Package ‘multiband’

February 20, 2015

Title Period Estimation for Multiple Bands

Description Algorithms for performing joint parameter estimation in astronomical survey data acquired in multiple bands.

Version 0.1.0

Maintainer Eric C. Chi <ecchi1105@gmail.com>

Author Eric C. Chi, James P. Long

Depends R (>= 3.0.2)

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-18 07:12:06

R topics documented:

bcd_all *Run BCD on all frequencies*

Description

`bcd_all` runs bcd_inexact on all input frequencies.

Usage

```
bcd_all(tms, sol_ls, omega_seq, gamma1, gamma2, at, max_iter = 100,
  verbose = FALSE)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,m,sigma) for each band.
`sol_ls`	list of lists where sol_ls is output from lomb_scargle
`omega_seq`	vector of freqencies used for producing sol
`gamma1`	nonnegative regularization parameter for shrinking amplitudes
`gamma2`	nonnegative regularization parameter for shrinking phases
`at`	prior vector
`max_iter`	maximum number of outer iterations for bcd
`verbose`	boolean flag; if TRUE diagnostic info is printed

bcd_express *Run BCD on a subset of frequencies*

Description

`bcd_express` runs bcd_inexact on a subset of frequencies determined by Lomb-Scargle fits that is guaranteed to include the minimum.

Usage

```
bcd_express(tms, sol_ls, omega_seq, gamma1, gamma2, at, max_iter = 100,
  verbose = FALSE)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,m,sigma) for each band.
`sol_ls`	list of lists where sol_ls is output from lomb_scargle
`omega_seq`	vector of freqencies used for producing sol
`gamma1`	nonnegative regularization parameter for shrinking amplitudes
`gamma2`	nonnegative regularization parameter for shrinking phases
`at`	prior vector
`max_iter`	maximum number of outer iterations for bcd
`verbose`	boolean flag; if TRUE diagnostic info is printed

`bcd_inexact` *Inexact Block coordinate descent*

Description

`bcd_inexact` performs inexact block coordinate descent on the penalized negative log likelihood of the multiband problem.

Usage

```
bcd_inexact(tms, beta, a, at, rho, omega, gamma1 = 0, gamma2 = 0,
  max_iter = 100, tol = 1e-04, mm_iter = 5)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,m,sigma) for each band
`beta`	initial intercept estimates
`a`	initial amplitude estimates
`at`	prior vector
`rho`	initial phase estimates
`omega`	frequency
`gamma1`	nonnegative regularization parameter for shrinking amplitudes
`gamma2`	nonnegative regularization parameter for shrinking phases
`max_iter`	maximum number of outer iterations
`tol`	tolerance on relative change in loss
`mm_iter`	number of MM iterations for rho update

Examples

```
test_data <- synthetic_multiband()
B <- test_data$B
tms <- test_data$tms
beta <- test_data$beta
a <- test_data$a
rho <- test_data$rho
omega <- test_data$omega
at <- rnorm(B)
at <- as.matrix(at/sqrt(sum(at**2)),ncol=1)
at <- rep(1/sqrt(B),B)

## Verify monotonicity of block coordinate descent
gamma1 <- 100
gamma2 <- 10
max_iter <- 1e2
loss <- double(max_iter)
beta_next <- beta
a_next <- a
rho_next <- rho
for (iter in 1:max_iter) {
  sol_bcd <- bcd_inexact(tms,beta_next,a_next,at,rho_next,omega,gamma1=gamma1,gamma2=gamma2,
    max_iter=1)
  beta_next <- sol_bcd$beta0
  a_next <- sol_bcd$A
  rho_next <- sol_bcd$rho
  loss[iter] <- pnll(tms,beta_next,a_next,at,rho_next,omega,gamma1,gamma2)
}
loss <- c(pnll(tms,beta,a,at,rho,omega,gamma1,gamma2),loss)
plot(1:(max_iter+1),loss,xlab='iteration',ylab='objective',pch=16)

## Check to see if the fixed points of the BCD algorithm stops at
## a stationary point of the original problem
gradient_check(tms,beta_next,a_next,at,rho_next,omega,gamma1,gamma2)

## Example Pipeline
## 1. Use Lomb Scargle to fit initial estimate using all bands treated as one band.
t <- c(); m <- c(); sigma <- c()
for (b in 1:B) {
  t <- c(t,tms[[b]][,1])
  m <- c(m,tms[[b]][,2])
  sigma <- c(sigma,tms[[b]][,3])
}
sol_ls <- lomb_scargle(t,m,sigma,omega)

beta0_ls <- rep(sol_ls$beta0,B)
A_ls <- rep(sol_ls$A,B)
rho_ls <- rep(sol_ls$rho,B)
sol_bcd <- bcd_inexact(tms,beta0_ls,A_ls,at,rho_ls,omega,gamma1=gamma1,gamma2=gamma2,max_iter=5)

sol_bcd_rand <- bcd_inexact(tms,rep(-1,B),rep(0.1,B),at,rep(1,B),omega,gamma1=gamma1,gamma2=gamma2,
  max_iter=5)
```

```
## Try several omega
nOmega <- 10
omega_seq <- seq(0.1,0.3,length=nOmega)
sol_ls <- vector(mode="list",length=nOmega)
sol_bcd <- vector(mode="list",length=nOmega)
RSS_seq <- double(nOmega)
RSS_ls_seq <- double(nOmega)
for (i in 1:nOmega) {
  sol_ls[[i]] <- lomb_scargle(t,m,sigma,omega_seq[i])
  RSS_ls_seq[i] <- sol_ls[[i]]$RSS
  beta0_ls <- rep(sol_ls[[i]]$beta0,B)
  A_ls <- rep(sol_ls[[i]]$A,B)
  rho_ls <- rep(sol_ls[[i]]$rho,B)
 sol_bcd[[i]] <- bcd_inexact(tms,beta0_ls,A_ls,at,rho_ls,omega_seq[i],gamma1=gamma1,gamma2=gamma2,
   max_iter=10,tol=1e-10)
 RSS_seq[i] <- pnll(tms,sol_bcd[[i]]$beta0,sol_bcd[[i]]$A,at,sol_bcd[[i]]$rho,omega_seq[i],0,0)
  print(paste0("Completed ", i))
}
plot(omega_seq,RSS_seq,xlab=expression(omega),ylab='RSS',pch=16)
ix_min <- which(RSS_seq==min(RSS_seq))
sol_bcd_final <- sol_bcd[[ix_min]]
```

---

cepii *Light curve data from two bands*

Description

A dataset containing band I and V data from OGLE-LMC-T2CEP-009. There are two data frames, iband and vband, each of which has three columns. (time,magnitude,magnitude error).

Format

Two data frames with 3 variables, and 554 (iband) and 65 (vband) rows.

Examples

```
## Load I and V bands
iband <- cepii[[1]]
vband <- cepii[[2]]

## Try finer grid sizes as well, e.g. 'nOmega <- 500'
nOmega <- 10
omega_seq <- seq(3.5,3.65,length.out=nOmega)
sol_ls <- vector(mode="list",length=nOmega)
RSS_ls <- double(nOmega)

## Drastically subsample the data and see if the methods can find the period.
subi <- seq(1,nrow(iband),by=20)
```

```
## 1. Lomb Scargle on I band and V band separately
for (i in 1:nOmega) {
  sol_ls[[i]] <- lomb_scargle(iband[subi,1],iband[subi,2],iband[subi,3],omega_seq[i])
  RSS_ls[i] <- sol_ls[[i]]$RSS
}
plot(omega_seq,RSS_ls,xlab=expression(omega),ylab='RSS',main='I band',pch=16)

subv <- seq(1,nrow(vband),by=4)
for (i in 1:nOmega) {
  sol_ls[[i]] <- lomb_scargle(vband[subv,1],vband[subv,2],vband[subv,3],omega_seq[i])
  RSS_ls[i] <- sol_ls[[i]]$RSS
}
plot(omega_seq,RSS_ls,xlab=expression(omega),ylab='RSS',main='V band',pch=16)

## 2. Naive pooled Lomb Scargle versus Fused
tms <- vector(mode="list",length=2)
tms[[1]] <- iband[subi,,drop=FALSE]
tms[[2]] <- vband[subv,,drop=FALSE]
B <- length(tms)
t <- c(); m <- c(); sigma <- c()
for (b in 1:B) {
  t <- c(t,tms[[b]][,1])
  m <- c(m,tms[[b]][,2])
  sigma <- c(sigma,tms[[b]][,3])
}

sol_ls <- vector(mode="list",length=nOmega)
sol_bcd <- vector(mode="list",length=nOmega)
RSS_seq <- double(nOmega)
RSS_ls_seq <- double(nOmega)
gamma1 <- 1000
gamma2 <- 10
for (i in 1:nOmega) {
  sol_ls[[i]] <- lomb_scargle(t,m,sigma,omega_seq[i])
  RSS_ls_seq[i] <- sol_ls[[i]]$RSS
  beta0_ls <- rep(sol_ls[[i]]$beta0,B)
  A_ls <- rep(sol_ls[[i]]$A,B)
  rho_ls <- rep(sol_ls[[i]]$rho,B)
 sol_bcd[[i]] <- bcd_inexact(tms,beta0_ls,A_ls,rep(1,B),rho_ls,omega_seq[i],gamma1,gamma2,
   max_iter=1e4,tol=1e-10)
  RSS_seq[i] <- pnll(tms,sol_bcd[[i]]$beta0,sol_bcd[[i]]$A,rep(1,B),
    sol_bcd[[i]]$rho,omega_seq[i],0,0)
  print(paste0("Completed ", i))
}

plot(omega_seq,RSS_seq,xlab=expression(omega),ylab='RSS',main='I & V band fusion',pch=16)
plot(omega_seq,RSS_ls_seq,xlab=expression(omega),ylab='RSS',main='naive Lomb-Scargle',pch=16)
ix_min <- which(RSS_seq==min(RSS_seq))
sol_bcd_final <- sol_bcd[[ix_min]]
```

get_freqs *Construct grid of frequencies*

Description

get_freqs constructs a grid of frequencies from a specified minimum and maximum period and frequency grid length.

Usage

```
get_freqs(period_min = 1, period_max = 100, freq_del = 0.1/4000)
```

Arguments

period_min	minimum period
period_max	maximum period
freq_del	gride size in frequency

gradient_check *Check Gradients*

Description

gradient_check computes the block coordinate gradients.

Usage

```
gradient_check(tms, beta, A, at, rho, omega, gamma1, gamma2)
```

Arguments

tms	list of matrices whose rows are the triple (t,m,sigma) for each band
beta	initial intercept estimates
A	initial amplitude estimates
at	prior vector
rho	Initial phase estimates
omega	frequency
gamma1	nonnegative regularization parameter for shrinking amplitudes
gamma2	nonnegative regularization parameter for shrinking phases

lomb_scargle — *Lomb Scargle method*

Description

`lomb_scargle` applies the Lomb Scargle method for performing weighted nonlinear least squares in a single band using magnitude measurements recorded at irregular intervals.

Usage

```
lomb_scargle(t, m, sigma, omega, weights_flag = TRUE)
```

Arguments

`t`	times
`m`	magnitudes
`sigma`	errors
`omega`	frequency
`weights_flag`	boolean (TRUE to use inverse errors as weights; FALSE uses uniform weights)

Examples

```
n <- 1e3
set.seed(12345)
sigma <- runif(n)
t <- cumsum(runif(n))
A <- 2.3
rho <- 0.1
omega <- 0.2
beta0 <- 0.3
m <- beta0 + A*sin(omega*t + rho) + sigma*rnorm(n)
plot(t,m,xlab='time',ylab='magnitude',main='Simulated Light Curve',pch=16)

## Try several omega
nOmega <- 500
omega_seq <- seq(0.1,0.3,length=nOmega)
sol_ls <- vector(mode="list",length=nOmega)
RSS_seq <- double(nOmega)
for (i in 1:nOmega) {
  sol_ls[[i]] <- lomb_scargle(t,m,sigma,omega_seq[i])
  RSS_seq[i] <- sol_ls[[i]]$RSS
}

plot(omega_seq,RSS_seq,xlab=expression(omega),ylab='RSS',pch=16)
ix_min <- which(RSS_seq==min(RSS_seq))
sol_final <- sol_ls[[ix_min]]
```

mm_phase_obj — *Majorization for phase*

Description

mm_phase_obj computes the convex quadratic majorization used in the phase estimation step.

Usage

```
mm_phase_obj(rho, tms, beta, a, atilde, rho_anchor, omega, gamma1, gamma2)
```

Arguments

rho	vector of new phase values.
tms	list of matrices whose rows are the triple (t,m,sigma) for each band.
beta	vector of intercepts
a	amplitude estimates
atilde	prior vector
rho_anchor	vector of the current estimates of the phase
omega	frequency
gamma1	nonnegative regularization parameter for shrinking amplitudes
gamma2	nonnegative regularization parameter for shrinking phases

multiband — *multiband*

Description

multiband

pgls *Penalized Generalized Lomb-Scargle*

Description

pgls estimates periods for a collection of lightcurves sampled over multiple bands. It borrows strength across multiple bands via shrinkage penalties on amplitudes an phases.

Usage

```
pgls(lclist, period_min = NULL, period_max = NULL, periods = NULL,
  gamma1 = 0, gamma2 = 20, at = rep(1, length(lclist)), LS_flag = TRUE,
  sol_ls = NULL, BCD_flag = TRUE, fast_BCD_flag = TRUE, max_iter = 100,
  tol = 1e-04, mm_iter = 5, verbose = FALSE)
```

Arguments

lclist	list of lightcurve data frames
period_min	minimum period
period_max	maximum period
periods	grid of periods
gamma1	vector of Amplitude regularization parameter
gamma2	vector of Phase regularization parameter
at	amplitude prior parameter
LS_flag	boolean whether to run Lomb-Scargle algorithm
sol_ls	Lomb-Scargle solution, used if LS_flag=FALSE
BCD_flag	boolean whether to run bcd algorithm
fast_BCD_flag	boolean whether to run BCD on relevant subset of periods
max_iter	maximum number of outer iterations - passed to bcd_inexact
tol	tolerance on relative change in loss - passed to bcd_inexact
mm_iter	number of MM iterations for rho update - passed to bcd_inexact
verbose	boolean whether to print progress

Examples

```
period_min <- 3.5
period_max <- 3.65
out <- pgls(cepii,period_min=period_min,period_max=period_max)
out$best_fitBCD
```

pnll *Penalized negative log likelihood*

Description

`pnll` computes the penalized negative log likelihood

Usage

```
pnll(tms, beta, a, atilde, rho, omega, gamma1, gamma2)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,m,sigma) for each band.
`beta`	vector of intercepts
`a`	vector of amplitudes
`atilde`	prior vector
`rho`	vector of phases.
`omega`	frequency
`gamma1`	nonnegative regularization parameter for shrinking amplitudes
`gamma2`	nonnegative regularization parameter for shrinking phases

single_band_lomb_scargle *Single band Lomb-Scargle*

Description

`single_band_lomb_scargle` runs Lomb-Scargle on all bands in tms at all frequencies in omega_seq

Usage

```
single_band_lomb_scargle(tms, omega_seq)
```

Arguments

`tms`	List of lightcurves
`omega_seq`	Sequence of frequencies

synthetic_multiband *Generate Synthetic Multiband Data*

Description

`synethetic_multiband` generates multiband data of a given fequency.

Usage

```
synthetic_multiband(B = 5, omega = 0.2, beta = 0, beta_sd = 0.1,
  a = 2, a_sd = 0.25, rho = 0, rho_sd = 0.1, n_max = 1000,
  n_min = 100, seed = 12345)
```

Arguments

B	number of bands
omega	frequency common to all bands
beta	mean intercept
beta_sd	standard deviation around mean intercept
a	mean amplitude
a_sd	standard deviation around mean amplitude
rho	mean phase
rho_sd	standard deviation around mean phase
n_max	maximum number of time points per band
n_min	minimum number of time points per band
seed	initialization parameter for random number generator

Examples

```
tms <- synthetic_multiband()
```

update_amplitude *Update Amplitude parameter*

Description

`update_amplitude` solves a simple linear system of equations (rank 1 perturbation on a diagonal matrix) to update the amplitude estimates.

Usage

```
update_amplitude(tms, beta, rho, omega, at, gamma)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,mu,sigma) for each band
`beta`	vector of the current intercept estimates
`rho`	vector of the current phase estimates
`omega`	frequency
`at`	prior vector
`gamma`	nonnegative regularization parameter

Examples

```
test_data <- synthetic_multiband()
B <- test_data$B
tms <- test_data$tms
beta <- test_data$beta
rho <- test_data$rho
omega <- test_data$omega
gamma <- 1
at <- rnorm(B)
at <- as.matrix(at/sqrt(sum(at**2)),ncol=1)

## Check answer
a_next <- update_amplitude(tms,beta,rho,omega,at,gamma)

e <- double(B)
xi <- double(B)
for (b in 1:B) {
  nb <- length(tms[[b]][,1])
  s <- sin(omega*tms[[b]][,1] + rho[b])
  w <- 1/(tms[[b]][,3]**2)
  e[b] <- t(s)%*%(w*s) + gamma
  xi[b] <- t(s)%*%(w*(tms[[b]][,2]-beta[b]))
}
a_direct <- solve(diag(e)-gamma*at%*%t(at),xi)
norm(as.matrix(a_direct-a_next),'f')
```

`update_beta` *Update Intercept beta*

Description

`update_beta` updates the vector of band specific intercepts.

Usage

```
update_beta(tms, a, rho, omega)
```

Arguments

- `tms` list of matrices whose rows are the triple (t,m,sigma) for each band.
- `a` Amplitude estimates
- `rho` vector of the current estimates of the phase
- `omega` frequency

Examples

```
test_data <- synthetic_multiband()
tms <- test_data$tms
a <- test_data$a
rho <- test_data$rho
omega <- test_data$omega

update_beta(tms,a,rho,omega)
```

---

update_beta_gamma *Update Beta0 parameter*

Description

`update_amplitude` solves a simple linear system of equations (rank 1 perturbation on a diagonal matrix) to update the beta0 estimates.

Usage

```
update_beta_gamma(tms, a, rho, omega, gamma)
```

Arguments

- `tms` list of matrices whose rows are the triple (t,mu,sigma) for each band
- `a` vector of the current amplitude estimates
- `rho` vector of the current phase estimates
- `omega` frequency
- `gamma` nonnegative regularization parameter

Examples

```
test_data <- synthetic_multiband()
B <- test_data$B
tms <- test_data$tms
beta <- test_data$beta
rho <- test_data$rho
omega <- test_data$omega
gamma <- 1
at <- rnorm(B)
```

```
at <- as.matrix(at/sqrt(sum(at**2)),ncol=1)

## Check answer
a_next <- update_amplitude(tms,beta,rho,omega,at,gamma)

e <- double(B)
xi <- double(B)
for (b in 1:B) {
  nb <- length(tms[[b]][,1])
  s <- sin(omega*tms[[b]][,1] + rho[b])
  w <- 1/(tms[[b]][,3]**2)
  e[b] <- t(s)%*%(w*s) + gamma
  xi[b] <- t(s)%*%(w*(tms[[b]][,2]-beta[b]))
}
a_direct <- solve(diag(e)-gamma*at%*%t(at),xi)
norm(as.matrix(a_direct-a_next),'f')
```

`update_Lipschitz` *Update Lipschitz constant for phase majorization*

Description

`update_Lipschitz` computes a Lipschitz constant for the gradient of the objective function with the amplitude parameters fixed.

Usage

```
update_Lipschitz(tms, beta, a, inflate = 1)
```

Arguments

`tms`	list of matrices whose rows are the triple (t,m,sigma) for each band.
`beta`	intercept estimates
`a`	amplitude estimates
`inflate`	factor by which to multiply the Lipschitz constants

Examples

```
test_data <- synthetic_multiband()
tms <- test_data$tms
beta <- test_data$beta
a <- test_data$a
update_Lipschitz(tms,beta,a)
```

update_rho_inexact *Update Phase parameter*

Description

update_rho_inexact inexactly updates the phase parameter rho via an MM algorithm using a convex quadratic majorization.

Usage

```
update_rho_inexact(tms, beta, a, rho, omega, gamma, max_iter = 5)
```

Arguments

tms	list of matrices whose rows are the triple (t,mu,sigma) for each band.
beta	vector of the current intercept estimates
a	amplitude estimates
rho	vector of the current estimates of the phase
omega	frequency
gamma	nonnegative regularization parameter
max_iter	maximum number of iterations

Examples

```
test_data <- synthetic_multiband()
tms <- test_data$tms
B <- test_data$B
beta <- test_data$beta
a <- test_data$a
rho <- test_data$rho
omega <- test_data$omega
gamma <- 1

## Check answer
rho_next <- update_rho_inexact(tms,beta,a,rho,omega,gamma,max_iter=1)

L <- update_Lipschitz(tms,beta,a)
f <- L + gamma
zeta <- update_zeta(tms,beta,a,rho,L,omega)
rho_direct <- solve(diag(f)-(gamma/B),zeta)
norm(as.matrix(rho_direct-rho_next),'f')

## Verify monotonicity of MM algorithm
max_iter <- 1e2
obj <- double(max_iter)
loss <- double(max_iter)
rho_last <- rho
```

```
at <- rep(1/sqrt(B),B)
for (iter in 1:max_iter) {
  rho_next <- update_rho_inexact(tms,beta,a,rho_last,omega,gamma,max_iter=1)
  obj[iter] <- mm_phase_obj(rho_next,tms,beta,a,at,rho_last,omega,gamma,gamma)
  loss[iter] <- pnll(tms,beta,a,at,rho_next,omega,gamma,gamma)
  rho_last <- rho_next
}
obj <- c(mm_phase_obj(rho,tms,beta,a,at,rho,omega,gamma,gamma),obj)
plot(1:(max_iter+1),obj,xlab='iteration',ylab='mm objective',pch=16)
loss <- c(pnll(tms,beta,a,at,rho,omega,gamma,gamma),loss)
plot(1:(max_iter+1),loss,xlab='iteration',ylab='loss',pch=16)
```

---

update_zeta — *Update working response in rho update*

Description

update_zeta computes the working response for the MM algorithm implemented in update_rho.

Usage

```
update_zeta(tms, beta, a, rho, L, omega)
```

Arguments

tms	list of matrices whose rows are the triple (t,m,sigma) for each band.
beta	intercept estimates
a	amplitude estimates
rho	vector of the current estimates of the phase
L	vector of Lipschitz constants
omega	frequency

Index