Package ‘FCSlib’

November 18, 2020

Type Package

Title A Collection of Fluorescence Fluctuation Spectroscopy Analysis Methods

Version 1.3.0

Date 2020-11-17

Author Raúl Pinto Cámara, Adán Guerrero, Alejandro Linares, José Damián Martínez Reyes, Haydee Hernández.

Maintainer Raúl Pinto Cámara <`support.fcslib@mail.ibt.unam.mx`>

Description This is a package for fluorescence fluctuation spectroscopy data analysis methods such as spFCS, FCCS, scanning-FCS, pCF, N&B and pCOMB, among others. In addition, several data detrending tools are provided. For an extensive user's guide for the use of FCSlib, please navigate to (<https://github.com/FCSlib/FCSlib/tree/master/Documentation>). Sample data can be found at (<https://github.com/FCSlib/FCSlib/tree/master/Sample%20Data>). The original paper where this package is presented can be found at (<doi:10.1093/bioinformatics/btaa876>).

License GPL-3

Depends R(>= 3.6.0), tiff, stringr, bitops, fields

RoxygenNote 7.1.1

Encoding UTF-8

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2020-11-18 09:40:09 UTC

R topics documented:

`asynACTCSPC` *Asynchronous Autocorrelation of Time-Correlated Single-Photon Counting*

Description

Calculates the auto-correlation of the Macrotime data, returning a correlation function.

Usage

```
asynACTCSPC(macro, n = 5, B = 10)
```

Arguments

`macro`	A numeric vector containig a Macrotime Data.
`n`	numeric parameter that represents the number of layers of the cascade.
`B`	numeric parameter that represents the number of values in every layer of the cascade.

Details

This function creates list of tau's with a length of n*B, this list is used to perform the correlation of the data.

Value

A numeric vector G containing either the autocorrelation for the input vector macro, with a length of n*B.

Author(s)

Raúl Pinto Cámara, José Damián Martínez Reyes.

References

wahl, M., Gregor, I., Patting, M. & Enderlein, J. Fast calculation of fluorescence correlation data with asynchronous time-correlated single-photon counting. Opt. Express 11, 3583–3591 (2003).

See Also

`readFileSPC`

Examples

```
spcData <- readFileSPC("atto532_atto655_m1.spc")
asynCorrData <- asynACTCSPC(macro = spcData$MacroTime)
```

`binMatrix` *Binned representation of a matrix*

Description

Groups each column in a matrix into several bins of a given length for better (and faster) data plotting

Usage

```
binMatrix(f, lineTime, nIntervals, columns, mode = "mean", plot = TRUE)
```

Arguments

`f`	A vector or a matrix
`lineTime`	Line (row) acquisition rate (in seconds)
`nIntervals`	Number of intervals into which the all columns will be grouped
`columns`	Number of columns of the resulting binned matrix
`mode`	Set to "mean" (default) or "sum" to average or sum all the points in every interval, respectively
`plot`	Boolean, set to TRUE (default) to plot the result

Details

This function groups all the points in each column of the matrix 'f' into 'nIntervals' bins of length = length(f)/nIntervals. Then, averages or sums all of the points in each bin and plots the result. If 'f' is a vector, 'columns' is used to build the resulting matrix. If 'f' is a matrix, then 'columns' takes the value of the number of columns in 'f'.

Value

A matrix of 'nIntervals' rows

Author(s)

Alejandro Linares

See Also

binTimeSeries

Examples

```
### Please navigate to
### (https://github.com/FCSlib/FCSlib/tree/master/Sample%20Data)
### to find this sample data

# Automatic plot
x <- read.table("Pax.dat")
x <- binMatrix(x[,1], lineTime =  1e-3, nIntervals =  500,
               columns = 64, mode = "mean", plot = T)

# Manual plot (useful for adding custom labels)
x <- read.table("Pax.dat")
x <- binMatrix(x[,1], lineTime =  1e-3, nIntervals =  500,
               columns = 64, mode = "mean", plot = F)
image.plot(x)
```

---

binTimeSeries *Binned representation of a time series*

Description

Groups large vectors into several bins of a given length for better (and faster) data plotting

Usage

```
binTimeSeries(f, acqTime, nIntervals, mode = "mean", plot = TRUE)
```

Arguments

f	Numeric vector
acqTime	Point acquisition rate (in seconds)
nIntervals	Number of intervals into which the vector will be grouped
mode	Set to "mean" (default) or "sum" to average or sum all the points in every interval, respectively
plot	Boolean, set to TRUE (default) to plot the result

Details

This function groups all the points in the vector 'f' into 'nIntervals' bins of length = length(f)/nIntervals. Then, averages or sums all of the points in each bin and plots the result.

Value

A data frame with two variables (Counts and Time) and 'nIntervals' observations

Author(s)

Alejandro Linares

See Also

binMatrix

Examples

```
### Please navigate to
### (https://github.com/FCSlib/FCSlib/tree/master/Sample%20Data)
### to find this sample data

# Automatic plot
x <- readFileTiff("Cy5.tif")
x <- as.vector(x)
x <- binTimeSeries(x[length(x):1], 2e-6, 100, mode = "mean", plot = T)

# Manual plot (useful for adding custom labels)
x <- readFileTiff("Cy5.tif")
x <- as.vector(x)
x <- binTimeSeries(x[length(x):1], 2e-6, 100, mode = "mean", plot = F)
plot(x$Counts~x$Time, type = "l")
```

`Cy5` *Experimental data of freely diffusing Cy5 dye in water at a concentration of 33 nM.*

Description

A single vector of single-point scan data, in the form of a TIFF file.

Usage

```
data(Cy5)
```

Format

A matrix with 2048 columns and 5000 rows

Details

To analyze this data set with FCSlib, import by typing: readFileTiff("Cy5.tif")

`detrendTimeSeries` *Algorithms for data detrending*

Description

Allows to perform Exponential, Polynomial or Boxcar Filter detrending over a time series vector

Usage

```
detrendTimeSeries(f, acqTime, nIntervals, algorithm, degree, w, pois = FALSE,
 max = FALSE, plot = TRUE)
```

Arguments

`f`	A vector
`acqTime`	Point acquisition rate (in seconds)
`nIntervals`	Number of intervals into which the vector will be grouped prior to the detrending process
`algorithm`	A character string. Choose between Exponential ("exp"), Polynomial ("poly") or Boxcar Filter ("boxcar") detrending
`degree`	The degree of the polynomial function
`w`	Moving average time window size
`pois`	Logical, set to TRUE for detrending performance by adding random, uncorrelated numbers sampled from a Poisson distribution (see details)
`max`	Logical, set to TRUE for detrending performance based on the highest value of the original data, rather than the first one (see details)
`plot`	Logical, set to TRUE (default) to plot de result

Details

First, the binTimeSeries() function is used to obtain a binned version of ’f’ of ’nIntervals’ points.

For exponential detrending, a model of the form (A0*e^(k*t) is adjusted to the binned vector.

A polynomial function of user-specified degree is rather used for polynomial detrending.

For the case of boxcar filtering, the moving average vector is calculated from the original series. An amount of zeroes equal to (w-1) is added at the tail of ’f’ to compensate for the moving average effect when position (length(f) - w + 1) is reached.

In either case, the residuals are then obtained and added a constant value for trend correction. When 'max' is set to TRUE, said value will be the highest in the binned vector of 'f'.

When 'pois' is set to FALSE (default), the trend correction value is directly added to the obtained residuals, in a quantity that will make the average counts remain constant throughout the whole time series. On the other hand, when 'pois = TRUE', the trend correction value is instead used as the 'lambda' parameter for a Poisson distribution from which uncorrelated counts will be randomly sampled and added to the whole series for trend correction. This procedure asures that only integer counts will be obtained after detrending, at the cost of adding some noise and making the detrending process a lengthier task.

Value

A vector

Author(s)

Alejandro Linares, Ad?n Guerrero, Haydee Hern?ndez

See Also

`binTimeSeries`

`fcs` *Fluorescence Correlation Spectroscopy*

Description

Calculates either the auto-correlation or cross-correlation between vectors x and y, returning a correlation function.

Usage

```
fcs(x , y = NULL, nPoints, pcf = FALSE)
```

Arguments

`x`	Numeric vector of length N.
`y`	Numeric vector of length N.
`nPoints`	The size of the sub-vectors in which the input vectors will be divided. This number must be less than N/2.
`pcf`	A boolean parameter to determine if an alternate version of the correlation function is used for the calculation of de pCF and pComb functions.

Details

Fluorescence correlation spectroscopy (FCS) is a technique with high spatial and temporal resolution used to analyze the kinetics of particles diffusing at low concentrations. The detected fluorescence intensity as a function of time is: F(t).

The correlation function is computed as the normalized autocorrelation function, G(tau) = <deltaF(t)deltaF(t+tau)>/(<F(t)>*<
to the collected data set, where t refers to a time point of flourescence acquisition, and tau refers to the temporal delay between acquisitions and <...> indicates average.

The correlation between deltaF(t) = F(t) - <F(t)> and deltaF(t+tau) = F(t+tau) - <F(t)> is calculated for a range of delay times. For temporal acquisitions as FCS point, x takes the value of F(t) and y = NULL. For cross-correlation experiments between two fluorescent signals x = F1(t) and y = F2(t), as channels, the correlation function is: G(tau) = <deltaF1(t) deltaF2(t+tau)> / (<F1(t)> <F2(t)>).

The function separate the original vector in sub-vectors of same length (n-points), then calculate an autocorrelation function form each sub-vector. The final result will be an average of all the autocorrelation functions.

Value

A numeric vector G containing either the autocorrelation for the input vector x, or the cross-correlation between x and y vectors, with a length of nPoints.

Note

The argument nPoints must be smaller than the total number of temporal observations N, it is recommended to set nPoints = 2^n, with n = 2, ..., infinity.

Author(s)

Raúl Pinto Cámara, Adan O. Guerrero

References

R.A. Migueles-Ramirez, A.G. Velasco-Felix, R. Pinto-Cámara, C.D. Wood, A. Guerrero. Fluorescence fluctuation spectroscopy in living cells. Microscopy and imaging science: practical approaches to applied research and education, 138-151,2017.

See Also

`gcf`

Examples

```
### Load the FCSlib package

library(FCSlib)

# As an example, we will use data from experiment adquisition
# of free Cy5 molecules diffusing in water at a concentration of 100 nM.
# Use readFileTiff() function to read the fcs data in TIFF format.
```

```
f<-readFileTiff("Cy5_100nM.tif")

### Note that $f$ is a matrix of 2048 x 5000 x 1 dimentions.
# This is due to the fact that this single-point FCS experimen twas collected
# at intervals of 2048 points each, with an acquisition time of 2 mu s.
# Let's now create a dataframe with the FCS data wich here-and-after will be called Cy5.

acqTime = 2E-6
f<-as.vector(f)
time <- (1:length(f))*acqTime
Cy5<-data.frame(t = time, f)

### The first 100 ms of the time series are:

plot(Cy5[1:5000,], type ="l", xlab = "t(s)", ylab ="Fluorescence Intensity", main = "Cy5")

# The fcs() function receives three parameters: 'x' (mandatory),
# 'y'(optional) and 'nPoints' (optional), where x is the main signal to analyze,
#  y is a secondary signal (for the case of cross-correlation instead of autocorrelation)
# and nPoints is the final length of the calculated correlation curve.
# This function divides the original N-size signal into sub-vectors with a size of nPoints*2.
# Once all the sub-vectors are analyzed, these are then averaged.
# To use the fcs() function type

g <- fcs(x = Cy5$f, nPoints = length(Cy5$f)/2)

# The result of the function is assigned to the variable 'g',
# which contains the autocorrelation curve

length <- 1:length(g)
tau <-Cy5$t[length]
G<-data.frame(tau,g)
plot(G, log = "x", type = "l", xlab = expression(tau(s)), ylab = expression(G(tau)), main = "Cy5")

# It is important to remove the first point from the data,
# where G(\tau=0) it is not properly computed

G<-G[-1,]
plot(G, log = "x", type = "l", xlab = expression(tau(s)), ylab = expression(G(tau)), main = "Cy5")

# The variable 'nPoints' can be adjusted to better assess the transport phenomena
# in study (i.e. free diffusion in three dimensions in the case of this example) and
# for better understanding of the diffusive nature of the molecules.
# In this example 'nPoints' will be set to 2048.

g <- fcs(x = Cy5$f,nPoints = 2048)
length <- 1:length(g)
tau <-Cy5$t[length]
G<-data.frame(tau,g)
G<-G[-1,]
plot(G, log = "x", type = "l", xlab = expression(tau(s)), ylab = expression(G(tau)), main = "Cy5")
```

`fitFCS` *Fitting FCS Data*

Description

Estimates the parameters based on a given equation, on the data generated with the fcs() function.

Usage

```
fitFCS(data = parent.frame(), start, low = -Inf, up = Inf,
              type = "D3D", model = NULL, trace = TRUE)
```

Arguments

`data`	data frame in which to evaluate the variables in formula and weights.
`start`	a named list or named numeric vector of starting estimates.
`low, up`	a named list or named numeric vector of lower and upper bounds, replicated to be as long as start. If unspecified, all parameters are assumed to be -Inf and Inf.
`type`	specification for the equation to model, is a character string. The default value is "D3D" equation for three-dimensional free diffusion. Another possibles values are: "D2D" for two-dimensional free diffusion, "D2DT" for two-dimensional free diffusion with triplet exited state, and "D3DT" for three-dimensional free diffusion with triplet exited state and D3D2S for two species in three-dimensional free diffusion.
`model`	a character type variable, that must contain the custom equation if needed, NULL by default.
`trace`	logical value that indicates whether the progress of the non-linear regression (nls) should be printed.

Details

A transport model, containing physical information about the diffusive nature of the fluorophores, can be fitted to the autocorrelation data to obtain parameters such as the diffusion coeficient D and the number of molecules within the observation volume N.

The fitFCS() function uses the 'Non-linear Least Squares' function to fit a physical model into a data set. There are four possible models to be fit:

"D2D" for two-dimensional diffusion

"D2DT" for two-dimensional diffusion with triplet state

"D3D" for three-dimensional diffusion

"D3DT" for three-dimensional diffusion with triplet state

Inside the equations for each model, gamma a geometric factor that depends on the illumination profile. For confocal excitation its magnitude approaches gamma = 1/sqrt8 ??? 0.35 fl. The diffusion time is defined as tau_D = s^2/4D, where s and u are the radius and the half-length of the focal volume, respectively. The parameter u is usually expressed as u = ks, with k being the eccentricity of the focal volume; for confocal excitation k ??? 3. The fraction of molecules in the triplet state is B, and tau_B is a time constant for the triplet state.

Value

A nls object (from nls).

Author(s)

Raúl Pinto Cámara.

See Also

nls, fcs

Examples

```
# Load the FCSlib package

library(FCSlib)

g <- fcs(x = Cy5$f,nPoints = 2048)
len <- 1:length(g)
tau <-Cy5_100nM$t[len]
G<-data.frame(tau,g)
G<-G[-1,]

# Once the correlation curve 'g' has been generated,
# a data frame containing known parameters must be then defined

df<-data.frame(G, s = 0.27, k = 3)
head(df)

# The radius of the focal volume must computed experimentally.
# For this example, we choose a s = 0.27~ mu m
# Then, three lists that contain the initial values of the data,
# as well as the upper and lower limits of these values, must be defined.
# The input values here are the expected values for the real experimental data
# to be very similar or close to, so that the function calculates them accurately.
# Initial values:

start <- list(D = 100, G0 = 0.1)
up <- list(D = 1E3, G0 = 10)
low <- list(D = 1E-1, G0 = 1E-2)

# Once the known parameters are defined, we now proceed to use the fitFCS() function.
# The result will be a nls object

modelFCS <- fitFCS(df, start, low, up, type = "D3D", trace = F)
# summary(modelFCS)
# By using the predict() function, the object generated in the previous step
# is transformed into a vector that contains the curve fitted by the desired model.

fit <- predict(modelFCS, tau)

# Finally, use the following command to obtain the resulting graph,
# where the blue line corresponds to the fitted data and the black surface
# corresponds to the unfitted

plot (G, log = "x", type = "l", xlab = expression(tau(s)),
      ylab = expression(G(tau)), main = "Cy5")
lines(fit~G$tau, col = "blue")

# To acquire access to the physical coefficients of the model type

s<-summary(modelFCS)
s$coefficients[,1]
```

gcf *General Correlation Function*

Description

Performs either the auto-correlation or cross-correlation between vectors x and y, returning a correlation function.

Usage

```
gcf(x, y, xmean = 1, ymean = 1, c = 0)
```

Arguments

`x`	A numerical signal with dimensions M x N x Z.
`y`	A numerical signal with dimensions M x N x Z.
`xmean`	The mean value of the signal x.
`ymean`	The mean value of the signal y.
`c`	A numeric variable to restrict the correlation to positives values.

Details

The number of emission events per unit time is determined and used to generate autocorrelation and cross-correlation curves from the intensity traces F(t) and the fluctuations deltaF(t) = F(t)-<F(t)>. The auto-correlation function of the collected data set, is computed as the normalized auto-correlation function, when y=x. The general auto-correlation function is defined as: G(tau) = (deltaF(t) deltaF(t+tau))/(<F(t)> <F(t)>), where t refers to a time point of fluorescence acquisition, and tau refers to the temporal delay between acquisitions. <...> is the temporal average of F(t); and deltaF(t) = F(t)-<F(t)>, deltaF(t+tau) = F(t+tau)-<F(t)>.

For temporal acquisitions such as point FCS, x and y are F(t). The cross-correlation function between two channels of fluorescent signals, x = F1(t) and y = F2(t), the cross-correlation function is defined as: G(tau) = (deltaF1(t) deltaF2(t+tau))/(<F1(t)><F2(t)>), where xmean = <F1(t)> and ymean = <F2(t)> are the mean values of the fluorescent signals.

Value

G A numerical signal with dimension N' x M' x Z'

Author(s)

Raúl Pinto Cámara.

References

Siegel, A. P., Hays, N. M., & Day, R. N. (2013). Unraveling transcription factor interactions with heterochromatin protein 1 using fluorescence lifetime imaging microscopy and fluorescence correlation spectroscopy. Journal of biomedical optics, 18(2), 025002.

See Also

`fcs`, `convolve`

Examples

```
# Load the FCSlib package

library(FCSlib)

# As an example, we will use data from experiment adquisition
# of free Cy5 molecules diffusing in water at a concentration of 100 nM.

oldpar <- par(no.readonly = TRUE)
g <- gcf(x = Cy5$f, y = Cy5$f, xmean = mean(Cy5$f), ymean = mean(Cy5$f))
length <- 1:length(g)
par(mfrow=c(1,1))
plot(y = g, x = Cy5$t[length], log = 'x', type = 'l',
xlab = expression(tau(mu~s)), ylab = expression(G(tau)),
main = "Cy5 100nM")
par(oldpar)
```

`nbline` *Number & Brightness (Single Image)*

Description

Performs the Number and Brightness Analysis (N&B) on an image

Usage

```
nbline(img, sigma0 = 0, offset = 0, S = 1, w = 0)
```

Arguments

`img`	The image to analyze.
`sigma0`	Variance of the optical system readout noise
`offset`	Constant number that depends on the optical system configuration. Signal values smaller that the offset should be considered zero.
`S`	Proportionality factor S. Indicates the ratio between the amount inicident photons in the detector and those converted to an electronic signal.
`w`	Time window at which the running average is calculated

Details

The Number and Brightness (N&B) method is a time-independent technique that provides an estimate of molecular concentration and aggregation state (or stoichiometry), based on the statistical moments of the fluorescence intensity fluctuations. In other words, this tool allows to distinguish between two or more homo-oligomeric states of a molecule present in a given region in the sample (Brightness) while also providing a direct indicator of the molecules relative abundance (Number). The intensity of the fluorescence signal is mostly due to the mere presence of fluorophores in the media, affected by the fluorophore quantum yield, the sensitivity of the detector and the photophysical characteristics of the optical instrumentation. The average particle number and brightness are calculated directly from the mean value <k> and variance (sigma^2) of the fluorescence intensity data (image) for a given pixel as follows: N = (<k>^2)/(sigma^2) and B = (sigma^2)/<k>

Value

A list containing two vectors, the Brightness and the Number of the image.

Author(s)

Raúl Pinto Cámara.

See Also

`var`,`mean`

Examples

```


### Load the FCSlib package

library(FCSlib)

# As an example, we will use a data set that corresponds
# to a population of Venus dimers and hexamers diffusing in HEK-293 cells.
# Use the readFileTiff() function to extract the information from the '.tiff' files.
```

```
nbv2 <- nbline(V2)
pixelSize = 0.05
r<- (1:dim(V2)[1])*pixelSize
```

`norm.vector` *Min-Max Feature scaling normalization*

Description

Normalizes a vector using the Min-Max Feature scaling method (a.k.a unity-based normalization)

Usage

```
norm.vector(x, a.b = NULL)
```

Arguments

`x`	A vector
`a.b`	A vector that indicates the minimum and maximum scaling values

Details

Feature scaling is used to bring all values into the range [0,1]. This is also called unity-based normalization. When 'a.b = NULL' (default), the highest and lowest values in 'x' will turn to 1 and 0, respectively, while all values in between will be re-scaled. Defining 'a.b' will bring all values into the range [a,b].

Value

A normalized vector

Author(s)

Alejandro Linares

Examples

```
x <- seq(from = 1, to = 100, by = 0.1)
y <- sin(sqrt(x))
plot(y~x, type = "l")

y.n <- norm.vector(y)
plot(y.n~x, type = "l")

y.ab <- norm.vector(y, a.b = c(5,20))
plot(y.ab~x, type = "l")
```

pcf *Pair Correlation Function*

Description

Calculates the correlation between the pixel i and pixel i + dr.

Usage

```
pcf(img, nPoints = 1000, one.col = FALSE, dr = 1)
```

Arguments

`img`	The image to analyze
`nPoints`	The size of the sub-vectors in which the input vectors will be divided. This number must be less than N/2.
`one.col`	By default FALSE. If TRUE the correlation will be performed in the fixed colum mode, else the distance mode.
`dr`	Distance between pixel at which the correlation is calculated. For a value of delta_r = 3, the columns are correlated as follows: (1,4), (2,5), ..., (n-3, n), with n being the last column.

Details

The pair correlation function (pCF) analyzes data of a periodically scanned line, measuring the time it takes a particle to go from one pixel to another, i.e. calculates the spatial cross-correlation function between pixels. G(tau,deltar) = (<F(t,0) F(t + tau, deltar)>/<F(t,0)> <F(t,deltar)>)-1

Value

An image depicting the correlation between the pixel i and pixel i + dr.

Author(s)

Raúl Pinto Cámara.

See Also

`fcs`,`pcomb`

Examples

```
### Load the FCSlib package

library(FCSlib)

### As an example, we will use a data set that corresponds to a population of Venus dimers
```

```
# diffusing in HEK-293 cells. Use the readFileTiff() function to extract the information
# from the '.tiff' files.

dmv2 <- data.matrix(V2)
pB <- pcf(dmv2, nPoints = 2500, dr = 10)

### Plot the result
library(fields)
di <- dim(pB)
tau <- (1:(di[2]))
image.plot( x = 1:di[1], y = log10(tau), z = pB, main = "Column Distance 10",
xlab = "Pixel", ylab = "Logarithmic tau",
cex.lab = 1.2, cex.main = 1.2, cex.axis = 1)
```

pcomb *Pair Correlation of Molecular Brightness*

Description

Performs the pair correlation of molecular brightness (pCOMB) analysis.

Usage

```
pcomb(img, nPoints = 25000, one.col = FALSE, dr = 1, w = 100)
```

Arguments

`img`	The image to analyze.
`nPoints`	The size of the sub-vectors in which the input vectors will be divided. This number must be less than N/2.
`one.col`	By default FALSE. If TRUE the correlation will be performed in the fixed colum mode, else the distance mode.
`dr`	Is the distance between the two columns that will be correlated. For a value of deltar = 3, the columns are correlated as follows: (1,4), (2,5), ..., (n-3, n), with n as the last column.
`w`	Range value that is used to calculate the brightness in the image.

Details

With the Pair Correlation of Molecular Brightness (pCOMB) method, one can distinguish between different homo-oligomeric species of the same molecule coexisting in the same microenvironment, while separately and specifically tracking each species' moblity across the cellular compartments. This technique amplifies the signal from the brightest species present and filters the dynamics of the extracted oligomeric population based on arrival time between two locations. This method is suitable for mapping the impact oligomerization on transcription factor dynamics. The resulting intensity fluctuations, pCF, are transformed into brightness fluctuations using B = (sigma^2)/mean, and the pair correlation analysis is then performed on the brightness fluctuations along the line scan , at a distance (delta(r)).

If the pcf is set as FALSE the pCcomb data will not be generated and will be NULL. In order to generate that data the pcf function must be used on the BCarpet data.

Value

A list containing the Brightness Carpet and the Pair Correlation of that carpet

Author(s)

Raúl Pinto Cámara.

See Also

`fcs,pcf`

Examples

```
### Load the FCSlib package

library(FCSlib)

# As an example, we will use a data set that corresponds to a population of Venus dimers
# diffusing in HEK-293 cells. Use the readFileTiff() function to extract the information
# from the '.tiff' files.

dmv2 <- data.matrix(V2)
pC <- pcomb(dmv2[1:32,1:2001], nPoints = 1000, type = 'd', dr = 10, w = 2, pcf = FALSE)
dmv2 <- data.matrix(v2DataSet)
pC <- pcomb(dmv2, nPoints = 5000, type = 'd', dr = 10, w = 100)
di <- dim(pC$pComb)
tau <- (1:(di[2]))

# Plot the result
library("fields")
image.plot( x = 1:di[1], y = log10(tau), z = pC$pComb, main = "pComb",
xlab = "Pixel", ylab = "Logarithmic tau",
cex.lab = 1.2, cex.main = 1.2, cex.axis = 1)
```

`readFileFCS` *Read File FCS*

Description

Reads a FCS file and returns the data sets within the file.

Usage

```
readFileFCS(filename)
```

Arguments

`filename`	the name of the file to read from.

Details

Read a FCS file using the scan function and extract the data contained in the file.

Value

dataList A list containing the data sets within the file.

Author(s)

Raúl Pinto Cámara.

Examples

```
raw_fcs <- readFileFCS(FileName)
```

`readFileModel` *Read File Model*

Description

Reads a txt file and returns the parameters and the model (equation).

Usage

```
readFileModel(filename)
```

Arguments

`filename`	The name of the file to read from.

Details

Read a txt file using the scan function and extracts the parameters and the model (equation) in the file.

Value

params A list containing the parameters as well as the model.

Author(s)

Raúl Pinto Cámara.

See Also

`fitFCS`

Examples

```
modelData <- readFileModel(filename)
```

`readFileSPC` *Read File SPC-140/150/130/830*

Description

Reads a SPC file and returns the Macrotime and Microtime.

Usage

```
readFileSPC(filename, nData = 1E8)
```

Arguments

`filename`	the name of the file to read from.
`nData`	parameter that defines the length of data to read.

Details

Read a SPC file, with SPC-140/150/130/830 version, using the readBin function and extract the data contained in the file.

Value

A list containing the Macrotime and the Microtime vectors.

Note

The nData parameter is used to overestimate the amount of data that the file can contain.

Author(s)

Raúl Pinto Cámara, José Damián Martínez Reyes.

References

Becker, W., 2019. The Bh TCSPC Handbook. 8th ed. Berlin, Germany: Becker & Hickl GmbH, pp. 855-856.

See Also

`asynACTCSPC`

Examples

```
spcData <- readFileSPC(FileName)
```

`readFileTiff` *Read File Tiff*

Description

Reads a TIFF file and converts it into a 2D-array. If the file contains multiple pages, a 3D-array will be then returned.

Usage

```
readFileTiff(filename, invert = TRUE)
```

Arguments

`filename`	Either name of the file to read from or a raw vector representing the TIFF file content.
`invert`	If set to TRUE then the order of the data will be reversed. Default TRUE.

Details

Read a TIFF file image using readTIFF and converts it to a matrix with n-dimensions.

Value

A matrix containing the image data.

Note

This function must be used in order to extract the information from the TIFF files needed to test the functions in this package. The TIFF file must be grayscale.

Author(s)

Adan O. Guerrero Cardenas.

See Also

readTIFF writeFileTiff

Examples

```
raw <- readFileTiff(FileName)
```

---

simplifyFCS *Simplify FCS*

Description

Reduces the amount of data in a data set without altering its overall structure

Usage

```
simplifyFCS(g, tau, step = 1)
```

Arguments

g	A vector containing the FCS data analysis
tau	A vector that represents the time frame between data acquisitions
step	A numeric value that affects the final length of the vector

Details

The simplifyFCS function performs a log10 weighted binning of the autocorrelation function (acf). It balance the weight of the long-time scale trending behavior of the acf curve, which commonly contain G(tau) points that fluctuate around the zero-correlation regime, hence overweighting fitting with ‘noisy data’. simplifyFCS reduce the weight of the long-time scale trending behavior (ms to sec), preserving the structure of the short-time scales.

Value

A vector of the FCS data with reduced length

Note

the step parameter must be between 0 and 1

Author(s)

Adan O. Guerrero

See Also

gcf, var,mean

Examples

```
f <- Cy5_100nM$f
acqTime <- 2E-6
f <- as.vector(f)
time <- (1:length(f))*acqTime
cy5 <- data.frame(t = time, f)

g <- fcs(x = cy5$f)
len <- 1:length(g)
tau <-cy5$t[len]
G <- data.frame(tau,g)

sfcs <- simplifyFCS(G$g, G$tau, step = 0.5)
plot(sfcs$g~sfcs$tau, log = "x", type = "l",
     xlab = expression(tau(s)),
     ylab = expression(G(tau)), main = "Cy5")

# Comparison, original with simplify
plot(G, type = 'l', log = 'x')
lines(sfcs$g~sfcs$tau, col = "red")
```

smoothCarpet *Smooth Carpet (Single Image)*

Description

Generates a smooth carpet.

Usage

```
smoothCarpet(img, dfV = 0, dfH = 0)
```

Arguments

img	The image to analyze.
dfV	The desired equivalent number of degrees of freedom in the vertical axis.
dfH	The desired equivalent number of degrees of freedom in the horizontal axis.

Details

The smoothCarpet function makes use of the smooth.spline method to smooth the vertical and horizontal axes of an image. The magnitude of the smoothing depends on the degrees of freedom set for and vertical ('dfV') and horizontal ('dfH') axes of the image.

Value

Smooth Carpet A smooth image.

Author(s)

Raúl Pinto Cámara.

See Also

`pcomb`,`smooth.spline`

Examples

```
### Load the FCSlib package

library(FCSlib)

### As an example, we will use a data set that corresponds to a population of Venus dimers
# diffusing in HEK-293 cells. Use the readFileTiff() function to extract the information
# from the '.tiff' files.

v2 <- data.matrix(V2)
nbv2 <- nbline(img = v2, S=3.5, sigma0 = 1,offset = 0, wSigma = 100);
sC <- smoothCarpet(img = nbv2$number, dfV = 5, dfH = 5)
```

`tiff_to_mtx` *Transformation of multiple-image TIFF files or arrays into a matrix*

Description

Transforms multiple-image TIFF files or 3D arrays into 2D matrices with a user-specified number of columns

Usage

```
tiff_to_mtx(data, columns)
```

Arguments

`data`	A character string indicating the name of a TIFF file or a 3D array
`columns`	The number of columns of the resulting matrix

Details

Creates a matrix with a user-specified number of columns and a number of rows equal to the total amount of points in 'data' divided by 'columns'.

Value

A matrix

Author(s)

Alejandro Linares

See Also

`binMatrix`

Examples

```
### Please navigate to
### (https://github.com/FCSlib/FCSlib/tree/master/Sample%20Data)
### to find this sample data

x <- readFileTiff("Example_file_name.tif")
class(x); dim(x)

x.m <- tiff_to_mtx(data = x, columns = 64)
class(x.m); dim(x.m)
```

`V2` *Line-scan data of HEK-293 cells expressing Venus (EYFP) dimers*

Description

This data set consists on a raster line scan performed over HEK-293 cells expressing dimers of the fluorescent protein Venus, also known as SEYFP-F46L. The scan line is 64 pixels long, and the scanning direction is from the cytoplasm to the nucleus, across the nuclear envelope. A pixel size of 50 nm was used, as well as a pixel dwell time of 12.5 us and a line scan time of 1.925 ms. Fluorescence excitation was provided by a 488 nm laser at 0.1 Fluorescence intensity data was collected using the photon-counting mode in an Olympus FV1000 Upright BX61WI confocal microscope.

Usage

```
data(V2)
```

Format

A matrix with 64 rows and 25000 columns

Details

To analyze this data set with FCSlib, import by typing: readFileTiff("V2.tif")[,,1]

writeFileTiff *Write File Tiff*

Description

Create a TIFF file from a 3D-array.

Usage

```
writeFileTiff(img,  file.name, invert = TRUE, bits.per.sample = NULL)
```

Arguments

img	Either an image or a list of images. An image is a real matrix or array of three dimensions.
file.name	Either the name of the file or the name of a raw vector.
invert	If set to TRUE then the order of the data will be reversed. Default TRUE.
bits.per.sample	Number of bits per sample (numeric scalar). Supported values in this version are 8, 16, and 32.

Details

Create a TIFF file using writeTIFF, converting a 2D-array. If the file contains multiple pages, a 3D-array is turned into a 2D-array to implement the aforementioned function.

Value

None.

Author(s)

Adan O. Guerrero Cardenas.

References

None

See Also

writeTIFF readFileTiff

Examples

```
imagsave <- array(data = 1:10, dim = c(100,100,10))
writeFileTiff(imagsave, paste(tempdir(), "/image_Test.tif", sep = ""))
```

Index