio-Temporal Point Process Model for Ambulance Demand. Journal of the American Statistical Association, 110, 509, 6-15.

Baddeley, A., Rubak, E. and Turner, R. (2015) Spatial Point Patterns: Methodology and Applications with R. Chapman and Hall/CRC Press.

`summary.bdmcmc_res` *Summarize BDMCMC results*

Description

Prints a brief summary of the results of a BDMCMC fit.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#summary.bdmcmc_res

Usage

```
## S3 method for class 'bdmcmc_res'
summary(object, num_comp, burnin = floor(object$L/10),
  alpha = 0.05, dgt = 4, ...)
```

Arguments

`object`	Object of class `bdmcmc_res`.
`num_comp`	Number of components requested. Only the posterior realizations that have this many components will be returned. The function fails if the BDMCMC chain never visited this number of components. We can also pass a vector of integer values and present the posterior means summary. If this argument is missing, the MAP estimator is chosen by default.
`burnin`	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
`alpha`	Level alpha for the credible sets. Default is 0.05, for 95 sets of the mixture parameters.
`dgt`	Number of digits to use (formatting the output).
`...`	Additional arguments for the S3 method.

Author(s)

Sakis Micheas

See Also

rnormmix, to_int_surf, rsppmix, est_mix_bdmcmc, owin

Examples

```
# generate data
truemix<- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(0, 5), ylim = c(0, 5))
summary(truemix)
intsurf=to_int_surf(truemix, lambda = 100, win =spatstat::owin( c(0, 5),c(0, 5)))
pp1 <- rsppmix(intsurf = intsurf)# draw points
#Run BDMCMC and get posterior realizations
postfit=est_mix_bdmcmc(pp1,m=5)
#summary of the posterior results
summary(postfit)
summary(postfit, num_comp=2)
summary(postfit, num_comp=c(2,4))
```

---

summary.damcmc_res *Summarize DAMCMC results*

Description

Prints a brief summary of the results of a DAMCMC fit.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#summary.damcmc_res

Usage

```
## S3 method for class 'damcmc_res'
summary(object, burnin = object$L/10, alpha = 0.05,
  dgt = 4, ...)
```

Arguments

object	Object of class damcmc_res.
burnin	Number of initial realizations to discard. By default, it is 1/10 of the total number of iterations.
alpha	Level alpha for the credible sets. Default is 0.05, for 95 sets of the mixture parameters.
dgt	Number of digits to use (formatting the output).
...	Additional arguments for the S3 method.

Author(s)

Jiaxun Chen, Sakis Micheas, Yuchen Wang

See Also

rnormmix, to_int_surf, rsppmix, est_mix_damcmc, owin

Examples

```
# generate data
truemix<- rnormmix(m = 3, sig0 = .1, df = 5, xlim= c(0, 5), ylim = c(0, 5))
summary(truemix)
intsurf=to_int_surf(truemix, lambda = 100, win =spatstat::owin( c(0, 5),c(0, 5)))
pp1 = rsppmix(intsurf = intsurf)# draw points
#Run DAMCMC and get posterior realizations
postfit=est_mix_damcmc(pp1,m=3)
#summary of the posterior results
summary(postfit)
```

---

to_int_surf *Convert a normal mixture to an intensity surface*

Description

This function converts a normmix object into an intensity_surface object. It can also be used to change the parameters lambda (average number of points over the window) or win (window of observation) of an intensity_surface object.

For examples see

http://faculty.missouri.edu/~micheasa/sppmix/sppmix_all_examples.html#to_int_surf

If the class of mix is normmix, lambda and win are used to convert mix into an intensity surface class. If the class of mix is intensity_surface already, lambda and win are used to change the original settings for these parameters.

Usage

```
to_int_surf(mix, lambda = NULL, win = NULL, return_normmix = FALSE)
```

Arguments

mix	Object of class normmix or intensity_surface.
lambda	Optional parameter treated as the average number of points over the window.
win	Optional parameter of class owin, defining the window of observation.
return_normmix	Logical variable requesting to return a normal mixture (discard lambda and win).

Value

Object of class `intensity_surface`.

Author(s)

Yuchen Wang

See Also

`normmix`, `rnormmix`, `square`

Examples

```
truemix <- normmix(ps=c(.4, .2,.4), mus=list(c(0.3, 0.3), c(.5,.5),c(0.7, 0.7)),
 sigmas = list(.02*diag(2), .05*diag(2),.01*diag(2)))
intsurf=to_int_surf(truemix, lambda = 100, win = spatstat::square(1))
#plot the true mixture
plot(intsurf,main = "True Poisson intensity surface (mixture of normal components)")
# using the demo_mix normmix object
summary(demo_mix)
demo_surf1=to_int_surf(demo_mix, lambda = 100, win = spatstat::square(1))
plot(demo_surf1)
# using an intensity_surface object
summary(demo_intsurf)
demo_surf2=to_int_surf(demo_intsurf, win = spatstat::square(2))
summary(demo_surf2)
plot(demo_surf2)
demo_surf3=to_int_surf(demo_intsurf, lambda = 50)
plot(demo_surf3)
```

Index

∗ **datasets**
Datasets, 10
demo_mix, 12

add_title, 3, *79*
approx_normmix, 4

CAQuakes2014 *(*Datasets*)*, 10
check_labels, 5, *26*, *29*, *51, 52*
ChicagoArea *(*Datasets*)*, 10
ChicagoCrime2015 *(*Datasets*)*, 10
CompareSurfs, 7, *36*
ContinentalUSA_state_names *(*Datasets*)*, 10
Count_pts, 8

Datasets, 10
demo_genPPP *(*demo_mix*)*, 12
demo_intsurf *(*demo_mix*)*, 12
demo_intsurf3comp *(*demo_mix*)*, 12
demo_mix, 12
Demo_sppmix, 13
demo_truemix3 *(*demo_mix*)*, 12
demo_truemix3comp *(*demo_mix*)*, 12
demo_truesurf3 *(*demo_mix*)*, 12
dnormmix, *4*, 14, *60*, *72*
drop_realization, *8*, 15, *26*, *64*

est_intensity_np, 16
est_MIPPP_cond_loc, 17, *18*, *64*, *75*
est_MIPPP_cond_mark, 21
est_mix_bdmcmc, *8*, *16*, *19*, *31–34*, *38*, *40*, *50, 51*, *64*, *67, 68*, *70*, *92*, *94*, *97*
est_mix_bdmcmc *(*est_mix_damcmc*)*, 24
est_mix_damcmc, *6*, *19*, *22*, 24, *29, 30*, *34–36*, *38–42*, *52*, *56*, *59*, *64*, *66–69*, *71*, *73*, *76*, *92*, *94*, *98*

fields, *78*, *94*
FixLS_da, *6*, *26*, 28, *36*, *51, 52*, *69*, *92*

geom_point, *67*
Get_Rdiag, 41
GetBDCompfit, *6*, *26*, 30, *64*
GetBDTable, *26*, *31*, 31, *51*, *64*
GetBMA, *6*, *8*, 32, *51*, *64*, *92*
GetDensityValues, 33, *37*
GetIPPPLikValue, 34
GetKLEst, 35
GetMAPEst, *36*, 37, *64*
GetMAPLabels, 38, *56*, *59*
GetPMEst, *5*, *26*, *30*, *35–38*, 39, *52*, *67*
GetStats, *20*, *23*, 40
ggtitle, *67*, *79*

im, *7*, *14*, *17*, *23*, *60*, *63*, *72*, *82*
im.object, *32*, *51*, *67*, *78*

kstest2d, 42, *44*
kstest2dsurf, 44

matern.image.cov, *78*
MaternCov, *4*, 45, *78, 79*, *81*
mc_gof, 46
MOAggIncomeLevelsPerCounty *(*Datasets*)*, 10

normmix, *4*, *6*, *14*, *29, 30*, *34*, *39*, *41*, *46*, 47, *54–57*, *59*, *69*, *89*, *92*, *94*, *99*

openwin_sppmix, 49
owin, *7–9*, *16*, *26*, *32*, *43*, *48*, *50*, *55–57*, *59*, *67*, *77*, *80*, *83*, *89*, *94*, *97, 98*

plot.bdmcmc_res, *26*, 50, *94*
plot.damcmc_res, *5*, *31*, 52, *94*
plot.intensity_surface, *26*, 53, *86*
plot.MIPPP_fit *(*est_MIPPP_cond_loc*)*, 17
plot.normmix, *31*, 54, *89*
plot.sppmix, 55, *89*
plot2dPP, 57, *89*
plot_autocorr, 65, *71*

plot_avgsurf, *5*, *8*, 67
plot_chains, *6*, *26*, *29*, *51, 52*, 68
plot_CompDist, *26*, *64*, 70
plot_convdiags, 71
plot_density, *14*, *33*, *51*, *67*, 72, *79*
plot_ind, *26*, 73
plot_MPP_fields, 74, *82*
plot_MPP_probs, *64*, 75
plot_runmean, *71*, 76
plot_true_labels, 77
plotmix_2d, *9*, *14*, *26*, *44*, 58, *64*, *84*, *86*, *89*
plotmix_3d, *8*, *17*, *33*, *51*, 59, *67*, *89*
Plots_off, 61, *90*
plotstring, 61
PlotUSAStates, *26*, *51, 52*, *59*, 62, *69*, *95*
ppp, *9, 10*, *12*, *16*, *22*, *24*, *42–44*, *55–58*, *62, 63*, *72*, *74*, *78*, *82*, *84*, *89*, *94*
print.intensity_surface *(*normmix*)*, 47
print.normmix *(*normmix*)*, 47

Rcpp, *94*
RcppArmadillo, *94*
rgl, *94*
rGRF, *45*, 78, *81*
rMIPPP_cond_loc, *20*, *74, 75*, 79
rMIPPP_cond_mark, *23*, *56*, 83
rmixsurf, *8, 9*, *35, 36*, *38*, *42*, *44*, *66*, *71*, *76*, 85, *89*
rnormmix, *4*, *14*, *17*, *26*, *43*, *48*, *60*, *67*, *77*, 86, *89*, *97–99*
rsppmix, *6*, *8, 9*, *17*, *29*, *34–36*, *38, 39*, *41, 42*, *44*, *46*, *55–57*, *59*, *66, 67*, *69*, *71*, *76, 77*, *80, 81*, 88, *89*, *92*, *94*, *97, 98*

Save_AllOpenRglGraphs, *61*, 90
selectMix, 91
sim.rf, *78*
spatstat, *94*
sppmix, *12*, *19*, *49*, *56*, 93
sppmix-package *(*sppmix*)*, 93
square, *9*, *89*, *92*, *99*
summary.bdmcmc_res, *94*, 96
summary.damcmc_res, *94*, 97
summary.intensity_surface, *86*
summary.intensity_surface *(*normmix*)*, 47
summary.MIPPP_fit *(*est_MIPPP_cond_loc*)*, 17
summary.normmix *(*normmix*)*, 47
summary.sppmix *(*rsppmix*)*, 88

to_int_surf, *14*, *17*, *26*, *39*, *41*, *43*, *54–57*, *59*, *67*, *77*, *97, 98*, 98
Tornadoes2011MO *(*Datasets*)*, 10
TornadoesAll *(*Datasets*)*, 10

USAStatesCounties2016, *63*
USAStatesCounties2016 *(*Datasets*)*, 10