Package ‘serrsBayes’

June 28, 2021

Type Package

Title Bayesian Modelling of Raman Spectroscopy

Version 0.5-0

Date 2021-06-28

Description Sequential Monte Carlo (SMC) algorithms for fitting a generalised additive mixed model (GAMM) to surface-enhanced resonance Raman spectroscopy (SERRS), using the method of Moores et al. (2016) <arXiv:1604.07299>. Multivariate observations of SERRS are highly collinear and lend themselves to a reduced-rank representation. The GAMM separates the SERRS signal into three components: a sequence of Lorentzian, Gaussian, or pseudo-Voigt peaks; a smoothly-varying baseline; and additive white noise. The parameters of each component of the model are estimated iteratively using SMC. The posterior distributions of the parameters given the observed spectra are represented as a population of weighted particles.

License GPL (>= 2) | file LICENSE

URL https://github.com/mooresm/serrsBayes,
       https://mooresm.github.io/serrsBayes/

BugReports https://github.com/mooresm/serrsBayes/issues

Depends R (>= 3.5.0), Matrix, truncnorm, splines

Imports Rcpp (>= 0.11.3), methods

LinkingTo Rcpp, RcppEigen

Suggests testthat, knitr, rmarkdown, Hmisc

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation yes

Author Matt Moores [aut, cre] (<https://orcid.org/0000-0003-4531-3572>),
    Jake Carson [aut] (<https://orcid.org/0000-0002-7896-0971>),
    Benjamin Moskowitz [ctb],
    Kirsten Gracie [dct],
    Karen Faulds [dct] (<https://orcid.org/0000-0002-5567-7399>),
Mark Girolami [aut],
Engineering and Physical Sciences Research Council [fnd] (EPSRC
programme grant ref: EP/L014165/1),
University of Warwick [cph]

Maintainer Matt Moores <mmoores@gmail.com>

Repository CRAN

Date/Publication 2021-06-28 12:20:02 UTC

R topics documented:

computeLogLikelihood .............................................. 3
copyLogProposals .................................................. 4
effectiveSampleSize ................................................ 4
fitSpectraMCMC ..................................................... 5
fitSpectraSMC ......................................................... 6
fitVoigtIBIS .......................................................... 7
fitVoigtPeaksSMC ...................................................... 9
getBsplineBasis ....................................................... 10
getVoigtParam ........................................................ 11
IsTamra ................................................................. 12
marginalMetropolisUpdate ........................................ 12
methanol ................................................................. 13
mhUpdateVoigt ......................................................... 14
mixedVoigt ............................................................. 15
resampleParticles .................................................... 16
residualResampling .................................................. 17
result ................................................................. 17
result2 ................................................................. 18
reWeightParticles .................................................... 19
serrsBayes ............................................................ 20
sumDexp ............................................................... 22
sumDlogNorm .......................................................... 22
sumDnorm .............................................................. 23
weightedGaussian .................................................... 24
weightedLorentzian .................................................. 25
weightedMean ........................................................ 25
weightedVariance ..................................................... 26

Index ................................................................. 27
computeLogLikelihood

Compute the log-likelihood.

Description

This is an internal function that is only exposed on the public API for unit testing purposes. It computes the log-likelihood of the spline and the noise, once the spectral signature has been subtracted from the observed data. Thus, it can be used with either Lorentzian, Gaussian, or pseudo-Voigt broadening functions.

Usage

```r
computeLogLikelihood(
  obsi, lambda, prErrNu, prErrSS, basisMx, eigVal, precMx, xTx, aMx, ruMx
)
```

Arguments

- `obsi` Vector of residuals after the spectral signature has been subtracted.
- `lambda` smoothing parameter of the penalised B-spline.
- `prErrNu` hyperparameter of the additive noise
- `prErrSS` hyperparameter of the additive noise
- `basisMx` Matrix of B-spline basis functions
- `eigVal` eigenvalues of the Demmler-Reinsch factorisation
- `precMx` precision matrix for the spline
- `xTx` sparse matrix cross-product
- `aMx` orthogonal matrix A from the Demmler-Reinsch factorisation
- `ruMx` product of Ru from the Demmler-Reinsch factorisation

Value

The logarithm of the likelihood.
copyLogProposals  

Initialise the vector of Metropolis-Hastings proposals.

Description

This is an internal function that is only exposed on the public API for unit testing purposes.

Usage

```r
copyLogProposals(nPK, T_Prop_Theta)
```

Arguments

- `nPK` number of Raman peaks in the spectral signature
- `T_Prop_Theta` Vector of logarithms of the MH proposals

Value

Vector of proposals

---

effectiveSampleSize  

Compute the effective sample size (ESS) of the particles.

Description

The ESS is a "rule of thumb" for assessing the degeneracy of the importance distribution:

\[
ESS = \frac{\left(\sum_{q=1}^{Q} w_q\right)^2}{\sum_{q=1}^{Q} w_q^2}
\]

Usage

```r
effectiveSampleSize(log_weights)
```

Arguments

- `log_weights` logarithms of the importance weights of each particle.

Value

the effective sample size, a scalar between 0 and Q

References

fitSpectraMCMC

Examples

```r
x <- runif(100)
effectiveSampleSize(log(x))
```

Description

Fit the model using Markov chain Monte Carlo.

Usage

```r
fitSpectraMCMC(wl, spc, peakWL, lPriors, sd_mh, niter = 10000, nchains = 4)
```

Arguments

- `wl`: Vector of `nwl` wavenumbers at which the spectra are observed.
- `spc`: `n_y * nwl` Matrix of observed Raman spectra.
- `peakWL`: Vector of locations for each peak (cm^-1).
- `lPriors`: List of hyperparameters for the prior distributions.
- `sd_mh`: Vector of `2 * npeaks` bandwidths for the random walk proposals.
- `niter`: Number of MCMC iterations per chain.
- `nchains`: Number of concurrent MCMC chains.

Value

A list containing MCMC samples for the model parameters:

- `amplitude`: `niter * nchains * npeaks` Array of amplitudes.
- `scale`: `niter * nchains * npeaks` Array of scale parameters.
- `sigma`: `niter * nchains` Matrix of standard deviations.
- `n_acc`: The number of RWMH proposals that were accepted.

See Also

`marginalMetropolisUpdate`
Examples

```r
wavenumbers <- seq(200, 600, by=10)
spectra <- matrix(nrow=1, ncol=length(wavenumbers))
peakLocations <- c(300, 500)
peakAmplitude <- c(10000, 4000)
peakScale <- c(10, 15)
signature <- weightedLorentzian(peakLocations, peakScale, peakAmplitude, wavenumbers)
baseline <- 1000*cos(wavenumbers/200) + 2*wavenumbers
spectra[1,] <- signature + baseline + rnorm(length(wavenumbers), 0, 200)
lPriors <- list(scale.mu=log(11.6) - (0.4^2)/2, scale.sd=0.4, bl.smooth=10^11, bl.knots=20,
               amp.mu=5000, amp.sd=5000, noise.sd=200, noise.nu=4)
result <- fitSpectraMCMC(wavenumbers, spectra, peakLocations, lPriors, rw_bw, 500)
result$n_acc
```

fitSpectraSMC  
Fit the model using Sequential Monte Carlo (SMC).

Description

Fit the model using Sequential Monte Carlo (SMC).

Usage

```r
fitSpectraSMC(
  wl,            # Vector of nwl wavenumbers at which the spectra are observed.
  spc,           # n_y * nwl Matrix of observed Raman spectra.
  peakWL,        # Vector of locations for each peak (cm^-1)
  lPriors,       # List of hyperparameters for the prior distributions.
  conc = rep(1, nrow(spc)),  # Vector of n_y nanomolar (nM) dye concentrations for each observation.
  npart = 10000,  # number of SMC particles to use for the importance sampling distribution.
  rate = 0.9,     # the target rate of reduction in the effective sample size (ESS).
  minESS = npart/2,  # minimum effective sample size, below which the particles are resampled.
  destDir = NA    # destination directory to save intermediate results (for long-running computations)
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wl</code></td>
<td>Vector of nwl wavenumbers at which the spectra are observed.</td>
</tr>
<tr>
<td><code>spc</code></td>
<td>n_y * nwl Matrix of observed Raman spectra.</td>
</tr>
<tr>
<td><code>peakWL</code></td>
<td>Vector of locations for each peak (cm^-1)</td>
</tr>
<tr>
<td><code>lPriors</code></td>
<td>List of hyperparameters for the prior distributions.</td>
</tr>
<tr>
<td><code>conc</code></td>
<td>Vector of n_y nanomolar (nM) dye concentrations for each observation.</td>
</tr>
<tr>
<td><code>npart</code></td>
<td>number of SMC particles to use for the importance sampling distribution.</td>
</tr>
<tr>
<td><code>rate</code></td>
<td>the target rate of reduction in the effective sample size (ESS).</td>
</tr>
<tr>
<td><code>minESS</code></td>
<td>minimum effective sample size, below which the particles are resampled.</td>
</tr>
<tr>
<td><code>destDir</code></td>
<td>destination directory to save intermediate results (for long-running computations)</td>
</tr>
</tbody>
</table>
Value

- weights: Vector of importance weights for each particle.
- beta: Matrix of regression coefficients for the amplitudes.
- scale: Matrix of scale parameters.
- sigma: Vector of standard deviations.
- alpha: Array of spline coefficients for the baseline.
- basis: A dense matrix containing the values of the basis functions.
- expFn: Matrix containing the spectral signature.
- ess: Vector containing the effective sample size (ESS) at each SMC iteration.
- logEvidence: Vector containing the logarithm of the model evidence (marginal likelihood).
- accept: Vector containing the Metropolis-Hastings acceptance rate at each SMC iteration.
- sd.mh: Matrix of random walk MH bandwidths at each SMC iteration.

References


Examples

```r
wavenumbers <- seq(200, 600, by=10)
spectra <- matrix(nrow=1, ncol=length(wavenumbers))
peakLocations <- c(300, 500)
peakAmplitude <- c(10000, 4000)
peakScale <- c(10, 15)
signature <- weightedLorentzian(peakLocations, peakScale, peakAmplitude, wavenumbers)
baseline <- 100*cos(wavenumbers/200) + 2*wavenumbers
spectra[1,] <- signature + baseline + rnorm(length(wavenumbers), 0, 200)

lPriors <- list(scale.mu=log(11.6) - (0.4^2)/2, scale.sd=0.4, bl.smooth=10^11, bl.knots=20,
                beta.mu=5000, beta.sd=5000, noise.sd=200, noise.nu=4)

## Not run:
result <- fitSpectraSMC(wavenumbers, spectra, peakLocations, lPriors, npart=500)
## End(Not run)
```

Description

Fit the model with Voigt peaks using iterated batch importance sampling (IBIS).
Usage

```r
fitVoigtIBIS(
  wl,
  spc,
  n,
  lResult,
  conc = rep(1, nrow(spc)),
  batch = rep(1, nrow(spc)),
  npart = 10000,
  rate = 0.9,
  mcAR = 0.234,
  mcSteps = 20,
  minESS = npart/2,
  minPart = npart,
  destDir = NA
)
```

Arguments

- `wl`: Vector of `nwl` wavenumbers at which the spectra are observed.
- `spc`: `n_y * nwl` Matrix of observed Raman spectra.
- `n`: Index of the new observation.
- `lResult`: List of results from the previous call to “fitVoigtPeaksSMC” or “fitVoigtIBIS”.
- `conc`: Vector of `n_y` nanomolar (nM) dye concentrations for each observation.
- `batch`: Identifies to which batch each observation belongs.
- `npart`: Number of SMC particles to use for the importance sampling distribution.
- `rate`: The target rate of reduction in the effective sample size (ESS).
- `mcAR`: Target acceptance rate for the MCMC kernel.
- `mcSteps`: Number of iterations of the MCMC kernel.
- `minESS`: Minimum effective sample size, below which the particles are resampled.
- `minPart`: Target number of unique particles for the MCMC iterations.
- `destDir`: Destination directory to save intermediate results (for long-running computations).

References

**fitVoigtPeaksSMC**

*Fit the model with Voigt peaks using Sequential Monte Carlo (SMC).*

**Description**

Fit the model with Voigt peaks using Sequential Monte Carlo (SMC).

**Usage**

```r
fitVoigtPeaksSMC(
  wl,  
  spc,  
  lPriors,  
  conc = rep(1, nrow(spc)),  
  npart = 10000,  
  rate = 0.9,  
  mcAR = 0.234,  
  mcSteps = 20,  
  minESS = npart/2,  
  destDir = NA,  
  minPart = npart  
)
```

**Arguments**

- `wl`: Vector of `nwl` wavenumbers at which the spectra are observed.
- `spc`: `n_y` x `nwl` Matrix of observed Raman spectra.
- `lPriors`: List of hyperparameters for the prior distributions.
- `conc`: Vector of `n_y` nanomolar (nM) dye concentrations for each observation.
- `npart`: Number of SMC particles to use for the importance sampling distribution.
- `rate`: The target rate of reduction in the effective sample size (ESS).
- `mcAR`: Target acceptance rate for the MCMC kernel.
- `mcSteps`: Number of iterations of the MCMC kernel.
- `minESS`: Minimum effective sample size, below which the particles are resampled.
- `destDir`: Destination directory to save intermediate results (for long-running computations).
- `minPart`: Target number of unique particles for the MCMC iterations.

**Examples**

```r
wavenumbers <- seq(200,600,by=10)
spectra <- matrix(nrow=1,ncol=length(wavenumbers))
peakLocations <- c(300,500)
peakAmplitude <- c(10000,4000)
```
peakScale <- c(10, 15)
signature <- weightedLorentzian(peakLocations, peakScale, peakAmplitude, wavenumbers)
baseline <- 1000*cos(wavenumbers/200) + 2*wavenumbers
spectra[,1] <- signature + baseline + rnorm(length(wavenumbers),0,200)
lPriors <- list(scaG.mu=log(11.6) - (0.4^2)/2, scaG.sd=0.4, scal.mu=log(11.6) - (0.4^2)/2,
scal.sd=0.4, bl.smooth=5, bl.knots=20, loc.mu=peakLocations, loc.sd=c(5,5),
beta.mu=c(5000,5000), beta.sd=c(5000,5000), noise.sd=200, noise.nu=4)
## Not run:
result <- fitVoigtPeaksSMC(wavenumbers, spectra, lPriors, npart=50, mcSteps=1)
## End(Not run)

getBsplineBasis
getBsplineBasis(V, n.b, pen, prec = 1e-08)

A vector of wavenumbers, Δ\\tilde{ν}.
the number of basis functions to use.
the smoothing penalty hyperparameter.
a constant scale factor.

a list containing:

basism A dense nwl by n.b matrix containing the values of the basis functions.
precision A sparse n.b by n.b dsCMatrix, the inverse of the prior covariance.
distance The distance between each knot (cm⁻¹).
knots The knot locations.

getVoigtParam

See Also

sparseMatrix

getVoigtParam

Compute the pseudo-Voigt mixing ratio for each peak.

Description

Calculates the mixing parameter $\eta_j$ from the scales of the Gaussian/Lorentzian components.

Usage

getVoigtParam(scale_G, scale_L)

Arguments

scale_G Vector of standard deviations $\sigma_j$ of the Gaussian components.
scale_L Vector of scale parameters $\phi_j$ of the Lorentzian components.

Details

First, calculate a polynomial average of the scale parameters according to the approximation of Thompson et al. (1987):

$$f_{G,L} = (\sigma_j^5 + 2.69\sigma_j^4\phi_j + 2.42\sigma_j^3\phi_j^2 + 4.47\sigma_j^2\phi_j^3 + 0.07\sigma_j\phi_j^4 + \phi_j^5)^{1/5}$$

Then the Voigt mixing parameter $\eta_j$ is defined as:

$$\eta_j = 1.36\frac{\phi_j}{f_{G,L}} - 0.47\left(\frac{\phi_j}{f_{G,L}}\right)^2 + 0.11\left(\frac{\phi_j}{f_{G,L}}\right)^3$$

Value

The Voigt mixing weights for each peak, between 0 (Gaussian) and 1 (Lorentzian).

References

**lsTamra**

*Surface-enhanced Raman spectrum of tetramethylrhodamine+DNA (T20)*

**Description**

Surface-enhanced Raman spectram of tetramethylrhodamine+DNA (T20)

**Usage**

lsTamra

**Format**

A list containing 2 variables:

- **wavenumbers** a numeric Vector of 2401 wavenumbers (cm⁻¹)
- **wavenumbers** a 1 * 2401 Matrix of intensity values (a.u.)

**marginalMetropolisUpdate**

*Update all of the parameters using a single Metropolis-Hastings step.*

**Description**

Updates all of the parameters using a single Metropolis-Hastings step, such that the baseline cancels out in the MH ratio, using the marginalisation identity of Chib (1995). If `npart > 1`, then multiple MCMC chains will be executed independently in parallel using OpenMP. This means that all functions used for the proposal distributions and to evaluate the MH ratio need to be thread-safe. Specifically, no calls to R::rnorm, R::dnorm, nor their Rcpp equivalents, can be made from within the parallel portion of the code.

**Usage**

marginalMetropolisUpdate(
    spectra,
    n,
    conc,
    wavelengths,
    peakWL,
    betaMx,
    scaleMx,
    sigma,
    expMx,
    baselines,
    sd_mh,
    priors
)
Arguments

- `spectra` \( n \times nwl \) Matrix of observed Raman spectra.
- `n` number of observations to use in calculating the likelihood
- `conc` Vector of \( n \) nanomolar (nM) dye concentrations
- `wavelengths` Vector of \( nwl \) wavenumbers at which the spectra are observed.
- `peakWL` Vector of locations for each peak (cm\(^{-1}\))
- `betaMx` \( npeaks \times npart \) Matrix of regression coefficients to update.
- `scaleMx` \( npeaks \times npart \) Matrix of scale parameters to update.
- `sigma` Vector of \( npart \) standard deviations to update.
- `expMx` \( nwl \times npart \) Matrix of expectations of the Lorentzian or Gaussian function.
- `baselines` \( nKnots \times n_y \times npart \) Array of smoothing splines.
- `sd_mh` Vector of \( 2 \times npeaks \) bandwidths for the random walk proposals.
- `priors` List of hyperparameters for the prior distributions.

Value

The number of RWMH proposals that were accepted.

References


---

methanol

Raman spectrum of methanol (CH3OH)

Description

Raman spectrum of methanol (CH3OH)

Usage

methanol

Format

A list containing 2 variables:

- `wavenumbers` a numeric Vector of 331 wavenumbers (cm\(^{-1}\))
- `wavenumbers` a 1 \( \times 331 \) Matrix of intensity values (a.u.)
mhUpdateVoigt

Description

Updates all of the parameters (location, amplitude, std. dev., and scale) using a single Metropolis-Hastings step, such that the baseline cancels out in the MH ratio, using the marginalisation identity of Chib (1995). Note: if npart > 1, then multiple MCMC chains will be executed independently in parallel using OpenMP. This means that all functions used for the proposal distributions and to evaluate the MH ratio need to be thread-safe. Specifically, no calls to R::rnorm, R::dnorm, nor their Rcpp equivalents, can be made from within the parallel portion of the code.

Usage

mhUpdateVoigt(
spectra,  
n,  
kappa,  
conc,  
wavenum,  
thetaMx,  
logThetaMx,  
mhChol,  
priors
)

Arguments

spectra n_y * nwl Matrix of observed Raman spectra.
n number of observations to use in calculating the likelihood.
kappa likelihood tempering parameter.
conc Vector of n_y nanomolar (nM) dye concentrations
wavenum Vector of nwl wavenumbers at which the spectra are observed.
thetaMx (4+npeaks*4) x npart Matrix of parameter values for each peak.
logThetaMx (4+npeaks*4) x npart Matrix of logarithms of the parameters.
mhChol lower-triangular Cholesky factorisation of the covariance matrix for the random walk proposals.
priors List of hyperparameters for the prior distributions.

Value

The number of RWMH proposals that were accepted.
mixedVoigt

References


mixedVoigt

Compute the spectral signature using Voigt peaks.

Description

Calculates the value of the pseudo-Voigt broadening function at the given wavenumbers, given the parameters of the peaks. This function is thread-safe.

Usage

mixedVoigt(location, scale_G, scale_L, amplitude, wavenum)

Arguments

location Vector of location parameters of the peaks (cm⁻¹)
scale_G Vector of standard deviations σ_j of the Gaussian components.
scale_L Vector of scale parameters φ_j of the Lorentzian components.
amplitude Vector of amplitudes of the peaks (a.u.)
wavenum Vector of wavenumbers at which to compute the function.

Value

The value of the pseudo-Voigt function at the given wavenumbers.

References


Examples

Cal_V <- seq(300,400,by=5)
loc <- c(320,350,375)
scG <- c(10,5,1)
scL <- c(3,20,7)
amp <- c(100,500,200)
mixedVoigt(loc,scG,scL,amp,Cal_V)
resampleParticles  

Resample in place to avoid expensive copying of data structures, using a permutation of the ancestry vector.

Description

Resample in place to avoid expensive copying of data structures, using a permutation of the ancestry vector.

Usage

resampleParticles(log_weights, ampMx, scaleMx, peaks, baselines, n_y, nwl)

Arguments

log_weights  
logarithms of the importance weights of each particle

ampMx  
npeaks x npart Matrix of amplitudes for each particle.

scaleMx  
npeaks x npart Matrix of scale parameters for each particle.

peaks  
nwl x npart Matrix containing the expectation of the Lorentzian mixture.

baselines  
nwl x n_y x npart Array of smoothing splines.

n_y  
number of observations

nwl  
number of wavenumbers

Value

Vector of indices to the parents of the resampled particles.

References


See Also

residualResampling
residualResampling

Compute an ancestry vector for residual resampling of the SMC particles.

Description

Compute an ancestry vector for residual resampling of the SMC particles.

Usage

residualResampling(log_wt)

Arguments

log_wt logarithms of the importance weights of each particle.

Value

Vector of indices to the particles that will be propagated forward to the next generation (i.e. the parents)

References


result SMC particles for TAMRA+DNA (T20)

Description

Posterior distribution for pseudo-Voigt parameters, obtained by running ‘fitVoigtPeaksSMC‘ on a spectrum from Gracie et al. (Anal. Chem., 2016). 1000 SMC particles with 32 peaks. For details, see the vignette.

Usage

result
Format

A list containing 15 variables:

- **weights** normalised importance weights for each particle
- **location** location parameters of 32 peaks
- **beta** amplitudes of 32 peaks
- **scale_G** scale of the Gaussian (RBF) broadening
- **scale_L** scale of the Lorentzian (Cauchy) broadening
- **sigma** standard deviation of the additive white noise
- **lambda** smoothing parameter of the cubic B-splines
- **priors** List of informative priors
- **ess** history of the effective sample size
- **kappa** history of the likelihood tempering
- **accept** history of Metropolis-Hastings acceptance rates
- **mhSteps** history of Metropolis-Hastings steps
- **times** history of times for each SMC iteration
- **time** computation time taken by the SMC algorithm

---

**result2**

*SMC particles for methanol (CH3OH)*

Description

Posterior distribution for pseudo-Voigt parameters, obtained by running `fitVoigtPeaksSMC` on a Raman spectrum of methanol with 4 peaks. For details, refer to the vignette.

Usage

`result2`

Format

A list containing 15 variables.
reWeightParticles  

Update the importance weights of each particle.

**Description**

Update the importance weights of each particle.

**Usage**

reWeightParticles(
    spectra,
    peaks,
    baselines,
    i,
    start,
    sigma,
    old_weights,
    alpha,
    idx
)

**Arguments**

- **spectra**  
  n_y * nwl Matrix of observed Raman spectra.
- **peaks**  
  nwl * npart Matrix containing the spectral signatures for each observation.
- **baselines**  
  nwl * npart Matrix containing the current values of the baselines.
- **i**  
  index of the current observation to use in calculating the likelihood
- **start**  
  index of the next wavelength to use in calculating the likelihood, permuted by idx
- **sigma**  
  Vector of npart standard deviations for each particle.
- **old_weights**  
  logarithms of the importance weights of each particle.
- **alpha**  
  the target learning rate for the reduction in effective sample size (ESS).
- **idx**  
  permutation of the indices of the wavelengths.

**Value**

a List containing:

- **ess**  
  The effective sample size, after reweighting.
- **weights**  
  Vector of updated importance weights.
- **index**  
  index of the last wavelength used.
- **evidence**  
  SMC estimate of the logarithm of the model evidence.
References


serrsBayes  
Bayesian modelling and quantification of Raman spectroscopy

Description

This R package implements sequential Monte Carlo (SMC) algorithms for fitting a generalised additive mixed model (GAMM) to Raman spectra. These multivariate observations are highly collinear and lend themselves to a reduced-rank representation. The GAMM separates the hyperspectral signal into three components: a sequence of Lorentzian or Gaussian peaks; a smoothly-varying baseline; and zero-mean, additive white noise. The parameters of each component of the model are estimated iteratively using SMC. The posterior distributions of the parameters given the observed spectra are represented as a population of weighted particles.

Details

Raman spectroscopy can be used to identify molecules by the characteristic scattering of light from a laser. The pattern of peaks in a Raman spectrum corresponds to the vibrational modes of the molecule. The shift in wavenumber of the photons is proportional to the change in energy state, which is reflected in the locations of the peaks. Surface-enhanced Raman scattering (SERS) is a technique that amplifies the Raman signal using metallic substrates, such as nanoparticles. The laser can also be tuned to the resonant frequency of the molecule, which is known as surface-enhanced resonance Raman scattering (SERRS). Under controlled experimental conditions, the amplitudes of the peaks are linearly related to the concentration of the molecule, from the limit of detection (LOD) up to monolayer coverage of the nanoparticle surface.

The GAMM represents the peaks and baseline as continuous functions. The background fluorescence is modelled using a penalised cubic spline, while the peaks are an additive mixture of squared exponential (Gaussian) or Lorentzian (Cauchy) kernels:

\[ Y = \sum_{m=1}^{M} \alpha_{i,m} B_m(\nu_j) + \sum_{p=1}^{P} s(\nu_j | l_p, A_p, \phi_p) + \epsilon_{i,j} \]

where \( Y \) is a matrix of hyperspectral observations \( y_{i,j} \) that have been discretised at wavenumbers \( \nu_j \); \( B_m \) are the \( M \) spline basis functions with coefficients \( \alpha_{i,m} \); \( s(\nu_j | l_p, A_p, \phi_p) \) are the radial basis functions for each peak, with location \( l_p \), amplitude \( A_p \), and scale \( \phi_p \) parameters. \( \epsilon_{i,j} \) is assumed to be zero mean, additive white noise with constant variance \( \sigma^2 \).

This model-based approach accounts for differences in resolution and experimental conditions, enabling comparison and alignment of heterogeneous spectra. The relationship between concentration and peak intensity can be quantified by fitting a Bayesian functional regression:

\[ A_p = c_i \beta_p \]
where $c_i$ is the nanomolar (nM) concentration of the molecule in the $i$th spectrum, $c_{LOD} < c_i <= c_{MLC}$. The regression model produces highest posterior density (HPD) intervals for the limit of detection of each peak. A consistent, unbiased estimate of the model evidence (also known as the marginal likelihood) is also computed. This can be used to evaluate whether Gaussian or Lorentzian peaks are a better fit to the data.

**Author(s)**

M. T. Moores, J. Carson & M. Girolami

Maintainer: Matt Moores <moores@gmail.com>

**References**

Moores, Gracie, Carson, Faulds, Graham & Girolami "Bayesian modelling and quantification of Raman spectroscopy," arXiv preprint

**Examples**

```r
# simulate some data with known parameter values
wavenumbers <- seq(700, 1400, by=2)
spectra <- matrix(nrow=1, ncol=length(wavenumbers))
peakLocations <- c(840, 960, 1140, 1220, 1290)
peakAmplitude <- c(11500, 2500, 4000, 3000, 2500)
peakScale <- c(10, 15, 20, 10, 12)
signature <- weightedLorentzian(peakLocations, peakScale, peakAmplitude, wavenumbers)
baseline <- 1000*cos(wavenumbers/200) + 2*wavenumbers
spectra[1,] <- signature + baseline + rnorm(length(wavenumbers), 0, 200)
plot(wavenumbers, spectra[1,], type="l", xlab="Raman offset", ylab="intensity")
lines(wavenumbers, baseline, col=2, lty=4)
lines(wavenumbers, baseline + signature, col=4, lty=2)

# fit the model using SMC
lPriors <- list(scale.mu=log(11.6) - (0.4^2)/2, scale.sd=0.4, bl.smooth=10^11, bl.knots=50,
beta.mu=5000, beta.sd=5000, noise.sd=200, noise.nu=4)
## Not run:
## takes approx. 1 minute for 100 SMC iterations with 10,000 particles
result <- fitSpectraSMC(wavenumbers, spectra, peakLocations, lPriors)
plot.ts(result$ess, xlab="SMC iterations", ylab="ESS")

# sample 200 particles from the posterior distribution
samp.idx <- sample.int(length(result$weights), 200, prob=result$weights)
plot(wavenumbers, spectra[1,], type='l', xlab="Raman offset", ylab="intensity")
for (pt in samp.idx) {
  bl.est <- result$basis %*% result$alpha[,1,pt]
  lines(wavenumbers, bl.est, col="#C3000009")
  lines(wavenumbers, bl.est + result$expFn[pt,], col="#0000C309")
}
```

## End(Not run)
sumDexp

*Sum log-likelihoods of i.i.d. exponential.*

**Description**

This is an internal function that is only exposed on the public API for unit testing purposes.

**Usage**

`sumDexp(x, rate)`

**Arguments**

- `x`: Vector of i.i.d. exponential random variables
- `rate`: parameter of the exponential distribution

**Details**

The sum of the log-likelihoods (log of the product of the likelihoods) for independent, identically-distributed, exponential random variables. Note: this Rcpp function is thread-safe, unlike the equivalent alternatives.

**Value**

log-likelihood of x

**See Also**

`sum(dexp(x, rate, log=TRUE))`

---

sumDlogNorm

*Sum log-likelihoods of i.i.d. lognormal.*

**Description**

This is an internal function that is only exposed on the public API for unit testing purposes.

**Usage**

`sumDlogNorm(x, meanlog, sdlog)`

**Arguments**

- `x`: Vector of i.i.d. lognormal random variables
- `meanlog`: mean of the distribution on the log scale
- `sdlog`: standard deviation on the log scale
Details

The sum of the log-likelihoods (log of the product of the likelihoods) for independent, identically-distributed, lognormal random variables. Note: this Rcpp function is thread-safe, unlike the equivalent alternatives.

Value

log-likelihood of x

See Also

sum(dlnorm(x,meanlog,sdlog,log=TRUE))

Examples

x <- rlnorm(100)
sumDlogNorm(x,0,1)

---

sumDnorm

Sum log-likelihoods of Gaussian.

Description

This is an internal function that is only exposed on the public API for unit testing purposes.

Usage

sumDnorm(x, mean, sd)

Arguments

x Vector of i.i.d. Gaussian random variables
mean Vector of means
sd Vector of standard deviations

Details

The sum of the log-likelihoods (log of the product of the likelihoods) for independent, identically-distributed, Gaussian random variables. Note: this Rcpp function is thread-safe, unlike the equivalent alternatives.

Value

log-likelihood of x

See Also

sum(dnorm(x,mean,sd,log=TRUE))

---
weightedGaussian

Compute the spectral signature using Gaussian peaks.

Description

Calculates the value of the squared exponential radial basis function at the given wavelengths, given
the parameters of the peaks. This function is thread-safe.

Usage

weightedGaussian(location, scale, amplitude, wavelengths)

Arguments

location Vector of location parameters of the peaks (mean).

scale Vector of scale parameters of the peaks (standard deviation).

amplitude Vector of amplitudes of the peaks.

wavelengths Vector of wavenumbers at which to compute the function.

Value

The value of the Gaussian function at the given wavelengths.

Examples

Cal_V <- seq(300,400,by=5)
loc <- c(320,350,375)
sca <- c(10,5,18)
amp <- c(1000,5000,2000)
weightedGaussian(loc,sca,amp,Cal_V)
### weightedLorentzian

*Compute the spectral signature using Lorentzian peaks.*

**Description**

Calculates the value of the Lorentzian function at the given wavelengths, given the parameters of the peaks. This function is thread-safe.

**Usage**

```
weightedLorentzian(location, scale, amplitude, wavelengths)
```

**Arguments**

- `location` Vector of location parameters of the peaks.
- `scale` Vector of scale parameters of the peaks.
- `amplitude` Vector of amplitudes of the peaks.
- `wavelengths` Vector of wavenumbers at which to compute the function.

**Value**

The value of the Lorentzian function at the given wavelengths.

**Examples**

```
Cal_V <- seq(300,400,by=5)
loc <- c(320,350,375)
sca <- c(10,5,18)
amp <- c(1000,5000,2000)
weightedLorentzian(loc,sca,amp,Cal_V)
```

### weightedMean

*Compute the weighted arithmetic means of the particles.*

**Description**

This SMC estimate of the means can be used to centre independent Metropolis-Hastings proposals.

**Usage**

```
weightedMean(particles, log_weights)
```

**Arguments**

- `particles` npkts * npart Matrix of parameter values for each particle.
- `log_weights` logarithms of the importance weights of each particle.
Weighted Variance

Value
A vector of means, one for each row.

See Also
weighted.mean

Weighted Variance
Compute the weighted variance of the particles.

Description
This SMC estimate of the variance can be used to scale the bandwidth of adaptive, Gaussian random walk Metropolis-Hastings proposals.

Usage
weightedVariance(particles, log_weights, mean)

Arguments
- particles: nppeaks x npart Matrix of parameter values for each particle.
- log_weights: logarithms of the importance weights of each particle.
- mean: Vector of weighted means of each particle.

Value
A vector of variances, one for each row.

See Also
wtd.var
**Index**

<table>
<thead>
<tr>
<th>datasets</th>
<th>weightedMean, 25</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsTamra, 12</td>
<td>weightedVariance, 26</td>
</tr>
<tr>
<td>methanol, 13</td>
<td>wtd.var, 26</td>
</tr>
<tr>
<td>result, 17</td>
<td></td>
</tr>
<tr>
<td>result2, 18</td>
<td></td>
</tr>
</tbody>
</table>

| computeLogLikelihood, 3  |                   |
| copyLogProposals, 4      |                   |
| effectiveSampleSize, 4   |                   |
| fitSpectraMCMC, 5        |                   |
| fitSpectraSMC, 6         |                   |
| fitVoigtIBIS, 7          |                   |
| fitVoigtPeaksSMC, 9      |                   |
| getBsplineBasis, 10      |                   |
| getVoigtParam, 11        |                   |
| lsTamra, 12              |                   |
| marginalMetropolisUpdate, 5, 12|
| methanol, 13             |                   |
| mhUpdateVoigt, 14        |                   |
| mixedVoigt, 15           |                   |

| resampleParticles, 16    |                   |
| residualResampling, 16, 17|
| result, 17               |                   |
| result2, 18              |                   |
| reWeightParticles, 19    |                   |
| serrsBayes, 20           |                   |
| sparseMatrix, 11         |                   |
| sumDexp, 22              |                   |
| sumDlogNorm, 22          |                   |
| sumDnorm, 23             |                   |
| weighted.mean, 26        |                   |
| weightedGaussian, 24     |                   |
| weightedLorentzian, 25   |                   |