Package ‘rcell2.magick’

May 11, 2022

Title shiny and magick tools for CellID Data Analysis

Version 0.0.3

Description Analyse cellID data in the tidyverse framework. This package also has utility functions to generate, filter and preview data
from fluorescence microscopy experiments (and maybe general cell cytometry).
It is meant as a successor to the older RCell package.
This package includes the CellID program source, and wraps it in a single R function.
Also included are the tools to process CellID output (from tidyCell).
Finally, a R-Shiny app will help users filter data graphically, with previews.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.6)

biocViews

Imports tibble,
dplyr,
IRanges,
sp,
tidyr,
ggplot2,
magick,
foreach,
shiny,
hexbin,
shinydashboard,
formattable,
grDevices,
reshape2,
readr,
shinyjs,
rlang,
data.table,
keys

Suggests tidyverse,
skimr

R topics documented:

`all.unique` *Test whether all elements in a vector are unique*

Description

Uses `length()`.

Usage

```
## S3 method for class 'unique'
all(test_vector)
```

`annotation_magick` *Load image to a ggplot2 layer*

Description

Funcion basada en magick para abrir una foto y mostrarla en un ggplot2 layer.

Usage

```
annotation_magick(picPath, interpolate = FALSE)
```

Arguments

`picPath`	Image path for the correct position, only one.
`interpolate`	Passed on to layer params.

Value

A custom annotation_raster for ggplot.

`bind_filters` *Bind shinyCell polygon filters by variable pairs*

Description

Unites polygons with matching dimensions. Useful to check out what areas the filters are covering (see `plot_bound_filters`).

Usage

```
bind_filters(saved_data, print_plots = TRUE)
```

Arguments

`saved_data`	The output of shinyCell.

Value

A list of polygons bound by variable, with names unique to variable pairs (by sort). Note that "x" and "y" column names may be swapped relative to the input order.

`cellGif` *cellMagick configured for prodicing a ucid's gif*

Description

cellMagick configured for prodicing a ucid's gif

Usage

```
cellGif(
  cdata,
  paths,
  equalize_images = F,
  normalize_images = F,
  channels = c("BF.out"),
  animation_delay = 1/3,
  id_column = "ucid",
  time_colum = "t.frame",
  stack_channels_horizontally = TRUE,
  ...
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`equalize_images`	Use magick's function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`normalize_images`	Use magick's function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`channels`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default.
`animation_delay`	Delay between animation frames in seconds.
`stack_channels_horizontally`	Time increases from left to right (TRUE) or from up to down (FALSE).
`...`	Arguments passed on to `magickCell` `max_composite_size` Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default. `cell_resize` Resize string for the individual cell images (`NULL` translates to `boxSizexboxSize` by default).

`boxSize` Size of the box containing the individual cell images. 50 by default.

`n.cells` Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).

`customize_images` Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`image_customize` A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).

`ch` Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).

`sortVar` Variable name used to sort the rows after sampling if a `seed` was specified. NULL by default, to preserve the original or random sampling order.

`seed` Seed value for sampling of cell images. NULL by default, to disable sampling.

`.debug` Print more messages if TRUE.

`return_single_imgs` If TRUE, return a vector of images instead of a tile.

`return_ucid_df` If TRUE, return is a list of magick images and ucid dataframes.

`annotation_params` Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.

`add_border` Add a 1x1 border to the pictures. Useful for stacking/appending.

`stack_vertical_first` Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).

`return_raw` Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).

`crop_images` Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

`cellMagick` *magickCell alias*

Description

El uso más básico es `magickCell(cdata=cell.data$data, paths=cell.data$images)`.

Para mostrar algunas celulas en particular, solo hay que pasarle un `cdata` filtrado.

Ver la descripción de argumentos más abajo para aprender sobre las opciones.

Al fondo, en los detalles, hay una descripción de como debería ser el `paths` dataframe.

Usage

```
cellMagick(...)
```

Arguments

`...`	Arguments passed on to `magickCell`

`cdata` A Rcell data.frame (not the object).

`paths` A paths dataframe with file path, t.frame, position and channel information of each picture.

`max_composite_size` Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default.

`cell_resize` Resize string for the individual cell images (`NULL` translates to `boxSize`x`boxSize` by default).

`boxSize` Size of the box containing the individual cell images. 50 by default.

`n.cells` Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).

`equalize_images` Use magick's function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`normalize_images` Use magick's function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`customize_images` Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`image_customize` A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).

`ch` Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).

`sortVar` Variable name used to sort the rows after sampling if a `seed` was specified. NULL by default, to preserve the original or random sampling order.

`seed` Seed value for sampling of cell images. NULL by default, to disable sampling.

`.debug` Print more messages if TRUE.

`return_single_imgs` If TRUE, return a vector of images instead of a tile.

`return_ucid_df` If TRUE, return is a list of magick images and ucid dataframes.

`annotation_params` Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.

`add_border` Add a 1x1 border to the pictures. Useful for stacking/appending.

`stack_vertical_first` Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).

`return_raw` Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).

`crop_images` Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

Details

Paths dataframe structure. Output example from `glimpse(paths)`:

```
Columns: 6
$ pos     <int> 1
$ t.frame <int> 0
$ channel <chr> "YFP"
$ image   <chr> "picture_filename.tif"
$ path    <chr> "/path/to/directory/with/pictures/"
$ file    <chr> "/path/to/directory/with/pictures/picture_filename.tif"
$ is.out  <lgl> FALSE
```

Value

A list of two elements: the magick image and the ucids in the image.

`cellSpread` *2D binning of data and tiling of cell pictures*

Description

2D binning of data and tiling of cell pictures

Usage

```
cellSpread(
  cdata,
  paths,
  ch = "BF.out",
  boxSize = 80,
  xvar = "a.tot",
  yvar = "fft.stat",
  x.cuts = 7,
  y.cuts = 7,
  for_plotting = T,
  cut.by.content = F,
  ...
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`ch`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).

`boxSize`	Size of the box containing the individual cell images. 50 by default.
`xvar, yvar`	Strings indicating names for the variables to plot in the horizontal (x) and vertical (y) axis.
`x.cuts, y.cuts`	Integers indicating the number of cuts for each variable.
`for_plotting`	Return value changes to list of elements important for plotting.
`cut.by.content`	Use quantile to generate cuts of "roughly equal content (rather than length)."
`...`	Arguments passed on to `magickCell`

`max_composite_size` Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default.

`cell_resize` Resize string for the individual cell images (`NULL` translates to `boxSizexboxSize` by default).

`n.cells` Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).

`equalize_images` Use magick's function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`normalize_images` Use magick's function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`customize_images` Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`image_customize` A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).

`sortVar` Variable name used to sort the rows after sampling if a `seed` was specified. NULL by default, to preserve the original or random sampling order.

`seed` Seed value for sampling of cell images. NULL by default, to disable sampling.

`.debug` Print more messages if TRUE.

`return_single_imgs` If TRUE, return a vector of images instead of a tile.

`return_ucid_df` If TRUE, return is a list of magick images and ucid dataframes.

`annotation_params` Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.

`add_border` Add a 1x1 border to the pictures. Useful for stacking/appending.

`stack_vertical_first` Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).

`return_raw` Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).

`crop_images` Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

cellSpreadPlot *Plot for 2D binning of data and tiling of cell pictures*

Description

Plot for 2D binning of data and tiling of cell pictures

Usage

```
cellSpreadPlot(
  cdata,
  paths,
  ch = "BF.out",
  boxSize = 80,
  xvar = "a.tot",
  yvar = "fft.stat",
  overlay_points = F,
  underlay_points = F,
  draw_contour_breaks = NULL,
  x.cuts = 7,
  y.cuts = 7,
  ...
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`ch`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).
`boxSize`	Size of the box containing the individual cell images. 50 by default.
`xvar`	Strings indicating names for the variables to plot in the horizontal (x) and vertical (y) axis.
`yvar`	Strings indicating names for the variables to plot in the horizontal (x) and vertical (y) axis.
`overlay_points`	Overlay data points to the image plot.
`underlay_points`	Underlay data points to the image plot.
`draw_contour_breaks`	Overlay a ggplot2::stat_density2d layer. If TRUE, use the default breaks. Otherwise NULL for none, or a numeric vector for the density breaks.
`x.cuts`	Integers indicating the number of cuts for each variable.

`y.cuts`	Integers indicating the number of cuts for each variable.
`...`	Arguments passed on to `cellSpread` `for_plotting` Return value changes to list of elements important for plotting. `cut.by.content` Use quantile to generate cuts of "roughly equal content (rather than length)."

`cellStrip` *cellMagick configured for prodicing a ucid's strip*

Description

cellMagick configured for prodicing a ucid's strip

Usage

```
cellStrip(
  cdata,
  paths,
  equalize_images = F,
  normalize_images = F,
  channels = c("BF.out"),
  animation_delay = 1/3,
  stack_time_horizontally = TRUE,
  id_column = "ucid",
  time_colum = "t.frame",
  ...
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`equalize_images`	Use magick's function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`normalize_images`	Use magick's function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`channels`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default.
`stack_time_horizontally`	Time increases from left to right (TRUE) or from up to down (FALSE).

`...` Arguments passed on to `magickCell`

- `max_composite_size` Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default.
- `cell_resize` Resize string for the individual cell images (`NULL` translates to `boxSize`x`boxSize` by default).
- `boxSize` Size of the box containing the individual cell images. 50 by default.
- `n.cells` Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).
- `customize_images` Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
- `image_customize` A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).
- `ch` Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).
- `sortVar` Variable name used to sort the rows after sampling if a `seed` was specified. NULL by default, to preserve the original or random sampling order.
- `seed` Seed value for sampling of cell images. NULL by default, to disable sampling.
- `.debug` Print more messages if TRUE.
- `return_single_imgs` If TRUE, return a vector of images instead of a tile.
- `return_ucid_df` If TRUE, return is a list of magick images and ucid dataframes.
- `annotation_params` Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.
- `add_border` Add a 1x1 border to the pictures. Useful for stacking/appending.
- `stack_vertical_first` Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).
- `return_raw` Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).
- `crop_images` Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

`cellStrips` *Wraps cellMagick to make strips, optionally cutting them.*

Description

First, ‘cdata‘ is split by ‘split_col‘ and then images are generated. Only the first ‘n_ucids‘ in ‘cdata‘ are processed. Then ‘images‘ are split with ‘cut‘, which is useful wen strips are too long.

Usage

```
cellStrips(
  cdata,
  paths,
  n_ucids = NULL,
  cut_breaks = 1,
  split_col = "ucid",
  ch = c("BF.out", "yfp.out"),
  sortVar = "t.frame",
  ...
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`n_ucids`	Will select the first ‘n_ucids‘, by default (NULL) it selects all.
`cut_breaks`	Will split each strip into ‘cut_breaks‘ pieces.
`split_col`	Column from cdata used to separate different sets of cells.
`ch`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).
`sortVar`	Column from cdata used to sort the pictures in the strips.
`...`	Arguments passed on to `magickCell`

`max_composite_size` Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default.

`cell_resize` Resize string for the individual cell images (`NULL` translates to `boxSizexboxSize` by default).

`boxSize` Size of the box containing the individual cell images. 50 by default.

`n.cells` Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).

`equalize_images` Use magick’s function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`normalize_images` Use magick’s function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`customize_images` Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).

`image_customize` A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).

`seed` Seed value for sampling of cell images. NULL by default, to disable sampling.

`.debug` Print more messages if TRUE.

`return_single_imgs` If TRUE, return a vector of images instead of a tile.

`return_ucid_df` If TRUE, return is a list of magick images and ucid dataframes.

`annotation_params` Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.

`add_border` Add a 1x1 border to the pictures. Useful for stacking/appending.

`stack_vertical_first` Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).

`return_raw` Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).

`crop_images` Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

`cut_strips` Use 'cut' to split the image series (by index; preserves sortVar order).

Examples

```
# Not run
if(F){
strips <-
  cellStrips(cdata = cdata %>% filter(ucid == 20308),
             paths = image.paths,
             ch = c("BF", "BF.out", "YFP.out"), equalize_images=T,
             # n_ucids = 2,
             cut_breaks = 3)

# Get one cell
strips$`20308` %>%

  # For "cut > 1"
  image_join() %>%
  image_append() %>%

  # For rmarkdown inline rendering
  rcell2::magickForKnitr() %>% knitr::include_graphics()
}
```

`get_workflow_template_magick`

A function to donwload the latest worflow tempalte in Rmarkdown

---

Description

Will donwload the .Rmd file to the current working directory.

Usage

```
get_workflow_template_magick(file_name = "rcell2.magick_workflow_template.Rmd")
```

Arguments

`file_name`	File name for the wokflow template.

`image_border_one` *Add a border to one side of the image*

Description

image_border_one is good for padding images to add captions later, while magick's original image_border adds borders symmetrically to all sides.

Usage

```
image_border_one(image, geometry = "0x15", color = "white")
```

Arguments

`image`	magick image object.
`geometry`	magick geometry string.
`color`	color string for the border background.

`magickCell` *Funcion copada para mostrar fotos de Cell-ID basada en magick*

Description

El uso más básico es `magickCell(cdata=cell.data$data, paths=cell.data$images)`.

Para mostrar algunas celulas en particular, solo hay que pasarle un `cdata` filtrado.

Ver la descripción de argumentos más abajo para aprender sobre las opciones.

Al fondo, en los detalles, hay una descripción de como debería ser el `paths` dataframe.

Usage

```
magickCell(
  cdata,
  paths,
  max_composite_size = 1000,
  cell_resize = NULL,
  boxSize = 80,
  n.cells = 25,
  equalize_images = F,
  normalize_images = F,
  customize_images = F,
  image_customize = NULL,
  ch = "BF.out",
  sortVar = NULL,
  seed = NULL,
  .debug = FALSE,
  return_single_imgs = FALSE,
  return_ucid_df = FALSE,
  annotation_params = list(color = "white", background = "black"),
  add_border = TRUE,
  stack_vertical_first = FALSE,
  return_raw = FALSE,
  crop_images = TRUE
)
```

Arguments

`cdata`	A Rcell data.frame (not the object).
`paths`	A paths dataframe with file path, t.frame, position and channel information of each picture.
`max_composite_size`	Maximum size of the final composite image (this resize is applied last) in pixels. 1000 by default.
`cell_resize`	Resize string for the individual cell images (`NULL` translates to `boxSize`x`boxSize` by default).
`boxSize`	Size of the box containing the individual cell images. 50 by default.
`n.cells`	Maximum number of cells to display (integer, set to `NULL` to display all cells in `cdata`).
`equalize_images`	Use magick's function to "equalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`normalize_images`	Use magick's function to "normalize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`customize_images`	Use a custom magick-like function to "customize" the image when TRUE (FALSE by default). Can be a logical vector, each value applied separately to each channel (recycled to the length of `ch`).
`image_customize`	A custom magick-like function to "customize" the channels specified in `customize_images`. Defaults to NULL (disabled).
`ch`	Name of the CellID channel (BF, BF.out, RFP, etc.). "BF.out" by default, use a vector to select more than one channel simultaneously (images will be stacked).
`sortVar`	Variable name used to sort the rows after sampling if a `seed` was specified. NULL by default, to preserve the original or random sampling order.

`seed`	Seed value for sampling of cell images. NULL by default, to disable sampling.
`.debug`	Print more messages if TRUE.
`return_single_imgs`	If TRUE, return a vector of images instead of a tile.
`return_ucid_df`	If TRUE, return is a list of magick images and ucid dataframes.
`annotation_params`	Set to NULL to skip annotations, or a named list with values for magick::annotate options (one or more of the names "color" "background" "size"). Note that size close to zero can be invisible.
`add_border`	Add a 1x1 border to the pictures. Useful for stacking/appending.
`stack_vertical_first`	Set to TRUE to stack images vertically first (useful when `return_single_imgs = T`).
`return_raw`	Returns loaded images prematurely (i.e. without any processing other than magick::image_read and magick::image_crop).
`crop_images`	Whether to crop images to a box centered on the cell's XY position (TRUE, default), or the full image (FALSE).

Details

Paths dataframe structure. Output example from `glimpse(paths)`:

```
Columns: 6
$ pos     <int> 1
$ t.frame <int> 0
$ channel <chr> "YFP"
$ image   <chr> "picture_filename.tif"
$ path    <chr> "/path/to/directory/with/pictures/"
$ file    <chr> "/path/to/directory/with/pictures/picture_filename.tif"
$ is.out  <lgl> FALSE
```

Value

A list of two elements: the magick image and the ucids in the image.

`magickForKnitr` *Display an image in rmarkdown with knitr*

Description

Display an image in rmarkdown with knitr

Usage

```
magickForKnitr(
  imgs,
  .prefix = "tile",
  .resize = NULL,
  .path = tempdir(),
  .print = F
)
```

Arguments

`imgs`	a magick image or cellMagick output
`.prefix`	A string prepended to the file name, "" by default.
`.resize`	a cellMagick image resize string as "200x200" (default NULL, for no resizing).
`.path`	Directory where the output should be saved.
`.print`	Print the image path.

Value

An path to a temporary image file.

`magickPaths` *Extaer paths para cellMagick del objeto cell.data*

Description

Extaer paths para cellMagick del objeto cell.data

Usage

```
magickPaths(cell.data)
```

Arguments

`cell.data`	directory where images are stored, full path.

Value

A dataframe with paths.

`plotApp` *Filtrar cdata usando gráficos y dibujando regiones*

Description

Filtrar cdata usando gráficos y dibujando regiones

Usage

```
plotApp(user_plot, debug_messages = F, print_plot_on_exit = F)
```

Arguments

`user_plot`	A ggplot object, ready to render.
`debug_messages`	Print internal stuff, useful for debugging.
`print_plot_on_exit`	Prints user_plot replacing it's data with the "brushed" subset

Value

A data.frame with the brushed points from the original plot's data.

`plot_bound_filters` *Plot bound shinyCell polygon filters by variable pairs*

Description

Useful to check out what areas the filters are covering.

Usage

```
plot_bound_filters(bound_filters)
```

Arguments

`bound_filters`	The output of `bind_filters`.

Value

Plots for the polygons in bound_filters.

`plot_filters` *Plot shinyCell polygon filters*

Description

Useful to check out what areas the filters are covering.

Usage

```
plot_filters(saved_data, print_plots = TRUE)
```

Arguments

`saved_data`	The output of shinyCell, or a list: `list(cdata = NULL, filter = saved_data$filters)`. Note that the only effect of `cdata = NULL` is that points will not be drawn.
`print_plots`	Set to false to prevent printing the plots on execution.

Value

A list of ggplots ready to print.

`polyFilterApply` *Apply polygonal filters to cdata*

Description

Adds a boolean "filter" column to the "cdata" dataframe, based on a polygon list (typically output by shinyCell).

Usage

```
polyFilterApply(
  polygon_df_list,
  cdata,
  truthMode = "all",
  cell_unique_id_field = "ucid"
)
```

Arguments

`polygon_df_list`	A list of polygon dataframes with columns: x (values) y (values) xvar (variable name for x values) yvar (variable name for y values) type ("Subtractive" or "Additive")
`cdata`	A "cdata" dataframe.

`truthMode`	Priority for "Subtractive" and "Additive" polygon filter types, passed to `calculateTruth`. Must be either "all" (Subtractive overcomes Additive) or "any" (Additive overcomes Subtractive).
`cell_unique_id_field`	Name for the column holding the unique identifier (a "primary key") for each data point (i.e. the "ucid" is not suficcient for time series datasets).

Value

a "saved_data" list object, where the cdata is appended a "filter" logical column.

`rcell2.magick` *Yeast Cell Cytometry Suite for CellID in R.*

Description

It is a rewrite and revamp of the previously awesome Rcell package, by Dr. Alan Bush. Plotting functions from that package have been excluded in the name of minimalism. Thoughfully, `ggplot2` definitions for all those "cplots" are available in our vignettes (happy copy-pasting!).

Details

The full `rcell2` functionality is split into three main packages: rcell2, `rcell2.cellid`, and `rcell2.magick`.

rcell2 (tidyCell) functions

The tidyCell functions manage CellID's output. Also useful to turn custom data into compatible dataframes for the other functions in this package.

rcell2.magick functions

`magickCell` renders images from individual cells, based on original images, user defined filters and data from cells. It should not be required that the data comes from images processed by CellID, it only requires appropriate data frames for observation data (cdata) and for image metadata (paths). It relies heavily on the magick package, so the equirement of the `imagemagick` system library might be inconvenient. That is why this awesome feature is optional.

`shinyCell` is an R-Shiny based graphical interface, meant to filter, inspect and annotate cells relying on 2D plots of arbitrary variables. It should not be required that the data comes from images processed by CellID, just as `magickCell`. It relies heavily on the magick package, so the equirement of the `imagemagick` system library might be inconvenient. That is why this awesome feature is optional. Also included are two apps: `tagCell` for annotation, and `plotApp` for plotting and subsetting.

rcell2.cellid functions

This package includes CellID's source code (written in C). During installation, it compiles a binary executable for Linux, Windows, and Mac systems. The executable is then wrapped seamlessly by the `cell2` function, such that CellID can now be used programatically within R. If you already have CellID in your system, you are welcome to use it instead of the bundled executable.

See Also

magick

`safe_select` *Safe select column*

Description

Checks whether the column selection with `[[]]` is null before returning.

Usage

```
safe_select(.df, .name)
```

`shinyCell` *Preview images and filter data using regions in 2D plots*

Description

This R-Shiny app helps the user browse the dataset graphically.

Usage

```
shinyCell(
  cdata,
  pdata = NULL,
  paths,
  filters = list(),
  filters.init_selected = T,
  plotType = "Dots",
  seed = 1,
  initial_facet = "",
  initial_vars = c("a.tot", "fft.stat"),
  facet_grid_option = TRUE,
  facets_scale_free = "fixed",
  n_max = 100,
  boxSize = 80,
  filter_progress_file = NULL,
  launch.browser = F,
  skip_input_check = F,
  ...
)
```

Arguments

`cdata`	A Rcell "cdata" data.frame with the CellID variables (not the cell.data object, `cell.data$data`).
`pdata`	An optional "pdata" data.frame, with positions' metadata (NULL by default).
`paths`	A "paths" data.frame, with paths to each positions' images (i.e. `cell.data$images`).
`filters`	An optional list with the filters from a previous shinyCell run (dataframes with points of 2D polygons). An empty `list()` by default.
`plotType`	Type of the filtering plot, either: "Dots" (point scatterplot, defaut), "Hex" (2D histogram with hexagonal bins), "Density", "Pics" (a `cellSpread` plot).
`seed`	Seed value for sampling of cell images.
`initial_facet`	Initial ggplot facet formula as a string (for example: "~pos+group_1")
`initial_vars`	Initial cdata variables as a character vector (defaults to `c('a.tot', 'fft.stat')`).
`facet_grid_option`	Use ggplot's facet_grid (TRUE, default) or facet_wrap (FALSE).
`facets_scale_free`	Use ggplot's facets with fixed scales (NULL, default) or free scales ("free").
`boxSize`	Size in pixels for individual cells' images.
`filter_progress_file`	Path to an RDS file, used for saving filtering progress (in case something goes wrong). Using FALSE disables this feature. Set to NULL (the default) to let tempfile() choose a path for the RDS, or set to a valid path of your choice.
`launch.browser`	Set to `'firefox'` or equivalent to launch the app in-browser (`FALSE` by default). Useful when launching fails with error `Error in utils::browseURL(appUrl)` or similar.
`skip_input_check`	If FALSE (default)
`...`	Further arguments passed to `magickCell`.

Details

The filtering logic in the "Filter mode" setting is documented and implemented in `polyFilterApply`. Briefly, it defines the priority of Subtractive and Additive filters: which of them should override the other?

Value

A named list with the original cdata and a list of filters. The cdata includes an extra "filter" column, indicating if a row is to be kept (TRUE) or filtered out (FALSE). The list of filters can be passed as a filter argument, and can be plotted with `plot_filters`.

See Also

magickCell

Examples

```
# Minimal example:

path <- "/path/to_your/cellid_images/"

cell.data <- rcell2::cell.load.alt(path = path)

cdata <- cell.data$data  # CellID dataframe

images <- cell.data$images  # Image paths

pdata <- read.csv("data/pdata.csv")  # "Position" metadata

filter.output <-
  rcell2::shinyCell(cdata = cdata,
                    pdata = pdata,
                    paths = images)

plot_filter(filter.output)

cdata.filtered <- dplyr::filter(filter.output, filter)
```

`square_tile` *Armar mosaicos cuadrados a partir de un vector de imagenes en magick de cualquier longitud*

Description

Armar mosaicos cuadrados a partir de un vector de imagenes en magick de cualquier longitud

Usage

```
square_tile(
  images,
  annot_labels = NULL,
  nRow = NULL,
  nCol = NULL,
  annotate_images_with_index = T,
  debug.msgs = F
)
```

Arguments

`images`	A magick image vector, with images of the same size preferably.
`annot_labels`	A vector with the same lenght as images, holding annotation labels.
`nRow`	An integer indicating number of rows. If specified, nCol must be specified too.
`nCol`	An integer indicating number of rows. If specified, nRow must be specified too.
`annotate_images_with_index`	Annotate images with their index (default TRUE).
`debug.msgs`	Print indexes used at each row.

Value

A single magick image of the squared tile.

`tagCell` *Filtrar cdata usando gráficos y dibujando regiones*

Description

Filtrar cdata usando gráficos y dibujando regiones

Usage

```
tagCell(
  cdata,
  pdata,
  paths,
  cell_tags,
  randomize_ucids = FALSE,
  tag_box_size = 50,
  cell_resize = NULL,
  tag_channels_select = c("BF", "BF.out"),
  seed = 1,
  tmp_output_file = NULL,
  tag_ggplot = NULL,
  equalize_images = F,
  normalize_images = F,
  max.frames = 10,
  tags.df = NULL,
  verbose = 0
)
```

Arguments

`cdata`	dataframe of "cell data".
`pdata`	dataframe "position data".
`paths`	dataframe of image paths.
`cell_tags`	list of named vectors corresponding to tag groups and tags: list(named_item1 = c(option1, option2, ...), named_item2 ...).
`randomize_ucids`	Randomize ucid order.
`tag_box_size`	size of the image crop in pixels (integer).
`cell_resize`	resize of the image crop in pixels (integer).
`tag_channels_select`	a vector giving names for the image channels: c("BF", "YFP.out", etc....).
`seed`	seed for random sampling of images.
`tmp_output_file`	File path into which tagging information will be dumped by user request. NULL by default, to automatically create and append to a tmp file.
`tag_ggplot`	a ggplot object to display in the second tab, may be used for something someday.
`equalize_images`	Use magick's function to "equalize" the images.
`normalize_images`	Use magick's function to "normalize" the images.
`max.frames`	Max number of t.frames to render in the cell strip. Set to 0 to disable.
`tags.df`	Previous tag dataframe, used to restore or view previous tags in the app (restores tags that are named in the cell_tags list).
`verbose`	Print debugging messages (with levels at either 0, 1 or 2).

Value

Lots of stuff.

Examples

```
path <- "/mac/apesta/trololololol/"

cell.data <- rcell2::cell.load.alt(path = path)

image.paths <- cell.data$d.paths  # Si usaste load_cell es: image.paths <- rcell2::magickPat

pdata <- read_tsv(paste0(path, "pdata.csv"))

cdata <- left_join(cell.data$d, pdata)

p <- ggplot() +
  geom_line(aes(x=t.frame, y=cf.y, group=ucid))
```

```
tag_channels_select <- c("BF", "BF.out", "YFP", "YFP.out")

saved <- rcell2::tagCell(cdata,
                         pdata,
                         image.paths,
                         cell_tags = list(far1_drop = c(TRUE,
                                                        FALSE),
                                          budding =   c("emergence",
                                                        "division",
                                                        "shmoo_o_algo"),
                                          artifact =  c("segmentation",
                                                        "crowding",
                                                        "out_of_focus",
                                                        "interesante",
                                                        "death",
                                                        "flown_away",
                                                        "not_a_cell")
                         ),
                         tag_channels_select = tag_channels_select,
                         equalize_images = T,
                         normalize_images = F,
                         n_max = 50,
                         tag_box_size = 75,
                         cell_resize = 300,
                         tag_ggplot = p,
                         tmp_output_file = "../output/annotations/progress.csv",
                         debug_messages = F
                         )
```

`tags.to.onehot` *Pivot cell tags to a cdata-joinable dataframe, with one hot encoding*

Description

Pivot cell tags to a cdata-joinable dataframe, with one hot encoding

Usage

```
tags.to.onehot(tags.df, exclude.cols = c("pos", "cellID", "viewed"))
```

Arguments

`tags.df`	Output from tagCell.
`exclude.cols`	Character vector with names of columns which should be removed from input.

Details

Ver: * ‘~/Projects/Academia/Doctorado/gitlabs_acl/rtcc/far1/analisis_Far1_arresto-lavado/R/analisis_pos_2_a_7_v7_tags_a
* https://stackoverflow.com/questions/55288338/r-multi-hot-encoding-among-multiple-columns *
https://stackoverflow.com/a/63454411/11524079

`updateList` *Update a list’s value using another list, by common names.*

Description

Names of ‘l2‘ present in ‘l1‘ will update values in ‘l1‘.

Usage

```
updateList(l1, l2, only.common.names = T)
```

Arguments

`l1`	List to be updated (with the "original" or "old" values).
`l2`	List used for updating (with the "newer" values). Note: it needn’t have all names
`...`	Arguments passed on to magickCell.

Details

Names of ‘l2‘ absent in ‘l1‘ will be ignored, unless ‘only.common.names=F‘.

Index