Package 'SpaCCr'

November 23, 2016

Type Package

Title Spatial Convex Clustering

Version 0.1.0

Author John Nagorski

Maintainer John Nagorski<jn13@rice.edu>

Description Genomic Region Detection via Spatial Convex Clustering. See <https://arxiv.org/abs/1611.04696> for details.

Depends R (>= 2.10)

License GPL-3

LazyData TRUE

LinkingTo Rcpp, RcppArmadillo

Imports abind, dplyr, ggplot2, parallel, Rcpp, tidyr

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-11-23 09:38:35

R topics documented:

`CNVPlotSeriesMeans` *Plot subjects' copy number data with cluster means overlayed for a single chromosome*

Description

Plot subjects' copy number data with cluster means overlayed for a single chromosome

Usage

```
CNVPlotSeriesMeans(Location, X, Cluster, NSubj = 3, lowery = -1,
  uppery = 1)
```

Arguments

`Location`	A vector of length p with chromosomal locations
`X`	A variable (p) by subject (n) data matrix
`Cluster`	A vector of length p with cluster labels
`NSubj`	A positve integer; number of randomly selected subjects to plot.
`lowery, uppery`	Scalars; limits for y-axis

Examples

```
NULL
```

`GetClusters` *Compute Clusters from fusions*

Description

Compute Clusters from fusions

Usage

```
GetClusters(V)
```

Arguments

`V`	An n by p-1 data matrix

Examples

```
NULL
```

`GetGammaCV` *Get optimal cross validated gamma values by various rules*

Description

Get optimal cross validated gamma values by various rules

Usage

```
GetGammaCV(ErrMat, rule = 1, gamma.seq)
```

Arguments

`ErrMat`	Matrix of cross validated errors outputted by `GetCVErrMat`
`rule`	A number indicating how optimal gamma should be chosen. 1 for minimum cv error, 2 for 1 standard error rule
`gamma.seq`	sequence of regularization parameters used for cross validation.

Value

A scalar. Optimal gamma selected by CV rule.

Examples

```
NULL
```

`GetParBlocks` *Function to compute blocks for parallization; should not be called directly*

Description

Function to compute blocks for parallization; should not be called directly

Usage

```
GetParBlocks(X, w)
```

Arguments

`X`	An n by p data matrix
`w`	A vector of positive scalars of length p-1

Examples

```
NULL
```

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`methy` *A subset of methylation data for 50 subjects*

Description

A subset of methylation data for 50 subjects

Usage

```
methy
```

Format

A data frame with 1000 rows and 53 columns. Rows are variables and columns are:

ProbeID Varibale ID string

Chromosome Chromosome Number

Genomic_Coordinate Probe location on chromosome in basepairs

TCGA-A7-A0CE-11A-21D-A10Q-05 Subject ID

TCGA-A7-A0CH-11A-32D-A10Q-05 Subject ID

TCGA-A7-A0DB-11A-33D-A093-05 Subject ID

TCGA-A7-A0DC-11A-41D-A10Q-05 Subject ID

TCGA-BH-A0AY-11A-23D-A10Q-05 Subject ID

TCGA-BH-A0BV-11A-31D-A10Q-05 Subject ID

TCGA-BH-A0DZ-11A-22D-A10Q-05 Subject ID

TCGA-A2-A1FV-01A-11D-A13K-05 Subject ID
TCGA-A2-A1FW-01A-11D-A13K-05 Subject ID
TCGA-A2-A1FX-01A-11D-A13K-05 Subject ID
TCGA-A2-A1G0-01A-11D-A13K-05 Subject ID
TCGA-A2-A1G1-01A-21D-A13K-05 Subject ID
TCGA-A2-A1G4-01A-11D-A13K-05 Subject ID
TCGA-A2-A1G6-01A-11D-A13K-05 Subject ID
TCGA-A7-A13G-01A-11D-A13K-05 Subject ID
TCGA-A7-A13G-01B-04D-A22R-05 Subject ID
TCGA-A7-A13G-11A-51D-A13T-05 Subject ID
TCGA-AO-A1KO-01A-31D-A13K-05 Subject ID
TCGA-AO-A1KP-01A-11D-A13K-05 Subject ID
TCGA-AO-A1KQ-01A-11D-A13K-05 Subject ID
TCGA-AO-A1KS-01A-11D-A13K-05 Subject ID
TCGA-AO-A1KT-01A-11D-A13K-05 Subject ID
TCGA-AQ-A1H2-01A-11D-A13K-05 Subject ID
TCGA-AQ-A1H3-01A-31D-A13K-05 Subject ID
TCGA-B6-A1KC-01A-11D-A13K-05 Subject ID
TCGA-B6-A1KC-01B-11D-A161-05 Subject ID
TCGA-B6-A1KF-01A-11D-A13K-05 Subject ID
TCGA-B6-A1KN-01A-11D-A13K-05 Subject ID
TCGA-BH-A1EN-01A-11D-A13K-05 Subject ID
TCGA-BH-A1EN-11A-23D-A13T-05 Subject ID
TCGA-BH-A1EX-01A-11D-A13K-05 Subject ID
TCGA-BH-A1EY-01A-11D-A13K-05 Subject ID
TCGA-BH-A1EY-11B-21D-A13T-05 Subject ID
TCGA-BH-A1F2-01A-31D-A13K-05 Subject ID
TCGA-BH-A1F2-11A-32D-A13T-05 Subject ID
TCGA-BH-A1F5-01A-12D-A13K-05 Subject ID
TCGA-BH-A1F5-11A-43D-A13T-05 Subject ID
TCGA-BH-A1F6-01A-11D-A13K-05 Subject ID
TCGA-BH-A1F6-11B-94D-A13T-05 Subject ID
TCGA-BH-A1F8-01A-11D-A13K-05 Subject ID
TCGA-BH-A1F8-11B-21D-A13T-05 Subject ID
TCGA-BH-A1FB-01A-11D-A13K-05 Subject ID
TCGA-BH-A1FB-11A-33D-A13T-05 Subject ID
TCGA-BH-A1FC-01A-11D-A13K-05 Subject ID

TCGA-BH-A1FC-11A-32D-A13T-05 Subject ID

TCGA-BH-A1FD-01A-11D-A13K-05 Subject ID

TCGA-BH-A1FD-11B-21D-A13T-05 Subject ID

TCGA-BH-A1FE-01A-11D-A13K-05 Subject ID

TCGA-BH-A1FE-06A-11D-A212-05 Subject ID

TCGA-BH-A1FE-11B-14D-A13T-05 Subject ID

Source

http://cancergenome.nih.gov/

`MethyRegionPlot` *Plots methylation data by Genomic Coordinates for a given chromosomal region with cluster means overlayed for each subject.*

Description

Plots methylation data by Genomic Coordinates for a given chromosomal region with cluster means overlayed for each subject.

Usage

```
MethyRegionPlot(X, Coord, Cluster, SubjInd = 1:3, Start, End)
```

Arguments

`X`	A Subject by Probe data matrix for a single chromosome of CNV data
`Coord`	A vector of Genomic Coordinates for a single chromosome
`Cluster`	Cluster labels for each probe
`SubjInd`	A vector of numeric indicies corresponding to the Subjects to be plotted.
`Start`	Genomic Coordinate minimum
`End`	Genomic Coordinate maximum

Examples

```
library(dplyr)
library(tidyr)
data("methy")
methy <- methy[1:20,1:10]
Coordinates <- methy$Genomic_Coordinate
methy %>%
 tbl_df() %>%
 select(-Chromosome,-Genomic_Coordinate) %>%
 gather(Subject,Value,-ProbeID) %>%
 spread(ProbeID,Value) -> X
```

```
SubjectLabels <- X$Subject
X <- X[,-1] %>% as.matrix()
nsubj <- nrow(X)
nprobes <- ncol(X)
nweights <- choose(nprobes,2)
diff.vals <- diff(Coordinates)
too.far <- diff.vals > 20000
sig = 1/5e3
w.values <- exp(-sig*diff.vals)
w.values[too.far] = 0

verbose=TRUE
tol.base = 1e-4
tol.miss = 1e-4
max.iter.base=5000
max.iter.miss=500
ngam = 20
gamma.seq <- exp(seq(log(1e-1),log(1e1),length.out=ngam))
CVRes <- SpaCC_CV(X=t(scale(t(X),center=TRUE,scale=FALSE)),
                  w=w.values,
                  gamma.seq=gamma.seq,
                  nfolds=5,
                  nu=1/nsubj,
                  verbose=TRUE,
                  tol.base=tol.base,
                  tol.miss=tol.miss,
                  max.iter.base=max.iter.base,
                  max.iter.miss=max.iter.miss,
                  parallel=FALSE,frac = .1)
PlotCV(CVRes$ErrMat,gamma.seq = CVRes$gamma.seq,rule = 1)
best.gam <- GetGammaCV(CVRes$ErrMat,rule = 1,gamma.seq = CVRes$gamma.seq)
bo <-t(scale(t(X),center=TRUE,scale=FALSE))
bo[is.na(bo)] <- mean(bo,na.rm=TRUE)
Sol <- SpaCC_Missing(t(scale(t(X),center=TRUE,scale=FALSE)),
                         w.values,
                         gamma = best.gam,
                         nu=1/nsubj,
                         verbose=TRUE,
                         tol.base=tol.base,
                         tol.miss=tol.miss,
                         max.iter.base=max.iter.base,
                         max.iter.miss=max.iter.miss,
                         bo,
                         t(diff(t(bo))),
                         t(diff(t(bo))))
VThreshed <- Sol$V
clustsThreshed <- GetClusters(VThreshed)
NEstRegion <- length(unique(clustsThreshed$cluster))
NEstRegion
VThreshed <- ThreshV(Sol$V,X,mult = 1)
clustsThreshed <- GetClusters(VThreshed)
NEstRegion <- length(unique(clustsThreshed$cluster))
NEstRegion
```

```
start.coord <- 2e5
end.coord <- 4e5
MethyRegionPlot(X,Coordinates,clustsThreshed$cluster,SubjInd = 1:3,Start=start.coord,End=end.coord)
```

PlotCV *A function for plotting cross validation errors*

Description

A function for plotting cross validation errors

Usage

```
PlotCV(ErrMat, rule = 2, gamma.seq)
```

Arguments

ErrMat	A matrix of error outputted by SpaCC_CV
rule	An interger indicating which CV rule to choose
gamma.seq	The sequence of regularization parameters

Examples

```
library(dplyr)
library(tidyr)
data("methy")
methy <- methy[1:20,1:10]
Coordinates <- methy$Genomic_Coordinate
methy %>%
 tbl_df() %>%
 select(-Chromosome,-Genomic_Coordinate) %>%
 gather(Subject,Value,-ProbeID) %>%
 spread(ProbeID,Value) -> X
SubjectLabels <- X$Subject
X <- X[,-1] %>% as.matrix()
nsubj <- nrow(X)
nprobes <- ncol(X)
nweights <- choose(nprobes,2)
diff.vals <- diff(Coordinates)
too.far <- diff.vals > 20000
sig = 1/5e3
w.values <- exp(-sig*diff.vals)
w.values[too.far] = 0

verbose=TRUE
tol.base = 1e-4
tol.miss = 1e-4
max.iter.base=5000
max.iter.miss=500
```

```
ngam = 20
gamma.seq <- exp(seq(log(1e-1),log(1e1),length.out=ngam))
CVRes <- SpaCC_CV(X=t(scale(t(X),center=TRUE,scale=FALSE)),
                  w=w.values,
                  gamma.seq=gamma.seq,
                  nfolds=5,
                  nu=1/nsubj,
                  verbose=TRUE,
                  tol.base=tol.base,
                  tol.miss=tol.miss,
                  max.iter.base=max.iter.base,
                  max.iter.miss=max.iter.miss,
                  parallel=FALSE,frac = 1)
PlotCV(CVRes$ErrMat,gamma.seq = CVRes$gamma.seq,rule = 1)
```

SpaCC *Base function for computing SpaCC solution for single regularization value.*

Description

Base function for computing SpaCC solution for single regularization value.

Usage

```
SpaCC(X, w, gamma, nu, verbose, tol, maxiter, Uinit, Vinit, Laminit)
```

Arguments

X	A subject (n) by probe (p) data matrix
w	A vector weights for adjacent probes. Should have length nprobes -1
gamma	A scalar value for the regularization parameter
nu	A scalar value for the step size in AMA algorithm
verbose	Logical value whether progress should be printed
tol	A scalar value for convergence tolerance.
maxiter	Maximum number of iterations
Uinit	A matrix used for warm starts with U
Vinit	A matrix used for warm start with V
Laminit	A matrix used for warm starts with Lam

Value

An RcppArmadillo field object. Has three components, each holds the U,V, and Lam matrix for the current regularization

SpaCC_CV *Perform Cross Validation to select gamma/sparsity level*

Description

Perform Cross Validation to select gamma/sparsity level

Usage

```
SpaCC_CV(X, w, gamma.seq, nfolds = 5, nu = 1/nrow(X), verbose = FALSE,
  tol.base = 1e-04, tol.miss = 1e-04, max.iter.base = 5000,
  max.iter.miss = 500, parallel = FALSE, frac = 1)
```

Arguments

X	A subject (n) by variable (p) matrix; the data
w	A vector of length p-1; weights for clustering
gamma.seq	A vector of positive scalars; regularization parameter sequence
nfolds	A positive scalar; number of cross validation folds
nu	A positive scalar; augmented Lagrangian paramter
verbose	Logical; should messages be printed?
tol.base	A small positive scalar; convergence tolerance for base SpaCC problem.
tol.miss	A small positive scalar; convergence tolerance for missing data problem.
max.iter.base	A positive integer; maximum number of iterations for base SpaCC problem
max.iter.miss	A positive integer; maximum number of iterations for missing data problem
parallel	A logical; should CV paths be done in parallel?
frac	A positive scalar between 0 and 1; fraction of hold out set to utilize

Value

A list with elements: ErrMat - a length(gamma.seq) by nfold matrix containing error on out of fold data; SpMat - a length(gamma.seq) by nfold matrix containing sparsity levels; gamma.seq - original gamma.seq sorted largest to smallest

Examples

```
library(dplyr)
library(tidyr)
data("methy")
methy <- methy[1:20,1:10]
Coordinates <- methy$Genomic_Coordinate
methy %>%
 tbl_df() %>%
 select(-Chromosome,-Genomic_Coordinate) %>%
 gather(Subject,Value,-ProbeID) %>%
```

```
 spread(ProbeID,Value) -> X
SubjectLabels <- X$Subject
X <- X[,-1] %>% as.matrix()
nsubj <- nrow(X)
nprobes <- ncol(X)
nweights <- choose(nprobes,2)
diff.vals <- diff(Coordinates)
too.far <- diff.vals > 20000
sig = 1/5e3
w.values <- exp(-sig*diff.vals)
w.values[too.far] = 0

verbose=TRUE
tol.base = 1e-4
tol.miss = 1e-4
max.iter.base=5000
max.iter.miss=500
ngam = 20
gamma.seq <- exp(seq(log(1e-1),log(1e1),length.out=ngam))
CVRes <- SpaCC_CV(X=t(scale(t(X),center=TRUE,scale=FALSE)),
                  w=w.values,
                  gamma.seq=gamma.seq,
                  nfolds=5,
                  nu=1/nsubj,
                  verbose=TRUE,
                  tol.base=tol.base,
                  tol.miss=tol.miss,
                  max.iter.base=max.iter.base,
                  max.iter.miss=max.iter.miss,
                  parallel=FALSE,frac = 1)
```

SpaCC_Methy *Performs Spatial Convex Clustering for methylation data*

Description

Performs Spatial Convex Clustering for methylation data

Usage

```
SpaCC_Methy(X, Coordinates, gamma.seq, dist.cutoff = 20000, sig = 1/5000,
  weights = NULL, center = TRUE, scale = FALSE, nfolds = 5, nu = NULL,
  tol.base = 1e-04, tol.miss = 1e-04, max.iter.base = 5000,
  max.iter.miss = 500, frac = 0.1, parallel = FALSE, gam.rule = 2,
  thresh.mult = 1, thresh.value = NULL)
```

Arguments

X	A subject (n) by variable (p) matrix; the data

Coordinates	a vector listing genomic coordinates
gamma.seq	a vector of regularization parameters
dist.cutoff	maximum distance at which probes should be regularized
sig	positive scalar controling spatial weight decay
weights	a vector of spatial weights
center	should data be centered
scale	should data be scaled
nfolds	number of folds for cross validation
nu	parameter for augmented lagrangian
tol.base	tolerance level for base function
tol.miss	tolerance for missing function
max.iter.base	maximum number of iterations for base function
max.iter.miss	maximum number of iterations for missing function
frac	fration of fold to use for cross validation
parallel	should algorithm be run in parallel
gam.rule	cross validation rule
thresh.mult	multiplier for threshold value
thresh.value	value of threshold

Value

Labels a vector of cluster labels

Examples

```
data("methy")
methy <- methy[1:20,1:10]
library(dplyr)
library(tidyr)
Coordinates <- methy$Genomic_Coordinate
methy %>%
 tbl_df() %>%
 select(-Chromosome,-Genomic_Coordinate) %>%
 gather(Subject,Value,-ProbeID) %>%
 spread(ProbeID,Value) -> X
SubjectLabels <- X$Subject
X <- X[,-1] %>% as.matrix()
verbose=TRUE
tol.base = 1e-4
tol.miss = 1e-4
max.iter.base=5000
max.iter.miss=500
ngam = 20
gamma.seq <- exp(seq(log(1e-1),log(1e1),length.out=ngam))
ClusterLabels <- SpaCC_Methy(X = X,Coordinates = Coordinates,gamma.seq = gamma.seq)
```

SpaCC_Missing *Solve Spatial Convex Clustering problem for missing data*

Description

Solve Spatial Convex Clustering problem for missing data

Usage

```
SpaCC_Missing(X, w, gamma, nu = 1/nrow(X), verbose = FALSE,
  tol.base = 1e-04, tol.miss = 1e-04, max.iter.base = 5000,
  max.iter.miss = 500, Uinit, Vinit, Laminit)
```

Arguments

X	A subject (n) by variable (p) matrix; the data
w	A vector of length p-1; weights for clustering
gamma	A positive scalar; regularization parameter
nu	A positive scalar; augmented Lagrangian paramter
verbose	Logical; should messages be printed?
tol.base	A small positive scalar; convergence tolerance for base SpaCC problem.
tol.miss	A small positive scalar; convergence tolerance for missing data problem.
max.iter.base	A positive integer; maximum number of iterations for base SpaCC problem
max.iter.miss	A positive integer; maximum number of iterations for missing data problem
Uinit	An n by p matrix; initial value for U
Vinit	An n by p-1 matrix; initial value for V
Laminit	An n by p-1 matrix; initial value for Lam

Value

A list with elements U,V, and Lam

Examples

```
library(dplyr)
library(tidyr)
data("methy")
methy <- methy[1:20,1:10]
Coordinates <- methy$Genomic_Coordinate
methy %>%
  tbl_df() %>%
  select(-Chromosome,-Genomic_Coordinate) %>%
  gather(Subject,Value,-ProbeID) %>%
  spread(ProbeID,Value) -> X
SubjectLabels <- X$Subject
```

```
X <- X[,-1] %>% as.matrix()
X[1:5,1:5]
nsubj <- nrow(X)
nprobes <- ncol(X)
nweights <- choose(nprobes,2)
diff.vals <- diff(Coordinates)
too.far <- diff.vals > 20000
sig = 1/5e3
w.values <- exp(-sig*diff.vals)
w.values[too.far] = 0

verbose=TRUE
tol.base = 1e-4
tol.miss = 1e-4
max.iter.base=5000
max.iter.miss=500
bo <-t(scale(t(X),center=TRUE,scale=FALSE))
bo[is.na(bo)] <- mean(bo,na.rm=TRUE)
best.gam = 1
Sol <- SpaCC_Missing(t(scale(t(X),center=TRUE,scale=FALSE)),
                         w.values,
                         gamma = best.gam,
                         nu=1/nsubj,
                         verbose=TRUE,
                         tol.base=tol.base,
                         tol.miss=tol.miss,
                         max.iter.base=max.iter.base,
                         max.iter.miss=max.iter.miss,
                         bo,
                         t(diff(t(bo))),
                         t(diff(t(bo))))
```

SpaCC_Path *Solve Spatial Convex Clustering problem for path of regularization parameters*

Description

Solve Spatial Convex Clustering problem for path of regularization parameters

Usage

```
SpaCC_Path(X, w, gamma.seq, nu = 1/nrow(X), verbose = FALSE,
  tol.base = 1e-04, tol.miss = 1e-04, max.iter.base = 5000,
  max.iter.miss = 500)
```

Arguments

X	A subject (n) by variable (p) matrix; the data

w	A vector of length p-1; weights for clustering
gamma.seq	A vector of positive scalars; regularization parameter sequence
nu	A positive scalar; augmented Lagrangian paramter
verbose	Logical; should messages be printed?
tol.base	A small positive scalar; convergence tolerance for base SpaCC problem.
tol.miss	A small positive scalar; convergence tolerance for missing data problem.
max.iter.base	A positive integer; maximum number of iterations for base SpaCC problem
max.iter.miss	A positive integer; maximum number of iterations for missing data problem

Value

A list with elements UPath, VPath, LamPath, and gamma.seq

Examples

```
NULL
```

`SpaCC_Path_Parallel` *Solve Spatial Convex Clustering problem for path of regularization parameters in parallel*

Description

Solve Spatial Convex Clustering problem for path of regularization parameters in parallel

Usage

```
SpaCC_Path_Parallel(X, w, gamma.seq, nu = 1/nrow(X), verbose = FALSE,
  tol.base = 1e-04, tol.miss = 1e-04, max.iter.base = 5000,
  max.iter.miss = 500, ncores = 2)
```

Arguments

X	A subject (n) by variable (p) matrix; the data
w	A vector of length p-1; weights for clustering
gamma.seq	A vector of positive scalars; regularization parameter sequence
nu	A positive scalar; augmented Lagrangian paramter
verbose	Logical; should messages be printed?
tol.base	A small positive scalar; convergence tolerance for base SpaCC problem.
tol.miss	A small positive scalar; convergence tolerance for missing data problem.
max.iter.base	A positive integer; maximum number of iterations for base SpaCC problem
max.iter.miss	A positive integer; maximum number of iterations for missing data problem
ncores	A positive integer; number of cores to use

Value

A list with elements UPath, VPath, LamPath, and gamma.seq

Examples

```
NULL
```

`ThreshV` *Threshold differences*

Description

Threshold differences

Usage

```
ThreshV(V, X, mult = 1, thresh.value = NULL)
```

Arguments

`V`	an n x p-1 matrix of differences
`X`	an n x p matrix
`mult`	scalar to multiply standard deviation
`thresh.value`	optional user specified threshold value.

Value

VThreshed an n x p-1 matrix of thresholded differences

Examples

```
NULL
```

Index