Package 'Rcell'

June 4, 2015

Type Package

Version 1.3-2

Date 2015-06-04

Title Microscopy Based Cytometry Data Analysis Package

Author Alan Bush <abush@fbmc.fcen.uba.ar>

Maintainer Alan Bush <abush@fbmc.fcen.uba.ar>

Depends R (>= 2.15.0), proto, plyr (>= 1.8), reshape, ggplot2 (>= 0.9.3), grid

Imports stats, digest

Suggests Hmisc, EBImage (>= 4.0.0), RcellData (>= 1.3.0)

Description Analyze microscopy based cytometry datasets created with 'Cell-ID'. It includes functions for manipulating and plotting the data. It can also create automatic image montages of cells in a user defined layout.

License GPL-2

Encoding latin1

URL http://www.embnet.qb.fcen.uba.ar, http://sourceforge.net/projects/cell-id

LazyData no

R topics documented:

`Rcell-package` *Microscopy Based Cytometry Data Analysis Package*

Description

Microscopy based cytometry can produce huge amount of images to be analized. Several programs can segment the acquired images and create a dataset with the features of the segmented cells. This package contains functions design to analyze such datasets. It was created to analyze data from Cell-ID (`http://sourceforge.net/projects/cell-id/`), but can be extended to analyze datsets created by other segmentation programs.

Details

Package:	Rcell
Type:	Package
Version:	1.3-2
Date:	2015-06-04
License:	GPL-2

Tutorials

For a introduction read the 'Getting Started with Rcell' vignette
`vignette(Rcell)`

To learn how to create complex plots read
`vignette(cplot)`

To see how to create layouts of cell's images read
`vignette(cimage)`

To lear how to normalize and manipulate your data read
`vignette(transform)`

To read a description of Cell-ID's variables and features
`vignette(Cell-ID-vars)`

Loading Cell-ID Data in R

Once you have processed the images with Cell-ID you will have to analyse the output dataset. The first thing you will have to do is load your data into R.

`load.cellID.data`: this function searches a specified directory (the working directory by default) for folders that match a customizable pattern, usually PositionXX where XX is the position number. The function loads these files and generates a suitable data structure. It returns a object of class cell.data that contains all the required information for the analysis. All the functions included in the package operate over this object, and its components should not be modified directly, but through the provided functions.

Quality Control and Filtering Cells

The algorithm used by Cell-ID to find the cells can occasionally make mistakes in the assignation of the cell boundaries and produce badly found and spurious cells (i.e. image structures erroneously scored as cells). Further more, the program does not discriminate out of focus and dead cells.

Normally you will want to get rid of all the spurious, badly found, out of focus and dead cells (referred collectively as 'bad' cells), which would constitutes a 'quality control' of the data. The R package contains some functions to aid in this process.

`QC.filter`: applies quality control filters over the data. The purpose of this function is to eliminate from the dataset 'bad' cells. You should not use this function to differentiate sub-groups of 'good' cells. This function treats a cell in different time points independently (i.e. it operates on registers of the dataset). To eliminate cells that are not scored in all the time frames, call `update.n.tot` and then filter by n.tot. Filters can be undone by `QC.undo`, or reseted by `QC.reset`. Use `summary.cell.data` to see a summary of the applied filters.

Plotting the Data

For plotting the data you can use the package plotting functions `cplot` and `cplotmeans`, which are wrappers over the **ggplot2** package functions.

Image Layout

To create image layouts (or montages) of your cells use the `cimage` functions.

Data Manipulation

Some common manipulations you can apply over a cell.data object are subsetting (`subset.cell.data`) which returns a cell.data object, extraction (`[[.cell.data`) and aggregation (`aggregate.cell.data`) which return a data.frame. You can also create new variables form existing ones. This can be done to save typing, or to normalize your data. To this end you can use the `transform.cell.data` and `transform.by.cell.data` functions.

Author(s)

Alan Bush Maintainer: Alan Bush <abush@fbmc.fcen.uba.ar>

References

http://sourceforge.net/projects/cell-id

See Also

EBImage ggplot2

---

aggregate *Compute Summary Statistics of Cell Data Subsets*

Description

Splits the data into subsets, computes summary statistics for each, and returns the result in a data frame

Usage

```
## S3 method for class cell.data
aggregate(x, form.by, ..., FUN=mean, subset=TRUE, select=NULL
  ,exclude=NULL, QC.filter=TRUE)

aggregateBy(x,.by,...)

## S3 method for class cell.data
aggregateBy(x, .by, select, ..., FUN=mean, subset=TRUE, exclude=NULL, QC.filter=TRUE)

## S3 method for class data.frame
aggregateBy(x,.by,select="all",...,FUN=mean,subset=NULL,exclude=NULL)

## Default S3 method:
aggregateBy(x,.by,select="all",...,FUN=mean,subset=NULL,exclude=NULL)
```

Arguments

x	cell.data object
form.by	either a formula or variables to split data frame by, as quoted variables or character vector
.by	variables to split data frame by, as quoted variables or character vector
...	further arguments passed to or used by methods
FUN	a function to compute the summary statistics which can be applied to all data subsets
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the datasets variable, that specifies which registers should be included

select	character vector defining variables names to be included in the returned data.frame
exclude	character vector defining variables names to be excluded from the returned data.frame
QC.filter	a boolean value indicating if the quality control filter should be applied over the data

Details

aggregate is a generic function. This version applies to cell.data objects. Two notations are allowed. If the second argument form.by is a formula it should be of the form cbind(measure.var1,measure.var2)~gro
If the second argument form.by are quoted variables or a character vector with variable names, these variables are taken as group.vars to split the dataset. The measure variables over which to apply FUN should be selected using the select and exclude arguments.

aggregateBy works as aggregate, but forces the use of quoted variables (or variable names) to define the groups by which the dataset is going to be split. This function also has a implementation for data frames. aggregateBy calls flatten before returning the data frame.

Value

a data frame with columns corresponding to the grouping variables followed by aggregated columns of the measure variables.

Author(s)

Alan Bush

See Also

aggregate

Examples

```
if(require(RcellData)){

  #load example dataset
   data(ACL394filtered)

  #aggregate by pos and calculate mean f.tot.y
  aggregate(X,.(pos),select="f.tot.y")

  #do the same aggregation using the formula notation
  aggregate(X,f.tot.y~pos)

  #aggregate by pos and t.frame
  aggregate(X,.(pos,t.frame),select="f.tot.y")
  aggregate(X,f.tot.y~pos+t.frame) #formula notation

  #aggregate several variables
  aggregate(X,.(pos),select="f.tot.?") # using wildcard pattern matching
  aggregate(X,cbind(f.tot.y,f.tot.c)~pos) #formula notation

  #subset before aggregating
  aggregate(X,.(pos),select="f.tot.y",subset=t.frame==13)

  #calculate the median instead of the mean
  aggregate(X,.(pos),select="f.tot.y",FUN=median)
```

```
  #dont apply the QC filter to the daset before aggregation
  aggregate(X,.(pos),select="f.tot.y",QC.filter=FALSE)

  #use aggregateBy on a cell.data object
  (agg<-aggregateBy(X,.(pos,AF.nM,t.frame),select="f.tot.y"))

  #use aggregateBy on a data.frame, calculate mean and sd among position means
  aggregateBy(agg,.(AF.nM,t.frame),select="f.tot.y",FUN=funstofun(mean,sd))

}
```

`as.cell.data` *Coerce to Cell Data*

Description

Coerces a list or data.frame to a cell.data object

Usage

```
as.cell.data(X,...)

## S3 method for class list
as.cell.data(X,path.images=NULL,...)

is.cell.data(X)
```

Arguments

`X`	list to be coerced to (or test for) cell.data object
`path.images`	string containing path to the image files
`...`	additional arguments to be passed to or from methods

Details

`as.cell.data` coerces objects to class cell.data. If a list is coerced, it is expected to have components 'data', 'bf.fl.mapping' and others. It is specially usefull to coerce data loaded with previous versions of Rcell. `is.cell.data` test if a object inherits from class cell.data

`path` is used to update the path of the image files.

Value

a cell.data object

Author(s)

Alan Bush

See Also

load.cellID.data

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #transforming dataset to list
  Xlist<-as.list(X);class(Xlist)<-"list";

  #re-coerce to cell.data
  Y<-as.cell.data(Xlist)
}
```

---

as.data.frame *Coerce to a Data Frame*

Description

Function for extracting a (subset) data.frame from a cell.data object

Usage

```
## S3 method for class cell.data
as.data.frame(x,row.names=NULL,optional=FALSE,...,subset=TRUE
  ,select=NULL,exclude=NULL,QC.filter=TRUE,na.rm=FALSE)

 ## S3 method for class cell.data
x[[subset=TRUE,select=NULL,exclude=NULL,QC.filter=TRUE,na.rm=TRUE,...]]

 cdata(x,subset=TRUE,select=NULL,exclude=NULL,QC.filter=TRUE,na.rm=TRUE,...)
```

Arguments

x	cell.data object
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset?s variable, which specifies which registers should be included in the returned data.frame
select	character vector defining variables names to be included in the returned data.frame
exclude	character vector defining variables names to be excluded from the returned data.frame
QC.filter	a boolean value indicating if the quality control filter should be applied over the data
na.rm	boolean indicating if registers with NA should be removed from the data.frame
...	further arguments passed to or used by methods
row.names	further arguments passed to or used by methods
optional	further arguments passed to or used by methods

Details

as.data.frame.cell.data coerces a cell.data object to a data.frame, subsetting it as defined by the other arguments. This function will be called when the generic function `as.data.frame` is applied over a cell.data object.

The extract ('[[') operator is an alias to this function.

`select` and `exclude` can be used to choose which variables should be included in the returned data.frame. Wildcard patterns (e.g. 'f.*.y') and keywords (e.g. 'all', 'id.vars', 'YFP', etc.) can be used as components of these arguments. Use `summary.cell.data` to see available variables and keywords. Variable names starting with '-' in `select` will be excluded from the data.frame.

Value

A data.frame, subset as specified by the functions arguments.

Author(s)

Alan Bush

See Also

`as.data.frame`

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #extract the dataset to a data.frame
  df<-as.data.frame(X)
  df<-X[[]]

  #extract a subset of the data.frame
  df<-X[[t.frame==13,]]

  #extract a selected group of variables
  df<-X[[,c("id.vars","f.tot.?","a.tot")]]
  #note the use of keywords, patterns and variable names

  #extract the dataset without applying the QC filter
  df<-cdata(X,QC.filter=FALSE)
}
```

`cell.data` *Cell Data Object*

Description

cell.data object description

Details

cell.data objects created by `load.cellID.data` and manipulated by the functions of the package. They are list with class 'cell.data' and contain the following elements

data main data.frame containing all the variables created by Cell-ID, plus additional variables created in R. To see a full description of Cell-ID's variables read the 'Cell-ID-vars' vignette (`vignette(Cell-ID-vars)`). It also contains the special QC variable, that contains the Quality Control filter created by `QC.filter`.

QC.history list containing the description of the different filters applied with `QC.filter`

subset.history list containing the description of the different subsets applied with `subset.cell.data`

transform list containing the description of the variables created with `transform.cell.data` or `transformBy.cell.data`

channels data.frame containing the names and posfix of the available fuorescence channels

variables list containing all the available variable names. The names of the items of the list work as a keyword. Each item contains a character vector with variable names (elements of data). Example of keywords ($variables elements) are 'id.vars', 'morpho', 'fluor', 'all', 'transformed', 'YFP', etc.

images data.frame containing information regarding the images run by Cell-ID.

software character describing the segmentation software used

load.date character containing the date in which the dataset was loaded to R.

Author(s)

Alan Bush

`cell.image` *Get Cells Images*

Description

Retrieves the images from single cells in an cell.image object

Usage

```
get.cell.image(X,...)

## S3 method for class cell.data
get.cell.image(X,subset=NULL,channel.subset=NULL,channel=NULL
,time.course=TRUE,group=NULL,na.rm=TRUE,N=7,select=NULL,exclude=NULL
,QC.filter=TRUE,box.size=20,...)

## S3 method for class data.frame
get.cell.image(X,box.size=20,contained.box=FALSE,bg.col=0,...)

## Default S3 method:
get.cell.image(X,box.size=20,...)

## S3 method for class cell.image
summary(object,...)
```

```
## S3 method for class summary.cell.image
print(x,...)

## S3 method for class cell.image
print(x,nx=ceiling(sqrt(length(x))),...)

img.desc(X)

img.desc(X)<- value

is.cell.image(X)
```

Arguments

X	cell.data object or data.frame that specifies the images
subset	logical expression indicating elements or rows to keep. Don't specify channel here.
channel.subset	logical expression to specify which image to retrieve with channel and t.frame variables.
channel	character vector of channels to retrieve. If specified, defines the order of the channels.
time.course	boolean indicating if the desired image montage is a time course (i.e. several images for the same cell)
group	character vector or quoted names of variables who's interaction define the groups from which select N random cells.
na.rm	boolean indicating if NAs should be removed.
N	Number of random cells to select from each group. If NULL all cells are selected
select	character vector defining variables names to be included in the returned cell.image object
exclude	character vector defining variables names to be excluded from the returned cell.image object
QC.filter	a boolean value indicating if the quality control filter should be applied over the data
box.size	size in pixels of the image containing the cells. This specifies the 'radius', i.e. the image will be a square of length 2*box.size+1
...	further arguments for methods
contained.box	boolean indicating if the XY position of the box should be corrected to be contained in the original image. Relevant for cells close to the image border. If FALSE the part of the box outside the original image will be filled with bg.col
bg.col	color to be used for the background of the images
object	cell.image object to summarize
x	object to print
nx	number of columns in the image tile
value	a data.frame to use as image description database

Details

`get.cell.image` is a generic method that returns a cell.image object.

If `get.cell.image` first argument is a data.frame, it should contain the columns path, image, xpos and ypos.

If the first argument when calling `get.cell.image` is a cell.data object, further arguments specify which images will be selected. The `subset` arguments filters the dataset as in other functions. If some variables are specified in `group`, the data is split in groups defined by these variables, and from each group `N` cells are selected at random. The `channel` argument specifies which channels to show. If a more complex image selection is required, you can use the `channel.subset` argument. For example if you want to see the BF only for the first t.frame, and then only the YFP channel, you can use `channel.subset=channel==YFP|(t.frame==0&channel==BF)`

`img.desc` returns a data.frame describing each image of the cell.image object

Value

a cell.image object. This object is basically a list who's elements are the cropped images of single cells. It has a attribute named 'img.desc' that is a data.frame with the image index (img.index) and description of all the components of the objects.

Author(s)

Alan Bush

See Also

EBImage

Examples

```
if(interactive() & require(EBImage,quietly=TRUE) & require(RcellData)){

  #load example dataset
  data(ACL394filtered)

  #select N=3 cells images from each pos (group),
  #from the first t.frame and pos 1,8,15,22,29.
  ci<-get.cell.image(X,subset=match(pos,c(1,8,15,22,29),nomatch=0)>0&t.frame==11,
    group=.(pos),N=3,channel=c(BF.out,YFP))
  print(ci) #print the cells images
  summary(ci) #get a summary of the content
  img.desc(ci) #get the image description data.frame

  #select the first 4 t.frames for YFP, and the first t.frame for BF
  ci<-get.cell.image(X,subset=pos==29,group=pos,
    channel.subset=channel==YFP|(t.frame==11&channel==BF))
  print(ci)

}
```

cimage *Images Layout*

Description

Arranges cell's images in a plot

Usage

```
cimage(X,...)

## S3 method for class cell.data
cimage(X,formula=NULL,facets=NULL,QC.filter=TRUE
,time.var=c("*time*","t.frame","z.scan","z.slice"),time.course=NULL
,select=NULL,exclude=NULL,normalize.group="channel",invert.lut=FALSE
,N=NULL,...)

## S3 method for class cell.image
cimage(X,formula=NULL,subset=NULL,facets=NULL
,scales="fixed",allow.expressions=FALSE
,nx=NULL,ny=NULL,facets.nx=NULL,facets.ny=NULL
,bg.col="white",border=1,facets.border=1,rev.y=TRUE
,font.size=14,font.col="black",display=interactive(),...)

## Default S3 method:
cimage(X,...)
```

Arguments

X	cell.data or cell.image object to plot
formula	formula of the form 'var1+var2~var3' specifying how the images are to be ordered. See details.
facets	formula of the form 'var1+var2~var3' specifying how to facet the plot. See details.
time.var	variables that indicate time and should be excluded from the grouping variables. See get.cell.image
time.course	boolean indicating if the image layout represents a time course and several images of the same cell at different times are expected
select	character vector defining further variables that are required for the plot
exclude	character vector defining variable names to be excluded
normalize.group	variable names that define groups of images that should be normalized together
scales	either 'none', 'fixed' or 'free' axis for each facet
allow.expressions	allow expressions in formulas, set to TRUE when called from cimage.cell.data
nx	number of columns of images within each facet. Used with formula '~var1' or 'var1~.'

ny	number of rows of images within each facet. Used with formulas '~var1' or 'var1~.'
facets.nx	number of columns of facets. Used with facets '~var1' or 'var1~.'
facets.ny	number of rows of facets. Used with facets '~var1' or 'var1~.'
bg.col	The background color of the plot
border	the width in pixels of the border between images
facets.border	the width in pixels of the border between facets
rev.y	boolean indicating if the y axis should be reversed
font.size	The size of the font to use, in pixels
font.col	The color of the font to use
display	boolean indicating if the created image should be displayed
QC.filter	a boolean value indicating if the quality control filter should be applied over the data
invert.lut	boolean indicating if Look Up Table should be inverted
N	Number of random cells to select from each group. If NA or 'all', all cells are selected.
subset	logical expression indicating elements or rows to keep. Don't specify channel here
...	further arguments for methods. cimage calls get.cell.image, so all the arguments of this function are available.

Details

channel.subset	logical expression to specify which image to retrieve with channel and t.frame variables
channel	character vector of channels to retrieve. If specified, defines the order of the channels
box.size	size in pixels of the image containing the cells. This specifies the 'radius', i.e. the image will be a squa
contained.box	boolean indicating if the XY position of the box should be corrected to be contained in the original im
bg.col	color to be used for the background of the images

Read the cimage vignette for a tutorial on how to use this function: vignette('cimage')

cimage is a generic method that returns a 'Image' object, from EBImage package.

If cimage's first argument is a cell.data object, it first calls get.cell.image and then the cimage method for cell.image objects. This function arranges the images of single cells according to the formula and facets arguments, and adds appropriated axis to the image.

For example, formula=channel~t.frame, will arrange different channels as rows and t.frame as columns. You can use several variables per term, for example formula=channel~pos+t.frame will arrange the columns first by position, and within each position by t.frame. The variable to the right varies faster than the one to the left. If only the right term of the formula is defined, as in formula=~t.frame, the images are 'wrapped' around, attempting to create a square plot. nx and ny can be used to define the number of columns or rows respectively. The special keyword 'cell' can be used to indicate the samples within a group, for example formula=cell~t.frame. The facets argument works in a similar way.

Value

The function returns an invisible 'Image' object of the EBImage package. Use display to render the image or writeImage to save it. You can also use `plot` to print to the active device and `img.desc` to retrieve the description of each cell.

Author(s)

Alan Bush

See Also

EBImage,display

Examples

```
if(interactive()&require(EBImage,quietly=TRUE)&require(RcellData)){

#load example dataset
data(ACL394filtered)

#display timecourse strip of cell 5 of pos 29, channels BF and YFP
cimage(X,channel~t.frame,subset=pos==29&cellID==5,channel=c(BF,YFP))

#display 7 cells (default value for N) of pos 29
cimage(X,cell+channel~t.frame,subset=pos==29,channel=c(BF,YFP))

#display 3 cells from each pos in a different facet
cimage(X,channel~cell,facets=~pos,channel=c(BF.out,YFP),N=3,
subset=t.frame==11&match(pos,c(1,8,15,22,29),nomatch=0)>0)

#select one BF and many YFP images
cimage(X,cell~channel+t.frame,subset=pos==29,N=3,
channel.subset=channel==YFP|(channel==BF.out&t.frame==11))

#make a movie!
cimage(X,.~cell|t.frame,subset=pos==29,channel=YFP,N=9)

}
```

`conform` *Conform a Data Frame*

Description

conforms the structure of a data.frame to that of an other

Usage

```
conform(df,to)
```

Arguments

df	data.frame to be conformed
to	data.frame to use as template (columns order)

Details

this function is useful do rbind between data frames that have different columns, or columns in diffent order.

Value

a data frame conformed to the template

Author(s)

Alan Bush

Examples

```
#creating example data frames
df1<-data.frame(a=1:4,b=5:8)
df2<-data.frame(b=9:14)
df3<-data.frame(b=9:14,a=20:25)

#using conform
conform(df2,to=df1)
conform(df3,to=df1)

#using conform with rbind
rbind(df1,conform(df2,to=df1))
rbind(df1,conform(df3,to=df1))
```

---

cplot *Plotting Cell Data Objects*

Description

Plotting functions for cell.data objects. These functions are wrappers over the functions of ggplot2 package.

Usage

```
cplot(X=NULL, x=NULL, subset = NULL, y=NULL, z=NULL, ...
  ,facets = NULL, margins=FALSE, geom = "auto"
  ,stat=list(NULL), position=list(NULL), log = "", as.factor="as.factor"
  ,xlim = c(NA, NA), ylim = c(NA, NA), xzoom = c(NA,NA), yzoom = c(NA,NA)
  ,xlab = deparse(substitute(x)), ylab = deparse(substitute(y)), asp = NA
  ,select = NULL, exclude = NULL, na.rm = TRUE, QC.filter = TRUE
  ,main = NULL, add = FALSE, layer = FALSE)
```

```
clayer(...,geom="auto")

cplotmeans(...,geom=c("point","errorbar","line"))

clayermeans(...,geom=c("point","errorbar","line"))

cplotmedian(...,geom=c("point","errorbar","line"))

clayermedian(...,geom=c("point","errorbar","line"))

## S3 method for class cell.data
plot(x,y,...)
```

Arguments

X	cell.data object
x	either a variable symbol or expression, or a formula of the form y~x or ~x
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset?s variable, which specifies which registers should be included in the plot
y	a variable symbol or expression to be plot in the y axis. Ignored if x is a formula. A vector of symbols is allowed
z	a variable symbol specifying the "z" aesthetic mapping
...	other arguments passed on to the geom functions
facets	faceting formula to use
margins	whether or not margins will be displayed
geom	geom to use (can be a vector of multiple names)
stat	statistic to use (can be a vector of multiple names)
position	position adjustment to use (can be a vector of multiple names)
log	which variables to log transform ("x", "y", or "xy")
as.factor	variable names (wildcard pattern or keyword) to be treated as factors
xlim	limits for x axis c(min,max) (filters the x variable BEFORE applying the stat transformation)
ylim	limits for y axis c(min,max) (filters the y variable BEFORE applying the stat transformation)
xzoom	zoom range for x axis c(min,max) (resizes the plotting region AFTER the stat transformation)
yzoom	zoom range for y axis c(min,max) (resizes the plotting region AFTER the stat transformation)
xlab	character vector or expression for x axis label
ylab	character vector or expression for y axis label
asp	the y/x aspect ratio
select	character vector defining variables names to be included in the returned ggplot object, beside the ones required for the plot
exclude	character vector defining variables names to be excluded from the returned ggplot object

na.rm	boolean indicating if registers with NA should be removed from the data.frame
QC.filter	a boolean value indicating if the quality control filter should be applied over the data before plotting
main	character vector or expression for plot title
add	the plot is added as a layer to the last plot (returned by last_plot)
layer	boolean. If TRUE a layer is returned instead of a new ggplot object. Mutually exclusive with add

Details

Read the cplot vignette for a tutorial on how to use this function: vignette('cplot')

cplot is a wrapper over the functions of ggplot2 package from Hadley Wickham. It is based on qplot and keeps many of its arguments. The main differences between cplot and qplot are the following:

- cplot's first argument is a cell.data object (or a data.frame)
- the 'x' and 'y' aesthetic mapping can be specified by a formula in cplot
- a vector of variables can be specified for 'y' aesthetic mapping. This produces a data restructuring and sets the color aesthetic to variable
- variables selected by as.factors are coerced to factors before plotting
- the plotting region can be easily specified with xzoom and yzoom. Useful when stat='summary'.
- a subset of the dataset can be performed before plotting
- only the required variables for the plot are included in the ggplot object, thus reducing the memory space it requires. Additional variables can be included with the select and exclude arguments.
- if a logical QC variable is present in the dataset, it is used to filter it before plotting
- unused levels of factors can be drop with droplevels
- the specified plot can be returned as a layer to add to other plots with the '+' operator

clayer is a wrapper for cplot with layer=TRUE. This function returns a layer that can be added to other ggplot objects with the '+' operator.

cplotmeans (alias cplotmean) is a wrapper over cplot with stat='summary' and fun.data='mean_cl_normal'. This function plots the mean and confidence limits for the mean of the data, grouped by levels of the x variable. The default confidence interval is of 95%, and can be modified with the conf.int argument (passed to smean.cl.normal).

clayermeans (alias clayermean) is a wrapper over cplot with stat='summary', fun.data='mean_cl_normal' and layer=TRUE.

cplotmedian (and clayermedian) is a wrapper over cplot with stat='summary', fun.data='median_hilow' and layer=FALSE (TRUE).

plot.cell.data is a wrapper over cplot. It only accepts formula notation for the 'x' and 'y' aesthetics. It can be called by plot over a cell.data object.

Value

a ggplot object or a list specifying plots layers

Author(s)

Alan Bush

References

H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.

See Also

qplot,ggplot

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #plotting YFP vs CFP fluorescence
  cplot(X,f.tot.y~f.tot.c)

  #subset the data before plotting and color by pos variable
  cplot(X,f.tot.y~f.tot.c,subset=t.frame==13,color=pos)

  #map the size aesthetic to the the cell area a.tot
  cplot(X,f.tot.y~f.tot.c,subset=t.frame==13,color=pos,size=a.tot)

  #adding description of the positions for futher plotting
  # (AF.nM: dose of alpha-factor yeast pheromone in nM)
  X<-merge(X,data.frame(pos=1:35,AF.nM=rep(c(1.25,2.5,5,10,20),each=7)))

  #plot time course for f.tot.y and facet by pheromone dose
  cplot(X,f.tot.y~t.frame,facets=~AF.nM)

  #jittering the points to reduce overplotting
  cplot(X,f.tot.y~t.frame,facets=~AF.nM,size=0.5,geom="jitter")

  #adding per t.frame mean to prevoius plot
  cplot(X,f.tot.y~t.frame,facets=~AF.nM,size=0.5,geom="jitter")+
    clayermean(color="red")

  #plot means for each dose in the same plot
  cplotmean(X,f.tot.y~t.frame,color=AF.nM,as.factor="AF.nM",yzoom=c(0,6.2e6))

  #plotting histograms
  cplot(X,~f.tot.y)

  #use position dodge instead of stack
  cplot(X,~f.tot.y,fill=AF.nM,as.factor="AF.nM",position="dodge")
}
```

---

draw.img *Draw on a Image*

Description

funcionts for modifying EBImage images

Usage

```
drawCross(img, x, y, size=2, col=0.75, z=1)

drawLine(img, x1, y1, x2, y2, col=0.75, z = 1)

drawText(img,labels,x=NULL,y=NULL,adj=c(0,0),reuseLabels=TRUE,col=NULL)
```

Arguments

img	EBImage Image to modify
x	vector of x positions to draw
y	vector of y positions to draw
x1	vector of x1 positions to draw
y1	vector of y1 positions to draw
x2	vector of x2 positions to draw
y2	vector of y2 positions to draw
labels	character vector of labels
adj	one or two values in [0, 1] which specify the x (and optionally y) adjustment of the labels. On most devices values outside that interval will also work.
col	color of the object
size	size of the cross
reuseLabels	boolean indicating if labels created in previous calls should be reused
z	image layer in which to draw

Details

drawCross, drawLine and drawLabel draw on EBImage images, at the specified x y positions.

Value

a EBImage image

Author(s)

Alan Bush

See Also

cimage

Examples

```
if(require(EBImage,quietly=TRUE)&interactive()&require(RcellData)){

  data(ACL394filtered)
  img<-show.img(X,pos=1,channel="BF",cross=FALSE)
  p1<-X[[pos==1&t.frame==0,c("?pos","cellID")]]
```

```
  display(drawCross(img,p1$xpos,p1$ypos,col=0))
  display(drawText(img,p1$cellID,p1$xpos,p1$ypos,col=0))
  display(drawLine(img,p1$xpos[1],p1$ypos[1],p1$xpos[2],p1$ypos[2]))

}
```

`flatten` *Flatten a Data Frame*

Description

converts matrix elements of data frames into columns

Usage

```
  flatten(df,...)

  ## S3 method for class data.frame
flatten(df,...)
```

Arguments

`df`	data.frame to be flattened
`...`	further arguments passed to or used by methods

Details

this function is useful to flatten data frames obtained by aggregate when using smean.cl.normal and other functions from Hmisc.

Value

a data frame

Author(s)

Alan Bush

See Also

`with`

Examples

```
if(require(Hmisc)&require(RcellData)){
  #load example dataset
  data(ACL394)
  agg<-aggregate(X,f.tot.y~pos,subset=t.frame==0,FUN=smean.cl.normal)
str(agg)
agg<-flatten(agg)
str(agg)
}
```

ggplot2.themes *ggplot2 themes*

Description

Themes for ggplot2 graphics

Usage

```
theme_Rcell()
theme_invisible()
```

Details

I found these functions posted at https://github.com/hadley/ggplot2/wiki/Themes. I included them here for convenience.

These functions provide more themes for ggplot2 graphics. They work just as theme_grey and theme_bw

Value

A list with theme elements

Examples

```
#creating example datset
mdf <- data.frame(x <- seq(0, 10), y=rnorm(x),
                  f=factor(rep(letters[1:2], each=3, length=length(x))))
#base plot
p <- qplot(x, y, data=mdf, colour=f, geom=c("line", "point"))

#compare themes
p + theme_grey() + labs(title="theme_grey()")
p + theme_bw() + labs(title="theme_bw()")
p + theme_Rcell() + labs(title="theme_Rcell()")
p + theme_invisible() + labs(title="theme_invisible()")
```

load.cellID.data *Load Cell-ID Data*

Description

load.cellID.data searches a specified directory (the working directory by default) for folders that match a customizable pattern, usually PositionXX where XX is the position number. This folders should contain the Cell-ID output files output_all and the output_bf_fl_mapping for each position. The function loads this files and generates a data structure suitable for filtering and plotting. The function returns a cell.data object that contains all the required information for the analysis. All the functions included in the package operate over this object, and its components should not be modified directly, but through the provided functions. Remember to assign the returned value to a variable (e.g. X<-load.cellID.data())

Usage

```
load.cellID.data(pattern="^[Pp]{1}os[:alpha:]*[:digit:]*",path=getwd()
  ,basename="out",select=NULL,exclude=NULL,load.vars="all",split.image=FALSE)
```

Arguments

pattern	regular expression (see regexp) pattern of the position folders, where the images and cell ID output files for each position are stored.
path	character containing path from where to apply the pos.pattern to the existing folders. It should point to the folder that contains the PosXX folders.
basename	character containing basename of the cell ID output files, should match the -o option passed to cellID when executed. 'out' by default.
select	character vector defining variables names to be included in the cell.data object
exclude	character vector defining variables names to be excluded of the cell.data object
load.vars	character specifying which variables or group of variables of the Cell-ID out_all file should be loaded.
split.image	boolean indicating if the images are split and upper cells should be matched to lower cells. Set to TRUE if analyzing a FRET split image experiment.

Details

reads Cell ID output files (basename)_all in folders that match pattern in path and loads them into a cell.data object.

It searches for the output_all files in folders of the form specified by pattern (regular expression). If the folder has a numeric value in its name that number is taken as the position index (for example pos01 is given the index 1) If no numeric value is found in the folder name, then a ordinal index is assign.

Possible values for load.vars are 'all', 'fl' or 'fluorescence', 'bg' or 'background', 'calc', 'morph' or 'morphological', 'vac' or 'vacuole', 'nucl' or 'nuclear', 'disc'. The group of variables can be specified in either a positive form (i.e. '+fl+bg+morph') or in a negative form (i.e. '-nucl-vac'). Combination of positive and negative form is not allowed. A character vector containing the variables names of the out_all file is also allowed. The selection of variables is done before restructuring, so the variable names should correspond to those of the out_all files. Using this argument can be useful if memory issues arise.

Alternatively `select` and `exclude` can be used to subset the dataset. This arguments are applied after the reshaping, so variables names as returned by `summary.cell.data` are used. Wildcard patterns (e.g. 'f.*.y') and keywords (e.g. 'all', 'id.vars', 'YFP', etc.) can be used as components of these arguments.

Value

a cell.data object

Note

The restructuring of the data involves arranging the information for each time point of each cell into a single row. In the output of Cell-ID this information appears in several rows, one for each channel. The restructured data 'collapses' this rows into a single one, adding and modifying the column names by appending a channel specific postfix. If `split.image` is set to TRUE a sub-image indentifier is also appended, 'u' for upper and 'l' for lower. When Cell-ID is run, the images

it uses have to be named in a specific way. The first three letters of the image name are used as a channel token, i.e. it identifies the channel. If you have YFP and CFP channels, the images should be named YFP_Position1, YFP_Position2,...,CFP_Position1, CFP_Postion2,... The channel postfix is the shortest unambiguous substring of the channel token in lower case. For example for the tokens 'YFP' and 'CFP', the selected postfix will be 'y' and 'c' respectively.

Author(s)

Alan Bush

See Also

read.table,dir, QC.filter, summary.cell.data

Examples

```
## Not run:
setwd(".")  #set the working directory to the folder with your images
X<-load.cellID.data()  #load the dataset to R


## End(Not run)
```

---

merge *Merge a Data Frame to a Cell Data Object*

Description

Merges the variables in a data.frame to a cell.data object, using common variables to do the merging

Usage

```
## S3 method for class cell.data
merge(x,y,by=NULL,na.rm=FALSE,add=FALSE,warn=TRUE,pos.offset=NULL,...)

load.pdata(X,pdata="pdata.txt",by=NULL,path=getwd())
```

Arguments

X	cell.data object
x	cell.data object
y	a data.frame with at least one common variable with x
by	character vector indicating which variables to use for the merging
na.rm	should NAs be removed before merging
add	boolean indicating if new values should be added to previously merged ones
warn	boolean indicating if warnings should be issued
pos.offset	position offset used when merginf cell.data objects

pdata	either a string with the filename of a tab delimited text file containing the data to be merged, or a data.frame to merge
path	string containing the path to the location of the tab delimited file to be loaded
...	additional arguments to be passed to or from methods

Details

merge is used to add the variables in a data.frame to the cell.data object. It uses common variables to do the merging. The variables can be specified with the by argument.

load.pdata is a wrapper over merge, used to load position information to the cell.data object. By default it looks for a file named 'pdata.txt' in the working directory. This file should have a 'pos' column.

Value

a cell.data object with the merged variables.

Author(s)

Alan Bush

See Also

merge

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)
  #creating data frame with information about each poistion
  #AF.nM: dose of alpha-factor yeast pheromone in nM
  pdata<-data.frame(pos=1:35,AF.nM=rep(c(1.25,2.5,5,10,20),each=7))

  #merging the data frame with the cell.data object
  X<-merge(X,pdata)

}
```

---

plot.Image *Plot Image*

Description

Plots a EBImage Image to the active device

Usage

```
## S3 method for class Image
plot(x,width=NULL,height=NULL,omi=1,interpolate=FALSE,vp=NULL,...)
```

Arguments

x	EBImage of class Image, as returned by cimage
width	the width in inches of the device. If width or height are NULL, both are replaced by the dimensions of the active device
height	the height in inches of the device. If width or height are NULL, both are replaced by the dimensions of the active device
omi	number between 0 and 1. Defines the outter margins. If set to 0.95, 5% of the device in each side will be set as margin
interpolate	A logical value indicating whether to linearly interpolate the image
vp	A Grid viewport object (or NULL)
...	further arguments for grid.raster

Details

plot.Image is the S3 plot method for objects of class 'Image'.

Value

none

Author(s)

Alan Bush

See Also

plot

Examples

```
if(interactive()&require(EBImage,quietly=TRUE)&require(RcellData)){

  #load example dataset
  data(ACL394filtered)

  #timecourse strip of cell 5 of pos 29, channels BF and YFP
  img<-cimage(X,channel~t.frame,subset=pos==29&cellID==5,channel=c(BF,YFP),display=FALSE)
  plot(img)

}
```

QC.filter *Quality Control Filter*

Description

Create, undo, reset and execute quality control filters

Usage

```
QC.filter(X, filter, subset=NULL)

QC.undo(X)

QC.reset(X)

QC.execute(X)
```

Arguments

X	the cell.data object as returned by load.cellID.data make sure to save the object when it's returned by the function i.e. do the calls as X=QC.filter(X,...))
filter	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the datasetYens variable, that specifies which rows pass the quality control (TRUE), and which ones don't (FALSE).
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset variable, which specifies over which registers filter should be applied.

Details

QC.filter function filters the cells based on a user define boolean vector filter Such vector can be obtained applying logical operations over the vectors (fft.stat<0.2, etc). The purpose of this filter is to eliminate from your dataset spurious, badly found, out of focus and dead cells. This filter is cumulative, meaning that each time one applies a QC.filter function it adds to the previous QC.filter, it does not replaced them. Many functions from the package have a QC.filter argument, that specifies if the created QC filter should be applied to the dataset before the function is executed. The filter treats the same cells in different time points independently. Don't use this function to select subgroups of cells (see select.cells) summary.cell.data returns a description of the applied filters.

QC.undo removes the last filter applied. QC.reset eliminates all filters, restoring the cell.data object to its original state. QC.execute permanently eliminates the filtered registers. This is recommended only if you have memory issues.

Value

Returns the cell.data with the specified filter applied.

Note

Some times it is useful to create additional filters to discriminate between cells. Dont use QC.filter for this. You can create a filter with transform.cell.data and use the subset argument of the function you want to apply.

Author(s)

Alan Bush

See Also

summary.cell.data,transform.cell.data,load.cellID.data

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394filtered)

  #resetting all the filters
  X<-QC.reset(X)

  #filtering by fft.stat
  cplot(X,~fft.stat) #see what cut to use
  X<-QC.filter(X,fft.stat < 0.5) #apply the cut

  #filtering by the total number of frames in which a cell appears
  cplot(X,cellID~t.frame,fill=f.tot.y,geom="tile",facets=~pos)
  X<-update_n.tot(X) #updating n.tot variable
  cplot(X,~n.tot) #define where to apply the cut
  X<-QC.filter(X,n.tot==14) #keep cells that appear in all t.frames

  #exclude cells by ucid (Unique Cell ID)
  cplot(X,f.total.y~time.min,facets=~AF.nM,size=0.3,geom="jitter")
  #selecting cells that dont respond
  c1=select.cells(X,f.total.y<10e4&t.frame>3,n.tot.subset=n.tot>=8)
  X<-QC.filter(X,!ucid %in% c1)

  #undoing the last filter
  X<-QC.undo(X)

}
```

---

read.cell.image *Reads a Cell Image*

Description

Reads a cell image object from disk.

Usage

```
read.cell.image(file,...)
```

Arguments

file	filename or filename with path to the saved image
...	further arguments passed to readImage

Details

This function is a wrapper over `readImage`. It reads a image saved by `write.cell.image`, with the image descripcion database.

Author(s)

Alan Bush

See Also

`readImage`

Examples

```
## Not run:
  #load example dataset
  library(RcellData)
  data(ACL394filtered)

  ci<-get.cell.image(X,subset=match(pos,c(1,8,15,22,29),nomatch=0)>0&t.frame==11,
    group=.(pos),N=3,channel=c(BF.out,YFP))

  write.cell.image(ci,"Example-cell-image.tif")

  ci2<-read.cell.image("Example-cell-image.tif")


## End(Not run)
```

---

remove.vars *Remove Variables from a Cell Data Object*

Description

Returns a cell.data object, with the specified variables removed

Usage

```
remove.vars(X,select,exclude=NULL)
```

Arguments

`X`	cell.data object
`select`	character vector defining variables names to be removed in the returned cell.data
`exclude`	character vector defining variable names to be kept (not removed). This argument is somewhat counterintuitive (see details).

Details

It defines variables to be excluded from the selected ones to be removed.

`remove.vars` is used to eliminate variables one is not interested in. This significantly reduces the size of the cell.data object and therefore the size of the working environment when saved (to a .RData). It also reduced the chance of memory issues. In the call to `remove.vars` select defines which variables are to be removed. You can use wildchars. For example to remove all nuclear variables use select="*nucl*". The exclude argument defines variables to be excluded from the selected ones to be deleted. For example if you want to remove all nuclear vars, except f.nucl.y use select="*nucl*", exclude="f.nucl.y".

Value

a cell.data object with the specified variables removed

Author(s)

Alan Bush

See Also

`subset`, `summary.cell.data`

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #remove a variable
  X<-remove.vars(X,select="f.vacuole.y")

  #remove all background variables
  X<-remove.vars(X,select="*bg*")

  #remove all nuclear variables, except for f.nucl.y
  X<-remove.vars(X,select="*nucl*",exclude="f.nucl.y")

  summary(X)
}
```

---

reshape.cell.data *Reshape a Cell Data Object*

Description

Reshapes the data in a cell.data object and returns a data.frame

Usage

```
reshape(data,...)

## S3 method for class cell.data
reshape(data,formula = pos + cellID ~ variable + t.frame
  ,fun.aggregate=NULL, ..., margins=FALSE, fill=NULL
  ,id.vars=NULL, measure.vars=NULL, variable_name = "variable", na.rm = FALSE
  ,subset=TRUE ,select=NULL ,exclude=NULL ,QC.filter=TRUE)
```

Arguments

data	cell.data object
formula	casting formula, see details for specifics
fun.aggregate	aggregation function
...	further arguments are passed to aggregating function
margins	vector of variable names (can include 'grand_col' and 'grand_row') to compute margins for, or TRUE to computer all margins
fill	value with which to fill in structural missing, defaults to value from applying fun.aggregate to 0 length vector
id.vars	character vector of id variables names, wildcard pattern or keyword. If NULL, will use all variables of the formula.
measure.vars	character vector of measure variables names, wildcard pattern or keyword. If NULL, will use all non id.vars variables.
variable_name	Name of the variable that will store the names of the original variables
na.rm	Should NA values be removed from the data set?
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset¥s variable, that specifies which registers should be included
select	character vector defining variables names to be included in the returned data.frame
exclude	character vector defining variables names to be excluded from the returned data.frame
QC.filter	a boolean value indicating if the quality control filter should be applied over the data

Details

This function is a wrapper over melt and cast from the reshape package of Hadley Wickham.

The id variables are selected by default. You can use summary.cell.data to see which variables are used as defaults for id.vars. The measured variables can be specified with select and exclude, or with measure.vars.

The casting formula has the following format: x_variable + x_2 ~ y_variable + y_2 ~ z_variable ~ ... | li
The order of the variables makes a difference. The first varies slowest, and the last fastest. There are a couple of special variables: '...' represents all other variables not used in the formula and '.' represents no variable, so you can do formula=var1 ~ .

If the combination of variables you supply does not uniquely identify one row in the original data set, you will need to supply an aggregating function, fun.aggregate. This function should take a vector of numbers and return a summary statistic(s). It must return the same number of arguments regardless of the length of the input vector. If it returns multiple value you can use

result_variable to control where they appear. By default they will appear as the last column variable.

The margins argument should be passed a vector of variable names, eg. c('pos','t.frame'). It will silently drop any variables that can not be margined over. You can also use 'grand_col' and 'grand_row' to get grand row and column margins respectively.

Value

a reshaped data.frame

Author(s)

Alan Bush

See Also

aggregate

Examples

```
if(require(RcellData)){
  #load example dataset
  data(ACL394)

  #rehape position 1 in pos + cellID ~ variable + t.frame for f.tot.y variable
  reshape(X,select="f.tot.y",subset=pos==1)

  #redefining the formula, reshape against time in minutes
  X<-transform(X,time.min=10+t.frame*15) #calculating the time of each t.frame
  reshape(X,pos+cellID~variable+time.min,select="f.tot.y",subset=pos==1&t.frame<10)
}
```

---

revFactor *Reverse Factor Levels*

Description

Reverse the order of the levels of a factor

Usage

```
revFactor(x)
```

Arguments

x	a factor

Details

Useful to use in calls to cimage

Value

a ordered factor with the levels in the reverse order of levels(x).

Author(s)

Alan Bush

Examples

```
#create a factor
f<-factor(paste0("f",1:9))
levels(f)

#reverse the order of the levels
rf<-revFactor(f)
levels(rf)
```

---

select.cells *Select Subset of Cells*

Description

Selects a subset of cells that satisfy the specified conditions.

Usage

```
select.cells(X, subset = TRUE, n.tot.subset=NULL ,QC.filter=TRUE)
```

Arguments

X	cell.data object
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset¥s variable, that specifies which registers should be included
n.tot.subset	a conditional statement usually involving n.tot, to filter the cells by the total number of frames in which they appear.
QC.filter	a boolean value indicating if the QC.filter should be applied over the data

Details

select a group of cells be a criteria specified in `subset`. After the first subset is applied the number of frames in which a selected cell appears (n.tot) is calculated and an additional filter (`n.tot.subset`) is applied. This can be useful to select cells that satisfy the specified `subset` filter in all the time frames, or a fraction of them.

You can do union, intersection and difference of these sets.

Value

Returns a vector of the selected cells 'ucid'. The ucid (or 'unique cell id') is defined as pos*100000+cellID. Because the returned value is an integer vector all the set operations may be applied directly over subsets of cells selected by select.cells. The returned vector should be assigned to a variable for further usage.

Author(s)

Alan Bush

See Also

intersect,union,setdiff

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #select cells that have f.tot.y>1e7 in at least one t.frame
  c1<-select.cells(X,f.tot.y>1e7)
  cplot(X,f.tot.y~t.frame,color="gray",size=0.5) +  #plotting the cells
    clayer(X,f.tot.y~t.frame,color=ucid,geom="line",subset=ucid%in%c1)

  #select cells that have f.tot.y<6e5 in all t.frames
  c1<-select.cells(X,f.tot.y<6e5,n.tot.subset=n.tot==14)
  cplot(X,f.tot.y~t.frame,color="gray",size=0.5) +  #plotting the cells
    clayer(X,f.tot.y~t.frame,color=ucid,geom="line",subset=ucid%in%c1)
}
```

---

select.vars *Select Variables*

Description

Selects a group of variable names from the dataset.

Usage

```
select.vars(X,select="all",exclude=NULL)
```

Arguments

X	cell.data object
select	character vector defining variables names, keywords or wildcard patters to be included in the returned vector
exclude	character vector defining variables names, keywords or wildcard patters to be excluded from the returned vector

Details

Selects a group of variables. If you only use the first argument it returns 'all' the columns of the dataset.

Value

A character vector with variable names.

Author(s)

Alan Bush

See Also

names

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #select all variables
  select.vars(X)

  #select morphological variables
  select.vars(X,"morpho")

  #select variables of the YFP channel
  select.vars(X,"*.y")

  #select id vars, area vars and f.tot.y
  select.vars(X,c("id.vars","a.*","f.tot.y"))

  #select id vars, area vars and f.tot.y, exlude bg variables
  select.vars(X,c("id.vars","a.*","f.tot.y"),exclude="*bg*")

}
```

---

show.img *Show a Image*

Description

Display one or several BF or fluorescent images, indicating the cells eliminated by the QC filter.

Usage

```
show.img(X,pos,t.frame=0,channel="BF.out",image.title=""
  ,annotate=NULL,cross=!QC,QC.filter=FALSE,subset=TRUE,cross.col=c(0.1,0.9)
  ,display=interactive(),normalize=TRUE,...)

show.image(X,pos,t.frame=0,channel="BF.out",image.title=""
  ,annotate=NULL,cross=!QC,QC.filter=FALSE,subset=TRUE,cross.col=c(0.1,0.9)
  ,display=interactive(),normalize=TRUE,...)
```

Arguments

X	cell.data object as returned by load.cellID.data
pos	The position(s) of interest, from which the image will be shown.
t.frame	The time frame(s) of interest, from which the image will be shown. If it is a vector shorter than pos, it is recycled.
channel	the fluorescent channel label of interest. Usual values are 'BF', 'BF.out', 'YFP', 'YFP.out', etc. If it is shorter than pos or t.frame it is recycled.
image.title	optional title for the image
annotate	character vector with variable names with which to annotate the image. A usual value is 'cellID'. NOT IMPLEMENTED CURRENTLY!
cross	conditional statement with X variables indicating over which cells should a cross be placed.
QC.filter	boolean value, indicating if X should be subset by QC.filter before cross or annotate are applied. Default to FALSE.
subset	conditional statement using X variables used to subset X before other arguments are applied.
cross.col	vector of colors (gray levels) to be used for the crosses. Each mark is composed of two cross with the specified colors, moved one pixel from each other. Using a black and white enhances contrast.
display	boolean indicating if the created image should be displayed
normalize	boolean indicating if the images should be normalized to enhance contrast
...	further arguments

Details

Displays the bright field and/or fluorescence images. Cells can be annotated with a cross or the value of a selected variable. This function can be used as a feedback to verify that the cuts used for QC.filter were adequate.

Value

It returns a invisible EBImage image.

Note

This function requires EBImage package installed which, in turn, requires the ImageMagick software.

Author(s)

Alan Bush

See Also

EBImage

Examples

```
if(interactive()&require(EBImage,quietly=TRUE)&require(RcellData)){

  #load example dataset
  data(ACL394filtered)

  #display the BF out image from position 8 and t.frame 11
  show.img(X,pos=8,t.frame=11)

  #display a stack of the YFP images of position 29
  show.img(X,pos=29,t.frame=7:11,channel="YFP")
}
```

`subset` *Subset a Cell Data Objects*

Description

Returns subset of the cell.data object which meet conditions

Usage

```
## S3 method for class cell.data
subset(x,subset=TRUE,select="all",exclude=NULL,QC.filter=FALSE,...)
```

Arguments

`x`	cell.data object
`subset`	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset's variable, that specifies which registers should be included
`select`	character vector defining variables names to be included in the returned cell.data
`exclude`	character vector defining variables names to be excluded from the returned cell.data
`QC.filter`	a boolean value indicating if the quality control filter should be applied over the data before creating the new cell.data object
`...`	further arguments passed to or used by methods

Details

`subset` is a generic function. This version applies to cell.data objects. `subset` is a close function, meaning it returns an object of the same class as its first argument, in this case a cell.data object. Subsetting is useful to divide a large experiment into smaller dataset that are more easily analyzed. It can also be used to reduce the memory space a cell.data object occupies, for example eliminating the QC filtered registers (`X<-subset(X,QC.filter=TRUE)`) or eliminating unused variables (`X<-subset(X,exclude=c("morpho","f.bg.y","f.*.c"))`)

The bracket (`Extract`) notation can also be used `Y<-X[pos==1]`

`remove.vars` is a wrapper over subset, it eliminates the specified variables.

A record of the subset history of the object is kept. Use `summary.cell.data` to see it.

Value

a subset cell.data object

Author(s)

Alan Bush

See Also

subset, summary.cell.data

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #subset the cell.data by pos
  X1<-subset(X,pos==1)
  X1<-X[pos==1]

  #subset by t.frame and select variables
  #note the use of keywords and pattern matching to select the variables
  X.t13<-X[t.frame==13,c("morpho","*.y","f.tot.c")]
  summary(X.t13) #take a look at the new cell.data object

  #eliminate registers that didnt pass the QC filter
  X<-subset(X,QC.filter=TRUE)

}
```

---

summary *Cell Data Object Summary*

Description

Returns a summary of the cell.data object content.

Usage

```
## S3 method for class cell.data
summary(object,...)
```

Arguments

object	cell.data object
...	further arguments passed to or used by methods

Details

Returns a description of the cell.data object, including from where and when it was loaded, the number of positions and time frames and information about the default, transformed and merged variables. It also returns a history of the QC filters and subsets applied.

The function returns a list of class summary.cell.data that is printed by print.summary.cell.data.

Value

a list of class summary.cell.data

Author(s)

Alan Bush

See Also

summary

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #see the object summary
  summary(X)

  #assign the object summary
  X.sum<-summary(X)
  names(X.sum)

}
```

---

transform *Transform a Cell Data Object*

Description

Transforms a cell.data object adding new variables

Usage

```
## S3 method for class cell.data
transform(_data,...,QC.filter=TRUE)

transformBy(_data,.by,...)

## S3 method for class cell.data
transformBy(_data,.by,...,QC.filter=TRUE)
```

```
## S3 method for class data.frame
transformBy(_data,.by,...,subset=NULL)

## Default S3 method:
transformBy(_data,.by,...,subset=NULL)
```

Arguments

_data	cell.data object or data.frame to transform
.by	variables to split data frame by, as quoted variable
...	new variable definition in the form tag=value
QC.filter	a boolean value indicating if the quality control filter should be applied over the data
subset	logical expression indicating elements or rows to keep: missing values are taken as false. Only valid for data.frames, not for cell.data

Details

NOTE: transform.by had to be removed from the package. Use transformBy instead.

Read the transform vignette for a tutorial on the use of these functions
> vignette("transform")

transform.cell.data is the implementation of the generic function transform to cell.data objects. It creates the new variables based on the ... argument; a tagged vector expressions, which are evaluated in the dataset.

transformBy is a generic function. Before transforming the dataset, the function splits it by the variables specified in the .by argument. This argument should be a quoted list of variables, that can be easily created with the quoted function, for example .(pos,t.frame). This can be useful to do group-wise normalizations.

The transformed variables are summarized in the output of summary.cell.data.

Value

for transform(By).cell.data a transformed cell.data object

for transformBy.data.frame a transformed data.frame

Author(s)

Alan Bush

See Also

transform

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394filtered)
```

```
  #creating a new variable
  X<-transform(X,f.total.y=f.tot.y-a.tot*f.local.bg.y)

  #create a new variable normalizing by position
  X<-transformBy(X,.(pos),norm.f.total.y=f.total.y/mean(f.total.y))

  #create a new delta variable in sigle cells
  X<-transformBy(X,.(pos,cellID),delta.f.total.y=f.total.y-f.total.y[t.frame==0])

  #transformBy can also be used on a data.frame
  df<-aggregate(X,f.total.y~t.frame+pos) #creates a aggregate data.frame from X
  df<-transformBy(df,.(pos),delta.f.total.y=f.total.y-f.total.y[t.frame==0])

}
```

---

transform.cell.image.rd

Transform Cell Image

Description

funcionts that transforms a cell image object before plotting

Usage

```
cnormalize(X=NULL,normalize.group=c("channel"),ft=c(0,1),...)

ciciply(X=NULL,group=c("pos","cellID","channel"),FUN=sum,MARGIN=c(1,2),warn=TRUE)

add.nucleus.boundary(X=NULL,radii=c(2,3,4,5,6,7),pos.nucl.channel="YFP",col=0.75,...)

add.maj.min.axis(X=NULL,col=0.75,angle.var=NA,...)
```

Arguments

X	cell.image object to transform
normalize.group	character vector indicating which variables should be used to group the images for normalization
ft	A numeric vector of 2 values, target minimum and maximum intensity values after normalization.
group	character vector indicating which variables should be used to group the images before applying FUN
FUN	function to apply to the grouped imaged matrix
MARGIN	a vector giving the subscripts which the function will be applied over. 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns.
warn	boolean indicating if warnings should be issued.
radii	radii of the concentric circles to be plot around the nucleus found position. The defaults correspond to Cell-ID default values

`pos.nucl.channel`
: string indicating channel from which the nucleus coordinates should be extracted

`col`
: color to use for the nucleus boundary

`angle.var`
: string indicating variable that measures the angle between the major axis and a horizontal line (not calculated by Cell-ID)

`...`
: further arguments for methods

Details

All these functions take a cell.image object as their first argument, and return a modified cell.image object. In combination with `get.cell.image` and `cimage.cell.image` they can be used to do custom manipulation to the cell's images.

`cnormalize` is called from `cimage` to normalize the images before plotting. It normalizes the images to enhance contrast. The normalization groups (defined by `normalize.group`) are applied the same normalization, so the intensities can be compared within a group.

`ciciply` is inspired on the plyr package. It divides the cell.image object into groups defined by the `group` argument, combines the images within a group in a stack (or array) and applyies the `FUN` function, over the defined margins. For example if FUN=sum and MARGIN=c(1,2), several images are add up together. This can be used to create Z-projections.

`add.nucleus.boundary` and `add.maj.min.axis` overlay the nucleous boundary and the major and minor axis respectively on the cell's images.

if `X` is NULL, the funcion returns a character indicating with variables of the dataset it requires.

Value

The transformed cell.image object

Author(s)

Alan Bush

See Also

`cimage`

Examples

```
#suggested package EBImage required for these functions
if(require(EBImage,quietly=TRUE)&require(RcellData)){

  #load example dataset
  data(ACL394)

  #select N=3 cells images from each pos (group),
  #from the first t.frame and pos 1,8,15,22,29.
  ci<-get.cell.image(X,subset=match(pos,c(1,8,15,22,29),nomatch=0)>0&t.frame==11,
   group=.(pos),N=3,channel=c(BF,YFP))

  #display a cell image without normalization
  if(interactive()) display(tile(combine(ci)))
```

```
  ci<-cnormalize(ci) #apply normalization
  if(interactive()) display(tile(combine(ci))) #display again
}
```

---

update_img.path *Update Image Path*

Description

Updates the path to the images folder. Useful if the images are in a different location from the one they were run by Cell-ID.

Usage

```
update_img.path(X,img.path=getwd(),subset=NULL)
```

Arguments

X	cell.data object
img.path	character with the new path to the images
subset	conditional expression to update the paths of a subset of images

Value

returns a cell.data object, with updated paths for the images

Author(s)

Alan Bush

See Also

cimage.cell.data,img.desc

Examples

```
## Not run:
#load example dataset
library(RcellData)
data(ACL394data)
summary(X)

#the default path has to be updated
new.path<-system.file(img, package=Rcell)
X<-update_img.path(X,new.path)
#a warning is issued because not all images were found
#(not all images are included in the package to reduce the package size)

#cimage can now find the images
cimage(X,channel~t.frame,subset=pos==29&cellID==5,channel=c(BF,YFP))

## End(Not run)
```

update_n.tot *Calculate Total Number of Frames for Each Cell*

Description

updates n.tot, the total amounts of frames in which a given cell appears

Usage

```
update_n.tot(object, QC.filter = TRUE,...)
```

Arguments

object	cell.data object
QC.filter	a boolean value indicating if the quality control filter should be applied
...	futher arguments for methods

Value

returns a cell.data object, with updated values for n.tot

Author(s)

Alan Bush

See Also

load.cellID.data,select.cells

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #update n.tot variable
  X<-update_n.tot(X)

  #this command is equivalent to
  X<-transformBy(X,.(ucid), n.tot=length(t.frame))
}
```

vplayout *Viewport functions*

Description

Multiple viewports per page

Usage

```
vplayout(x, y)
```

Arguments

x	x index of grid to use to print the ggplot2 figure
y	y index of grid to use to print the ggplot2 figure

Details

See documentation in package 'grid' for more details.

Author(s)

Alan Bush

See Also

transform.cell.data

Examples

```
if(require(RcellData)){
  #put several figures in a page
  data(ACL394)
  grid.newpage() #create a new plot
  pushViewport(viewport(layout = grid.layout(1, 2))) #define the grid for the plots
  print(cplot(X,f.tot.y~pos), vp = vplayout(1, 1))
  print(cplot(X,f.tot.y~a.tot,color=pos), vp = vplayout(1, 2))
}
```

with *Evaluates an Expression in a Cell Data Object.*

Description

Evaluate an R expression in an environment constructed from the cell.data object.

Usage

```
  ## S3 method for class cell.data
with(data,expr,subset=TRUE,select=NULL,exclude=NULL,QC.filter=TRUE,...)
```

Arguments

data	cell.data object
expr	expression to evaluate
...	arguments to be passed to future methods
subset	a boolean vector of length equal to the number of rows of the dataset, or a conditional statement using the dataset¥s variable, that specifies which registers should be included
select	character vector defining variables names to be included
exclude	character vector defining variables names to be excluded
QC.filter	a boolean value indicating if the quality control filter should be applied over the data

Details

with is a generic function. The version for cell.data objects is a wrapper over the version for data.frame, calling as.data.frame.cell.data with the specified arguments.

Value

The value of the evaluated expr

Author(s)

Alan Bush

See Also

with

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #calculate the mean f.tot.y from pos 2
  with(X,mean(f.tot.y[pos==2]))

  #use base plotting
  with(X,plot(f.tot.y~f.tot.c))

}
```

write.cell.image *Write a Cell Image*

Description

Writes a cell image object to disk.

Usage

```
write.cell.image(x, file = "",...)
```

Arguments

x	the cell.image object, as returned by get.cell.image, to be saved
file	filename or filename with path to save the image
...	further arguments passed to writeImage

Details

This function is a wrapper over writeImage. It combines images of the cell.image object in a stack, saves it to disk, and saves de image description database as a tab delimimited file, with the name given in file argument.

Author(s)

Alan Bush

See Also

writeImage

Examples

```
## Not run:
  #load example dataset
  library(RcellData)
  data(ACL394filtered)

  #select N=3 cells images from each pos (group),
  #from the first t.frame and pos 1,8,15,22,29.
  ci<-get.cell.image(X,subset=match(pos,c(1,8,15,22,29),nomatch=0)>0&t.frame==11,
    group=.(pos),N=3,channel=c(BF.out,YFP))

  write.cell.image(ci,"Example-cell-image.tif")


## End(Not run)
```

`write.delim` *Data output*

Description

Writes a Tab Delimited Table text table to disk.

Usage

```
write.delim(x, file = "", quote = FALSE, sep = "\t", row.names = FALSE,...)
```

Arguments

`x`	the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce x to a data frame.
`file`	either a character string naming a file or a connection open for writing. "" indicates output to the console.
`quote`	a logical value (TRUE or FALSE) or a numeric vector. If TRUE, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. In both cases, row and column names are quoted if they are written. If FALSE, nothing is quoted.
`sep`	the field separator string. Values within each row of x are separated by this string.
`row.names`	either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
`...`	further arguments passed to `write.table`

Details

This function is a wrapper over `write.table` with defaults to write a nice tab delimited text file.

Author(s)

Alan Bush

See Also

`write.table`

Examples

```
if(require(RcellData)){
  data(ACL394)
  agg<-aggregateBy(X,.(pos),select="f.tot.y",subset=t.frame==0,FUN=mean)
  if(interactive()) write.delim(agg,"myTable.txt")
}
```

zoom *Zoom in a ggplot Object*

Description

Sets the plotting region and axes breaks for a ggplot object

Usage

```
zoom(xzoom=c(NA,NA),yzoom=c(NA,NA),expand.y=c(0,0),expand.x=c(0,0),nx.breaks=n.breaks
,ny.breaks=n.breaks,n.breaks=7,...)
caxis(xzoom=c(NA,NA),yzoom=c(NA,NA),expand.y=c(0,0),expand.x=c(0,0),nx.breaks=n.breaks
,ny.breaks=n.breaks,n.breaks=7,...)
xzoom(xzoom=c(NA,NA),nx.breaks=7,...)
yzoom(yzoom=c(NA,NA),ny.breaks=7,...)
```

Arguments

xzoom	numeric vector. If length=2 it specifies the range of the x axis, if length>2 it gives the braks to be used.
yzoom	numeric vector. If length=2 it specifies the range of the x axis, if length>2 it gives the braks to be used.
expand.x	numeric vector of length two, with x axis additive expansion. Note the first element is usually negative. This expansion does not modify the position of the ticks.
expand.y	idem for y axis
nx.breaks	number of breaks for the x axis
ny.breaks	number of breaks for the y axis
n.breaks	number of breaks for both axes, if not specified by nx.breaks or ny.breaks
...	further arguments for pretty or scale_continuous

Details

xzoom and yzoom are convenient functions to specify only one of the limits.

Value

a layer to be added to a ggplot object, that specifies the plotting region after the statistical transformations have been done.

Note

A zoom function exists in Hmisc package. Use Rcell::zoom or caxis if both package namespaces are loaded.

Author(s)

Alan Bush

See Also

cplot,limits

Examples

```
if(require(RcellData)){

  #load example dataset
  data(ACL394)

  #zoom in the y axis
  cplotmeans(X,f.tot.y~t.frame,color=pos) + zoom(y=c(0,7e6))

  #define plotting region and ticks
  cplotmeans(X,f.tot.y~t.frame,color=pos) + caxis(y=c(0,7e6),x=c(0,13),nx=14,expand.x=c(-.75,.75))
}
```

Index