Package 'BayesBD'

August 21, 2017

Type Package

Title Bayesian Inference for Image Boundaries

Version 1.2

Date 2017-08-18

Author Nicholas Syring, Meng Li

Maintainer Nicholas Syring <nasyrin@gmail.com>

Description

Provides tools for carrying out a Bayesian analysis of image boundaries. Functions are provided for both binary (Bernoulli) and continuous (Gaussian) images. Examples, along with an interactive shiny function
illustrate how to perform simulations, analyze custom data, and plot estimates and credible intervals.

License GPL (>= 3)

Imports Rcpp (>= 0.12.5), shiny, mritc, png, jpeg

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-08-21 17:32:06 UTC

R topics documented:

`BayesBD-package` *Bayesian Inference for Image Boundaries*

Description

This package provides tools for carrying out a Bayesian analysis of image boundaries. Functions are provided for both binary (Bernoulli) and continuous (Gaussian) images. Examples, along with an interactive shiny function illustrate how to perform simulations, analyze custom data, and plot estimates and credible intervals.

Details

Package: BayesBD License: GPL (>= 3) Imports: Rcpp (>= 0.12.5), RcppArmadillo, shiny, png, jpeg, mritc LinkingTo: Rcpp, RcppArmadillo

Author(s)

Nicholas Syring, Meng Li

Maintainer: Nicholas Syring <nasyrin@gmail.com>

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

`BayesBDbinary` *Bayesian boundary estimation for binary images*

Description

Estimate boundaries in a binary image. This function may be used directly with list objects with the format of par2obs output.

Usage

```
BayesBDbinary(obs, inimean, nrun, nburn, J, ordering, mask, slice, outputAll)
```

Arguments

obs	The noisy observation which is a list with the following required elements: • intensity: observed intensity at each pixel. • theta.obs, r.obs: the location of the pixel at which the intensity is observed, using polar coordinates with respect to a reference point. • center: the reference point for polar coords (theta.obs, r.obs).
inimean	a constant to specify the initial mean functions in the Bayesian estimation.
nrun	the number of MCMC samples to keep for estimation.
nburn	the number of initial MCMC samples to discard.
J	truncation number of the Gaussian process kernel. The number of eigenfunctions is $2J + 1$.
ordering	Indicates which Bernoulli distribution has larger success probability: "I", the Bernoulli distribution inside the boundary; "O", the Bernoulli distribution outside the boundary; "N", no ordering information is available.
mask	Logical vector (same length as obs$intensity) to indicate region of interest. Should this data point be included in the analysis?
slice	boolean where TRUE means that slice sampling will be used to sample Fourier basis function coefficients and FALSE means that Metropolis-Hastings will be used instead.
outputAll	boolean controlling the amount of output produced, see value below.

Value

If outputAll is FALSE,

estimate	Posterior mean estimate of image boundary at theta values.
theta	A grid of 200 values on $[0, 2\pi]$ at which to retrun the estimated boundary.
lower, upper	The lower and upper bounds of a 95% uniform credible band for the image boundary.

If outputAll is TRUE, same as above, and additionally,

pi.smp	posterior samples of π_1 and π_2.
coef.smp	posterior samples of Fourier basis function coefficients.

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

See Also

fitBinImage

`BayesBDnormal` *Bayesian boundary estimation for continuous intensity images*

Description

Estimate boundaries in a continuous intensity image.

Usage

```
BayesBDnormal(obs, inimean, nrun, nburn, J, ordering_mu,
ordering_sigma, mask, slice, outputAll)
```

Arguments

`obs`	The noisy observation which is a list with the following required elements: • intensity: observed intensity at each pixel. • theta.obs, r.obs: the location of the pixel at which the intensity is observed, using polar coordinates with respect to a reference point. • center: the reference point for polar coords (theta.obs, r.obs).
`inimean`	a constant to specify the initial mean functions in the Bayesian estimation.
`nrun`	the number of MCMC samples to keep for estimation.
`nburn`	the number of initial MCMC samples to discard.
`J`	truncation number of the Gaussian process kernel. The number of eigenfunctions is $2J + 1$.
`ordering_mu`	Indicates which Gaussian distribution has larger mean intensity: "I", the Gaussian distribution inside the boundary; "O", the Gaussian distribution outside the boundary; "N", no ordering information is available.
`ordering_sigma`	Indicates which Gaussian distribution has larger intensity variance: "I", the Gaussian distribution inside the boundary; "O", the Gaussian distribution outside the boundary; "N", no ordering information is available.
`mask`	Logical vector (same length as obs$intensity) to indicate region of interest. Should this data point be included in the analysis?
`slice`	boolean where TRUE means that slice sampling will be used to sample Fourier basis function coefficients and FALSE means that Metropolis-Hastings will be used instead.
`outputAll`	boolean controlling the amount of output produced, see value below.

Value

If outputAll is FALSE,

`estimate`	Posterior mean estimate of image boundary at theta values.
`theta`	A grid of 200 values on $[0, 2\pi]$ at which to retrun the estimated boundary.

`lower, upper` The lower and upper bounds of a 95% uniform credible band for the image boundary.

If outputAll is TRUE, same as above, and additionally,

`musig.smp` posterior samples of μ_1, μ_2, σ_1, and σ_2.

`coef.smp` posterior samples of Fourier basis function coefficients.

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

See Also

`fitContImage`

`BayesBDshiny` *Boundary detection and shiny integration*

Description

This function produces a local shiny instance for easily producing graphics of boundary detection simulations and real-data analyses.

Usage

```
BayesBDshiny()
```

Examples

```
?BayesBDshiny()
## Not run:
BayesBDshiny()

## End(Not run)
```

besselIs *Bessel Function*

Description

Exponentially scaled Bessel function of first kind.

Usage

```
besselIs(x, nu, expon.scaled = FALSE)
```

Arguments

x	numeric, ≥ 0.
nu	numeric; The order (may be fractional) of the Bessel function.
expon.scaled	This argument does not affect the result; besselIs always returns the exponentially scaled version of the Bessel function of the first kind.

See Also

besselI

dsmError *Dice Similarity Coefficient of boundary estimate*

Description

Utility function to calculate the Dice Similarity Coefficient between true image boundary and estimated image boundary in a simulated image.

Usage

```
dsmError(fit)
```

Arguments

fit	the output of fitBinImage or fitContImage

Examples

```
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
norm.obs = parnormobs(m = 100, mu.in = 4, mu.out = 1, sd.in = 1.5, sd.out = 1, design = 'J',
 center = c(0.5,0.5), gamma.fun)
## Not run:
norm.samp = fitContImage(norm.obs, NULL,NULL,.4, 1000, 0, 10,"I","I",rep(1,10000), FALSE, FALSE)
par(mfrow = c(1,3))
plotBD(norm.samp, 1)
plotBD(norm.samp, 2)
plotBD(norm.samp, 3)
dsmError(norm.samp)

## End(Not run)
```

---

eigenfun *Generate the Fourier series*

Description

eigenfun(n, x) generates the first n fourier series, which are $\{1/\sqrt{2*\pi}, 1/\sqrt{\pi} * cos(x), 1/\sqrt{\pi} * cos(x), ...\}$ evaluated at x.

Usage

```
eigenfun(n,x)
```

Arguments

n	The number of eigenfunctions used to approximate the boundary. It should be a odd number.
x	The argument at which the eigenfunctions are evaluated.

---

ellipse *Generate ellipse boundaries*

Description

Generate general ellipse boundaries with semidiameters a and b, centered at (r0, theta0), with the a axis rotated by phi relative to the polar axis.

Usage

```
ellipse(a, b, r0 = 0, theta0 = 0, phi = 0)
```

Arguments

a, b	semidiameter parameters where a > b.
r0, theta0	the polar form of the center of the ellipse
phi	the angle rotated by the major axis.

Examples

```
gamma.fun = ellipse(a = 0.35, b = 0.25)
theta.plot = seq(from = 0, to = 2*pi, length.out = 200)
x = gamma.fun(theta.plot)*cos(theta.plot)
y = gamma.fun(theta.plot)*sin(theta.plot)
plot(x,y,type = 'l', axes=TRUE, frame.plot=FALSE)
```

---

fitBinImage *Data pre-processing and binary image analysis*

Description

This function can be used to analyze a binary image in .png or .jpeg, or an image represented as a list object in the format of par2obs.

Usage

```
fitBinImage(image, gamma.fun = NULL, center = NULL, inimean = NULL, nrun,
 nburn, J, ordering, mask = NULL, slice, outputAll)
```

Arguments

image	This may be a string representing the path to a .png or .jpeg file, or a list object in the same format as par2obs output, with intensity, r.obs, theta.obs, and center the required list contents.
gamma.fun	This is a function, like triangle2 or ellipse, denoting the true boundary. It is optional and only used when the image input refers to a .png or .jpeg file.
center	This is required if the image input refers to a .png or .jpeg file, otherwise it is unused.
inimean	a constant to specify the initial mean functions in the Bayesian estimation.
nrun	the number of MCMC samples to keep for estimation.
nburn	the number of initial MCMC samples to discard.
J	truncation number of the Gaussian process kernel. The number of eigenfunctions is $2J + 1$.
ordering	Indicates which Bernoulli distribution has larger success probability: "I", the Bernoulli distribution inside the boundary; "O", the Bernoulli distribution outside the boundary; "N", no ordering information is available.

mask	Logical vector (same length as obs$intensity) to indicate region of interest. Should this data point be included in the analysis?
slice	boolean where TRUE means that slice sampling will be used to sample Fourier basis function coefficients and FALSE means that Metropolis-Hastings will be used instead.
outputAll	boolean controlling the amount of output produced, see value below.

Value

output

If outputAll is FALSE,

estimate	Posterior mean estimate of image boundary at theta values.
theta	A grid of 200 values on $[0, 2\pi]$ at which to retrun the estimated boundary.
lower, upper	The lower and upper bounds of a 95% uniform credible band for the image boundary.

If outputAll is TRUE, same as above, and additionally,

pi.smp	posterior samples of π_1 and π_2.
coef.smp	posterior samples of Fourier basis function coefficients.
image	the input image passed to fitBinImage.
obs	the processed image data passed to BayesBDbinary.

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

See Also

par2obs

Examples

```
## Not run:
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
bin.obs = par2obs(m = 100, pi.in = 0.5, pi.out = 0.2,
 design = 'J', center = c(0.5,0.5), gamma.fun)
bin.fit = fitBinImage(image = bin.obs, nrun=1000,
 nburn=1000, J=10, ordering='I', slice = FALSE, outputAll=FALSE)
par(mfrow = c(1,3))
plotBD(bin.fit, 1)
plotBD(bin.fit, 2)
plotBD(bin.fit, 3)

## End(Not run)
```

`fitContImage` *Data pre-processing and continuous image analysis*

Description

This function can be used to analyze a continuous image in .png or .jpeg format, or an image represented as a list object in the format of parnormobs.

Usage

```
fitContImage(image, gamma.fun = NULL, center = NULL, inimean = NULL, nrun,
 nburn, J, ordering_mu, ordering_sigma, mask = NULL, slice, outputAll)
```

Arguments

`image`	This may be a string representing the path to a .png or .jpeg file, or a list object in the same format as par2obs output, with intensity, r.obs, theta.obs, and center the required list contents.
`gamma.fun`	This is a function, like triangle2 or ellipse, denoting the true boundary. It is optional and only used when the image input refers to a .png or .jpeg file.
`center`	This is required if the image input refers to a .png or .jpeg file, otherwise it is unused.
`inimean`	a constant to specify the initial mean functions in the Bayesian estimation.
`nrun`	the number of MCMC samples to keep for estimation.
`nburn`	the number of initial MCMC samples to discard.
`J`	truncation number of the Gaussian process kernel. The number of eigenfunctions is $2J + 1$.
`ordering_mu`	Indicates which Gaussian distribution has larger mean intensity: "I", the Gaussian distribution inside the boundary; "O", the Gaussian distribution outside the boundary; "N", no ordering information is available.
`ordering_sigma`	Indicates which Gaussian distribution has larger intensity variance: "I", the Gaussian distribution inside the boundary; "O", the Gaussian distribution outside the boundary; "N", no ordering information is available.
`mask`	Logical vector (same length as obs$intensity) to indicate region of interest. Should this data point be included in the analysis?
`slice`	boolean where TRUE means that slice sampling will be used to sample Fourier basis function coefficients and FALSE means that Metropolis-Hastings will be used instead.
`outputAll`	boolean controlling the amount of output produced, see value below.

Value

`output`

If outputAll is FALSE,

`estimate`	Posterior mean estimate of image boundary at theta values.
`theta`	A grid of 200 values on $[0, 2\pi]$ at which to retrun the estimated boundary.
`lower, upper`	The lower and upper bounds of a 95% uniform credible band for the image boundary.

If outputAll is TRUE, same as above, and additionally,

`musig.smp`	posterior samples of μ_1, μ_2, σ_1, and σ_2.
`coef.smp`	posterior samples of Fourier basis function coefficients.
`image`	the input image passed to fitContImage.
`obs`	the processed image data passed to BayesBDnormal.

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

See Also

parnormobs

Examples

```
## Not run:
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
norm.obs = parnormobs(m = 100, mu.in = 4, mu.out = 1,
 sd.in = 1.5, sd.out = 1, design = 'J',
 center = c(0.5,0.5), gamma.fun)
norm.samp = fitContImage(image = norm.obs, nrun = 1000, nburn = 0,
 J = 10,ordering_mu = "I",ordering_sigma = "I", slice = FALSE, outputAll = FALSE)
par(mfrow = c(1,3))
plotBD(norm.samp, 1)
plotBD(norm.samp, 2)
plotBD(norm.samp, 3)

## End(Not run)
```

hausdorffError *Hausdorff Error of boundary estimate*

Description

Utility function to calculate the Hausdorff error between true image boudnary and estimated image boundary in a simulated image.

Usage

```
hausdorffError(fit)
```

Arguments

fit	the output of fitBinImage or fitContImage

Examples

```
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
norm.obs = parnormobs(m = 100, mu.in = 4, mu.out = 1, sd.in = 1.5, sd.out = 1, design = 'J',
 center = c(0.5,0.5), gamma.fun)
## Not run:
norm.samp = fitContImage(norm.obs, NULL,NULL,.4, 1000, 0, 10,"I","I",rep(1,10000), FALSE, FALSE)
par(mfrow = c(1,3))
plotBD(norm.samp, 1)
plotBD(norm.samp, 2)
plotBD(norm.samp, 3)
hausdorffError(norm.samp)

## End(Not run)
```

lebesgueError *Lebesgue Error of boundary estimate*

Description

Utility function to calculate the Lebesgue error of the symmetric difference between true image boudnary and estimated image boundary in a simulated image.

Usage

```
lebesgueError(fit)
```

Arguments

fit	the output of fitBinImage or fitContImage

Examples

```
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
norm.obs = parnormobs(m = 100, mu.in = 4, mu.out = 1, sd.in = 1.5, sd.out = 1, design = 'J',
 center = c(0.5,0.5), gamma.fun)
## Not run:
norm.samp = fitContImage(norm.obs, NULL,NULL,.4, 1000, 0, 10,"I","I",rep(1,10000), FALSE, FALSE)
par(mfrow = c(1,3))
plotBD(norm.samp, 1)
plotBD(norm.samp, 2)
plotBD(norm.samp, 3)
lebesgueError(norm.samp)

## End(Not run)
```

---

par2obs *Simulate binary intensity images*

Description

The generated data $Y\ Bernoulli(p)$ is the image intensity, where the success probability p is determined by whether the location X, given by polar coords (r, θ), is inside the boundary or not.

Usage

```
par2obs(m, pi.in, pi.out, design, center, gamma.fun)
```

Arguments

m	$m * m$ observations will be generated over the unit square centered at (0,0).
pi.in	The success probability, $P(Y_i = 1)$, where Y_i is intensity of pixel i if the location is inside the boundary.
pi.out	The success probability, $P(Y_i = 1)$, where Y_i is intensity of pixel i if the location is outside the boundary.
design	Taking values: 'D' for deterministic (equally-spaced grid) design, 'U' for completely uniformly random, or 'J' for jitteredly random design.
center	a two-dimensional vector of Euclidean coordinates (x,y) of the reference point.
gamma.fun	The function to generate boundaries, see ellipse or triangle2.

Examples

```
set.seed(2015)
# use ellipse boundary
gamma.fun = ellipse(a = 0.35, b = 0.25)
obs = par2obs(m = 100, pi.in = 0.5, pi.out = 0.2, design = 'J', center = c(0.5,0.5), gamma.fun)
plotBD(obs)
```

parnormobs *Simulate Gaussian intensity images*

Description

The generated data $Y\ N(\mu, \sigma)$ is the image intensity, where the params (μ, σ) are determined by whether the location X, given by polar coords (r, θ), is inside the boundary or not.

Usage

```
parnormobs(m, mu.in, mu.out, sd.in, sd.out, design, center, gamma.fun)
```

Arguments

m	$m * m$ observations will be generated over the unit square centered at (0,0).
mu.in	The mean intensity for pixels inside the image boundary.
mu.out	The mean intensity for pixels outside the image boundary.
sd.in	The standard deviation of intensity for pixels inside the image boundary.
sd.out	The standard deviation of intensity for pixels outside the image boundary.
design	Taking values: 'D' for deterministic (equally-spaced grid) design, 'U' for completely uniformly random, or 'J' for jitteredly random design.
center	a two-dimensional vector of Euclidean coordinates (x,y) of the reference point.
gamma.fun	The function to generate boundaries, see ellipse or triangle2.

Examples

```
set.seed(2015)
# use ellipse boundary
gamma.fun = ellipse(a = 0.35, b = 0.25)
obs = parnormobs(m = 100, mu.in = 1, sd.in = 1, mu.out = 0, sd.out = 1, design = 'J',
 center = c(0.5,0.5), gamma.fun)
plotBD(obs)
```

plotBD *Visualization of posterior boundary estimates and data*

Description

Produces plots of image data, posterior boundary estimates,
and 95

Usage

```
plotBD(fitted.image, plot.type)
```

Arguments

`fitted.image`	An object containing the output of either fitBinImage or fitContImage.
`plot.type`	takes values 1, 2 or 3: 1 to plot the data only; 2 to plot the estimate and 95% uniform credible bands; and 3 to plot the data with the estimated boundary overlaid.

Value

There is no output to console; plotBD produces plots.

References

Li, M. and Ghosal, S.(2015) "Bayesian Detection of Image Boundaries." arXiv 1508.05847.

Examples

```
## Not run:
set.seed(12345)
gamma.fun = ellipse(a = 0.35, b = 0.25)
bin.obs = par2obs(m = 100, pi.in = 0.5, pi.out = 0.2,
 design = 'J', center = c(0.5,0.5), gamma.fun)
bin.fit = fitBinImage(image = bin.obs, nrun=1000, nburn=1000,
 J=10, ordering='I', slice = FALSE, outputAll=FALSE)
par(mfrow = c(1,3))
plotBD(bin.fit, 1)
plotBD(bin.fit, 2)
plotBD(bin.fit, 3)

## End(Not run)
```

`rectToPolar` *Simulate binary intensity images*

Description

Utility function to convert rectangular (Euclidean) coordinates to polar coordinates.

Usage

```
rectToPolar(x,y)
```

Arguments

`x`	x-axis in rectangular coords
`y`	y-axis in rectangular coords

Examples

```
x = runif(100,-1,1)
y = runif(100,-1,1)
polar_coords = rectToPolar(x,y)
```

`triangle2` *Generate triangle boundaries*

Description

Generate boundaries of equilateral triangles of height S.

Usage

```
triangle2(S)
```

Arguments

`S`	height of the generated triangle.

Examples

```
gamma.fun = triangle2(0.5)
theta.plot = seq(from = 0, to = 2*pi, length.out = 200)
x = gamma.fun(theta.plot)*cos(theta.plot)
y = gamma.fun(theta.plot)*sin(theta.plot)
plot(x,y,type = 'l', axes=TRUE, frame.plot=FALSE)
```

`tum1` *Brain tumor .png file*

Description

This is an example of image data for a Gaussian-noised image and can be used as input to fitContImage.

`tum2` *Brain tumor .png file*

Description

This is an example of image data for a Gaussian-noised image and can be used as input to fitContImage.

`unisliceL` *Slice Sampler*

Description

Performs slice sampling for lambda parameter. This is a helper function used by BayesBDbinary and BayesBDnormal.

Usage

```
unisliceL(x0, gx0, i_J, tauini, anini, alphalambda, betalambda, lambdaini)
```

Arguments

`x0`	initial value of lambda.
`gx0`	likelihood at x0.
`i_J`	number of basis functions (2*i_J+1).
`tauini`	tau parameter in covariance function.
`anini`	2J+1 vector of eigenfunction coefficients.
`alphalambda`	hyperparameter for sampling tau, usually 1.
`betalambda`	hyperparameter for sampling tau, usually 1.
`lambdaini`	current value of lambda.

Index