Package 'DynClust'

February 19, 2015

Type Package

Title Denoising and clustering for dynamical image sequence (2D or 3D)+T

Version 3.13

Date 2014-02-18

Author Yves Rozenholc (MAP5, Univ. Paris Descartes), Christophe Pouzat (MAP5, Univ. Paris Descartes) and Tiffany Lieury (Cerebral Physiology lab, Univ. Paris Descartes)

Maintainer Yves Rozenholc <yves.rozenholc@parisdescartes.fr>

Description DynClust is a two-stage procedure for the denoising and clustering of stack of noisy images acquired over time. Clustering only assumes that the data contain an unknown but small number of dynamic features. The method first denoises the signals using local spatial and full temporal information. The clustering step uses the previous output to aggregate voxels based on the knowledge of their spatial neighborhood. Both steps use a single key-tool based on the statistical comparison of the difference of two signals with the null signal. No assumption is therefore required on the shape of the signals. The data are assumed to be normally distributed (or at least follow a symmetric distribution) with a known constant variance. Working pixelwise, the method can be time-consuming depending on the size of the data-array but harnesses the power of multicore cpus.

License MIT + file LICENSE

Depends R (>= 2.10), parallel

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-25 11:02:51

R topics documented:

`DynClust-package` *Denoising and clustering for dynamical image sequence (2D or 3D)+T*

Description

DynClust is a two-stage procedure for the denoising and clustering of stack of noisy images acquired over time. Clustering only assumes that the data contain an unknown but small number of dynamic features. The method first denoises the signals using local spatial and full temporal information. The clustering step uses the previous output to aggregate voxels based on the knowledge of their spatial neighborhood. Both steps use a single keytool based on the statistical comparison of the difference of two signals with the null signal. No assumption is therefore required on the shape of the signals. The data are assumed to be normally distributed (or at least follow a symmetric distribution) with a known constant variance. Working pixelwise, the method can be time-consuming depending on the size of the data-array but harnesses the power of multicore cpus.

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

`adu340_4small` *Calcium-imaging dataset using Fura-2*

Description

adu340_4small is the result from a calcium-buffering experiment performed in a glomerulus of the Cockroach (*Periplaneta americana*) antennal lobe, where the projection neurons were imaged using Fura-2. Calcium variations were measured through time and space at 340 nm.

Usage

```
data(adu340_4small)
```

Format

50x50x128 array of integers

Author(s)

Debora Fusca at Kloppenburg Lab –University of Cologne, Germany

GetClusteringResults *Get clustering step result*

Description

GetClusteringResults returns the results of the clustering procedure RunClustering

Usage

```
GetClusteringResults(data.array, res.listdenois, res.cluster)
```

Arguments

data.array	a (2D or 3D)+T array containing the original dynamic sequence of images (the dataset). The last dimension is the time.
res.listdenois	the list resulting from the RunDenoising procedure applied to data.array. This parameter may be replaced by the component info.den of the former.
res.cluster	the list resulting from a call to RunClustering

Value

a list containing two components clust.array and clust.map. clust.array is an array with same dimension as the original sequence data.array containing the clustered version. clust.map is an array with only spatial dimensions of data.array whose elements provide the cluster number at each location.

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

References

Rozenholc, Y. and Reiss, M. (2012) *Preserving time structures while denoising a dynamical image*, Mathematical Methods for Signal and Image Analysis and Representation (Chapter 12), Florack, L. and Duits, R. and Jongbloed, G. and van~Lieshout, M.-C. and Davies, L. Ed., Springer-Verlag, Berlin

Lieury, T. and Pouzat, C. and Rozenholc, Y. (submitted) *Spatial denoising and clustering of dynamical image sequence: application to DCE imaging in medicine and calcium imaging in neurons*

See Also

RunDenoising, RunClustering

Examples

```
## Not run:
    library(DynClust)

   ## use fluorescence calcium imaging of neurons performed with Fura 2 excited at 340 nm
    data('adu340_4small',package='DynClust')

    ## Gain of the CCD camera:
    G <- 0.146
    ## readout variance of the CCD camera:
    sro2 <- (16.4)^2
    ## Stabilization of the variance to get a normalized dataset (variance=1)
    FT <- 2*sqrt(adu340_4small/G + sro2)
    FT.range = range(FT)

    ## launches the denoising step on the dataset with a statistical level of 5%
    FT.den.tmp <- RunDenoising(FT,1,mask.size=NA,nproc=2)

    ## launches the clustering step on the dataset with a statistical level of 5%
    FT.clust.tmp  <- RunClustering(FT,FT.den.tmp,nproc=2)
    n.cluster <- length(FT.clust.tmp$clusters)
    print(paste(n.cluster,'clusters using variance set to',sqrt(FT.den.tmp$var),'^2'))

    ## get the classified version of the data array and the map of the clusters
    FT.clust.res <- GetClusteringResults(FT,FT.den.tmp,FT.clust.tmp)

    ## plotting results of the clusterization
    par(mfrow=c(2,2))
    image(FT.clust.res$clust.map,col=rainbow(n.cluster))
    title('Cluster map')
    matplot(FT.clust.res$clust.center,col=rainbow(n.cluster),type="l",lwd=0.1,lty=1)
    title('Cluster centers')

    ## and more: original and clustered slices at time 50
    image(FT[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
    title('Original sequence at time 50')
   image(FT.clust.res$clust.array[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
    title('Clustered sequence at time 50')

    ##############################################################################
    ## reapply clustering with twice the nominal variance: forces stronger clustering ##
    ##############################################################################

    ## launches the denoising step on the dataset with a statistical level of 5%
    FT.den.tmp <- RunDenoising(FT,2,mask.size=NA,nproc=2)

    ## launches the clustering step on the dataset with a statistical level of 5%
    FT.clust.tmp  <- RunClustering(FT,FT.den.tmp,nproc=2)
    n.cluster <- length(FT.clust.tmp$clusters)
    print(paste(n.cluster,'clusters using twice the nominal variance'))
```

```
## get the classified version of the data array and the map of the clusters
FT.clust.res <- GetClusteringResults(FT,FT.den.tmp,FT.clust.tmp)

## plotting results of the clusterization
par(mfrow=c(2,2))
image(FT.clust.res$clust.map,col=rainbow(n.cluster))
title('Cluster map')
matplot(FT.clust.res$clust.center,col=rainbow(n.cluster),type="l",lwd=0.1,lty=1)
title('Cluster centers')

## and more: original and clustered slices at time 50
image(FT[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
title('Original sequence at time 50')
image(FT.clust.res$clust.array[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
title('Clustered sequence at time 50')

## End(Not run)
```

`GetDenoisingResults` *Get denoising step result*

Description

`GetDenoisingResults` returns the denoised version of a dynamical image sequence as an array having the same dimensions as the original sequence.

Usage

```
GetDenoisingResults(data.array, res.listdenois)
```

Arguments

`data.array`	a (2D or 3D)+T array containing the original dynamic sequence of images (the dataset). The last dimension is the time.
`res.listdenois`	the list resulting from the `RunDenoising` procedure applied to `data.array`. This parameter may be replaced by the component `info.den` of the former.

Value

an array with same dimension as `data.array` containing the denoised version.

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

References

Rozenholc, Y. and Reiss, M. (2012) *Preserving time structures while denoising a dynamical image*, Mathematical Methods for Signal and Image Analysis and Representation (Chapter 12), Florack, L. and Duits, R. and Jongbloed, G. and van~Lieshout, M.-C. and Davies, L. Ed., Springer-Verlag, Berlin

Lieury, T. and Pouzat, C. and Rozenholc, Y. (submitted) *Spatial denoising and clustering of dynamical image sequence: application to DCE imaging in medicine and calcium imaging in neurons*

See Also

RunDenoising

Examples

```
## Not run:
    library(DynClust)

   ## use fluorescence calcium imaging of neurons performed with Fura 2 excited at 340 nm
    data('adu340_4small',package='DynClust')

    ## Gain of the CCD camera:
    G <- 0.146
    ## readout variance of the CCD camera:
    sro2 <- (16.4)^2
    ## Stabilization of the variance to get a normalized dataset (variance=1)
    FT <- 2*sqrt(adu340_4small/G + sro2)
    FT.range = range(FT)

    ## launches the denoising step on the dataset with a statistical level of 5%
    FT.den.tmp <- RunDenoising(FT,1,mask.size=NA,nproc=2)

    ## get the results of the denoising step
    FT.den.res <- GetDenoisingResults(FT,FT.den.tmp)

    ## plot results at time 50 in same grey scale
    par(mfrow=c(1,3))
    image(FT[,,50],zlim=FT.range,col=gray(seq(0,1,l=128)))
    title('Original')
    image(FT.den.res[,,50],zlim=FT.range,col=gray(seq(0,1,l=128)))
    title('Denoised')
    image(FT.den.res[,,50]-FT[,,50],col=gray(seq(0,1,l=128)))
    title('Residuals')

## End(Not run)
```

`MultiTestH0` *Statistical test of zero mean for dynamics*

Description

MultiTestH0 tests if each column vectors of a matrix seen as a noisy dynamic is of zero mean (H_0) or not. The multiple statistical test assumes known variance and is based on a multiple χ^2 test.

Usage

```
MultiTestH0(proj.matrix, data.var, thrs = NULL)
```

Arguments

`proj.matrix`	a matrix whose columnns are tested to have zero mean or not.
`data.var`	a numeric providing the known variance
`thrs`	a numeric vector of thresholds specified as the $1 - \alpha$ quantiles of the multiple test under the null on each partition. If thrs=NULL (default) return the global p-value which is the minimum of all p-values obtained on each partition.

Value

If thrs is provide returns a Boolean vector with length the number of columns of proj.matrix. Element j is TRUE if the null hypothesis (no difference with the null vector) is accepted for column j of proj.matrix. Otherwise, return one p-value per column.

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

References

Baraud Y., Huet S., Laurent B. *Ann. Stat.* (2003) Durot C., Rozenholc Y. *Methods Math. Stat.* (2006)

`RunClustering` *Clustering of a dynamical image sequence*

Description

Clusters dynamics of an image sequence. Assumes prior sequence denoising.

The clustering procedure is an iterative procedure. At each step, the (available) children of the voxel associated to a largest neighborhoods (result of the denoising step or made of its children as a result of the getChildren function) are used to build a robust cluster. The center of the latter is compared to previously build ones using the MultiTestH0 procedure. The clusters with equivalent centers are merged together until no further merging are possible. The resulting cluster is added to the cluster list if the number of outliers generated by the robust cluster procedure does not exceed the original number of children. If not the voxel is added to the closest existing cluster.

Further details about the clustering procedure can be found in the references.

Usage

```
RunClustering(data.array, denois, nproc = 1, min.size = 1, alpha = 0.05,
  do.children.first = FALSE)
```

Arguments

`data.array`	a (2D or 3D)+T array containing the dynamic sequence of images (the dataset). The last dimension is the time.
`denois`	the result of the denoising procedure RunDenoising.
`nproc`	a numeric value indicating the number of processors used for parallel computation. Default set to 1 (no parallelization).
`min.size`	a numeric value indicating the smallest size of the neighborhood for a voxel to be clusterized. Default set to 1 (no limitation).
`alpha`	a numeric value indicating the global level of the multitest. Default set to 0.05.
`do.children.first`	an boolean. If TRUE compute children list for all voxels before starting the clusterization which will use these lists as neighborhoods.If FALSE (default) neighborhood are those resulting from the denoising step.

Value

a list containing:

- `cluster`, a list of vectors—each vector contains the voxel indexes of one cluster.
- `centers`, a matrix whose columns contain the average dynamics of each cluster.
- `bad.voxels`, vector of non clusterized voxel indexes.

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

References

Rozenholc, Y. and Reiss, M. (2012) *Preserving time structures while denoising a dynamical image*, Mathematical Methods for Signal and Image Analysis and Representation (Chapter 12), Florack, L. and Duits, R. and Jongbloed, G. and van~Lieshout, M.-C. and Davies, L. Ed., Springer-Verlag, Berlin

Lieury, T. and Pouzat, C. and Rozenholc, Y. (submitted) *Spatial denoising and clustering of dynamical image sequence: application to DCE imaging in medicine and calcium imaging in neurons*

See Also

GetClusteringResults

Examples

```
## Not run:
    library(DynClust)

   ## use fluorescence calcium imaging of neurons performed with Fura 2 excited at 340 nm
    data('adu340_4small',package='DynClust')

    ## Gain of the CCD camera:
    G <- 0.146
    ## readout variance of the CCD camera:
    sro2 <- (16.4)^2
    ## Stabilization of the variance to get a normalized dataset (variance=1)
    FT <- 2*sqrt(adu340_4small/G + sro2)
    FT.range = range(FT)

    ## launches the denoising step on the dataset with a statistical level of 5%
    FT.den.tmp <- RunDenoising(FT,1,mask.size=NA,nproc=2)

    ## launches the clustering step on the dataset with a statistical level of 5%
    FT.clust.tmp  <- RunClustering(FT,FT.den.tmp,nproc=2)
    n.cluster <- length(FT.clust.tmp$clusters)
    print(paste(n.cluster,'clusters using variance set to',sqrt(FT.den.tmp$var),'^2'))

    ## get the classified version of the data array and the map of the clusters
    FT.clust.res <- GetClusteringResults(FT,FT.den.tmp,FT.clust.tmp)

    ## plotting results of the clusterization
    par(mfrow=c(2,2))
    image(FT.clust.res$clust.map,col=rainbow(n.cluster))
    title('Cluster map')
    matplot(FT.clust.res$clust.center,col=rainbow(n.cluster),type="l",lwd=0.1,lty=1)
    title('Cluster centers')

    ## and more: original and clustered slices at time 50
    image(FT[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
    title('Original sequence at time 50')
   image(FT.clust.res$clust.array[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
```

```
title('Clustered sequence at time 50')

##############################################################################
## reapply clustering with twice the nominal variance: forces stronger clustering ##
##############################################################################

## launches the denoising step on the dataset with a statistical level of 5%
FT.den.tmp <- RunDenoising(FT,2,mask.size=NA,nproc=2)

## launches the clustering step on the dataset with a statistical level of 5%
FT.clust.tmp  <- RunClustering(FT,FT.den.tmp,nproc=2)
n.cluster <- length(FT.clust.tmp$clusters)
print(paste(n.cluster,'clusters using twice the nominal variance'))

## get the classified version of the data array and the map of the clusters
FT.clust.res <- GetClusteringResults(FT,FT.den.tmp,FT.clust.tmp)

## plotting results of the clusterization
par(mfrow=c(2,2))
image(FT.clust.res$clust.map,col=rainbow(n.cluster))
title('Cluster map')
matplot(FT.clust.res$clust.center,col=rainbow(n.cluster),type="l",lwd=0.1,lty=1)
title('Cluster centers')

## and more: original and clustered slices at time 50
image(FT[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
title('Original sequence at time 50')
image(FT.clust.res$clust.array[,,50],zlim=FT.range,col=grey(seq(0,1,length=n.cluster)))
title('Clustered sequence at time 50')

## End(Not run)
```

`RunDenoising` *Denoising step of a dynamical image sequence*

Description

Performs the denoising step of a dynamic sequence of images. It is also the first step of the clustering.

The denoising procedure is iteratively applied on each voxel.

The denoised version of Fx is obtained with a three stages procedure: 1) Selection of time-homogeneous voxels in the sub-mask around the voxel of interest; 2) Growth of spatial neighborhoods made of time-homogeneous voxels obtained at stage 1 with sizes growing geometrically—each neiborhood is associated to a denoised version by averaging over its members; 3) Selection of the largest spatial neighborhood such that its associated denoised version is time-homogeneous with all the previous ones.

Time homogeneity is tested with function `MultiTestH0`.

Further details about the denoising method and the statistical test of homogeneity can be found in the references.

Usage

```
RunDenoising(data.array, data.var = 1, depth = 1, alpha = 0.05,
  mask.size = NA, nproc = 1, enhStart = ifelse(is.null(var), 2, 1))
```

Arguments

data.array	a (2D or 3D)+T array containing the dynamic sequence of images (the dataset). The last dimension is the time.
data.var	a numeric indicating the variance of the dataset (default 1). If set to NULL, the variance is computed using a baseline image. See enhStart parameter.
depth	a numeric indicating the depth of a voxel.
alpha	a numeric value indicating the global level of the multitest.
mask.size	a vector indicating the size of the spatial hypercube defined around voxels used to search for neighbors. If NA (default): sqrt(dim(data.array)[1:length(dim(data.array))-1]). If NULL (complete image): dim(data.array)[1:length(dim(data.array))-1]
nproc	a numeric value indicating the number of processors used for parallel computation.
enhStart	an integer, if larger than 1, a baseline is computed as a median image obtain from time indexes between 1 and enhStart-1. Default value ifelse(is.null(var),2,1).

Value

a list containing:

- info.den, a list of list whose length is the number of voxels, each sub-list contains the result of buildEstimate for one voxel.
- data.proj, the projections of the dynamics. a list containing a denoised version of the dataset as an array, as well as a list for which each element contains a list with the voxel index, the indexes of its neighboors, the resulting denoised signal, and the variance of the denoised signal
- var, a numeric providing the known variance

Author(s)

Tiffany Lieury, Christophe Pouzat, Yves Rozenholc

References

Rozenholc, Y. and Reiss, M. (2012) *Preserving time structures while denoising a dynamical image*, Mathematical Methods for Signal and Image Analysis and Representation (Chapter 12), Florack, L. and Duits, R. and Jongbloed, G. and van~Lieshout, M.-C. and Davies, L. Ed., Springer-Verlag, Berlin

Lieury, T. and Pouzat, C. and Rozenholc, Y. (submitted) *Spatial denoising and clustering of dynamical image sequence: application to DCE imaging in medicine and calcium imaging in neurons*

See Also

GetDenoisingResults, MultiTestH0.

Examples

```
## Not run:
    library(DynClust)

   ## use fluorescence calcium imaging of neurons performed with Fura 2 excited at 340 nm
    data('adu340_4small',package='DynClust')

    ## Gain of the CCD camera:
    G <- 0.146
    ## readout variance of the CCD camera:
    sro2 <- (16.4)^2
    ## Stabilization of the variance to get a normalized dataset (variance=1)
    FT <- 2*sqrt(adu340_4small/G + sro2)
    FT.range = range(FT)

    ## launches the denoising step on the dataset with a statistical level of 5%
    FT.den.tmp <- RunDenoising(FT,1,mask.size=NA,nproc=2)

    ## get the results of the denoising step
    FT.den.res <- GetDenoisingResults(FT,FT.den.tmp)

    ## plot results at time 50 in same grey scale
    par(mfrow=c(1,3))
    image(FT[,,50],zlim=FT.range,col=gray(seq(0,1,l=128)))
    title('Original')
    image(FT.den.res[,,50],zlim=FT.range,col=gray(seq(0,1,l=128)))
    title('Denoised')
    image(FT.den.res[,,50]-FT[,,50],col=gray(seq(0,1,l=128)))
    title('Residuals')

## End(Not run)
```

Index