Package 'rcell2.cellid'

April 29, 2022

Title CellID bundled in R for the rcell2 package: CellID Data Analysis Inside the Tidyverse

Version 0.0.3

Description Generate cellID data entirely within R.
This package includes the CellID program source, and wraps it in a single R function ``cell2", while providing additional functions to prepare its input, test parameters, load it's output, etc.
To analyze CellID's data, we susgest using our related packages: ``rcell2" for a tidy analysisi framework, and ``rcell2.magick" for manipulating data using shiny apps, and images with R's magick package.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 3.6)

biocViews

Imports data.table (>= 1.14.2),
doParallel (>= 1.0.17),
dplyr (>= 1.0.8),
foreach (>= 1.5.2),
parallel,
purrr (>= 0.3.4),
readr (>= 2.1.2),
rlang (>= 1.0.2),
stringr (>= 1.4.0),
tibble (>= 3.1.6),
tidyr (>= 1.2.0)

Suggests skimr (>= 2.1.3),
tidyverse (>= 1.3.1),
rcell2,
rcell2.magick

R topics documented:

`arguments` *Obtener argumentos para CellID*

Description

Obtener argumentos para CellID

Usage

```
arguments(
  path,
  parameters = rcell2.cellid::parameters_write(),
  BF.pattern = "^BF",
  file.pattern = "^(BF|[A-Z]FP)_Position(\\d+)_time(\\d+).tif$",
  file.pattern.groups.order = c("ch", "pos", "t.frame"),
  output.dir.basename = "Position",
  tiff.ext = "tif$"
)
```

Arguments

`path`	directory where images are stored, full path.
`parameters`	path to the parameters file or a data.frame with "pos" (position number) and "parameter" (path) columns. Defaults to `parameters_write()`.
`BF.pattern`	regex pattern to detect BF images only. Defaults to: `"^BF"`
`file.pattern`	regex pattern for all tif files, with one group for each of `c("ch","pos","t.frame")` in `file.pattern.groups.order`. Uses `"^(BF|[A-Z]FP)_Position(\d+)_time(\d+).tif$"` by default. To omit time, use an empty group for the t.frame in the regex, for example: `"^(BF\|[A-Z]FP)_Position(\d+)().tif$"`. To consider Z-stacks, use something like "^(BF|[A-Z]\d+)_Position(\d+)_time(\d+).tif$"
`file.pattern.groups.order`	a character vector of components `c("ch","z","pos","t.frame")` with order corresponding to the order of groups in `file.pattern`.
`output.dir.basename`	Basename for the CellID output directories for each position.
`tiff.ext`	regex pattern for the tif file extension

Details

All 4 regex groups are mandatory, ‘t.frame‘ may be left as empty parenthesis, while also preserving group order defined by ‘file.pattern.groups.order‘.

The "channel" and "pos" regex groups _must always_ match pos and channel identifiers in the file name.

Example ‘file.pattern‘ regex, when ‘file.pattern.groups.order = c("ch", "pos", "t.frame")‘:

With Z planes time: `file.pattern = "^(BF|[TYR]FP|[TYR]\d{2})_Position(\d+)_time(\d+).tif$"`

No Z planes, with time (note the empty parentheses): `file.pattern = "^(BF|[A-Z]FP)_Position(\d+)_time(\d+).t`

No Z planes, no time: `file.pattern = "^(BF|[TYR]FP)_Position(\d+)().tif$"`

Value

a data.frame with all the information needed to run CellID

---

arguments_summary *Print rcellid arguments summaries*

Description

A function to print some summaries, to check cellArgs2 output.

Usage

```
arguments_summary(arguments)
```

Arguments

`arguments`	The "arguments" dataframe, output from `rcell2.cellid::arguments()`.

---

arguments_to_images *Make and "images" dataframe from "arguments" dataframe*

Description

The images dataframe is needed by many rcell2 functions. If it is not available from the output of `load_cell_data` or `cell.load.alt`, then this function can help.

Usage

```
arguments_to_images(arguments)
```

Arguments

`arguments`	The "arguments" dataframe, output from `rcell2.cellid::arguments()`.

Details

It essentially does a pivot_longer of the arguments.

Value

A data.frame similar to `cell.load.alt()$images`.

`cell.load.alt` *Cargar el output de cell-id*

Description

Cargar el output de cell-id

Usage

```
cell.load.alt(
  path,
  pdata = NULL,
  position.pattern = ".*Position(\\d+).*",
  fluorescence.pattern = "^(BF|[GCYRT]FP|[GCYRT]\\d+)_Position\\d+.*.tif$",
  ucid.zero.pad = 4,
  append.posfix = NULL,
  ...
)
```

Arguments

`path`	Path to CellID's output directory, tipically also the images directory.
`pdata`	Path to metadata CSV file.
`position.pattern`	Regex describing what the position string looks like (default ".*Position(\d+).*") including a capturing group for the position ID number (coerced to integer).
`fluorescence.pattern`	Regex describing what the fluorescence/channel ID string looks like (default "^([GCYRT]FP|[GCYRT]\d+)_Position\d+_time\d+.tif$"). There must be only one capturing group, ant it must be for the channel identifier.
`ucid.zero.pad`	Amount of decimal digits for the cellID (defaults 4, corresponding to a maximum of 9.999 cellIDs and 9999 positions).
`append.posfix`	String appended to the channel ID extracted by 'fluorescence.pattern' ('NULL' by default, but "FP" is usual).
`...`	Arguments passed on to `load_out_all` `out_file_pattern` Regex matching CellID's main output file. `out_mapping_pattern` Regex matching CellID's image mapping output file.

Value

A list of dataframes: data (CellID data), images (images metadata and paths), image_maping (extra mapping metadata from CellID: BF to FL correspondence, channel flag, bf_as_fl flag, and one-letter channel encoding).

`cell2` *Function to run CellID*

Description

Function to run CellID

Usage

```
cell2(
  arguments,
  cell.command = NULL,
  n_cores = NULL,
  debug_flag = 0,
  dry = F,
  encode_cellID_in_pixels = F,
  fill_interior_pixels = F,
  label_cells_in_bf = F,
  output_coords_to_tsv = F,
  save.logs = T
)
```

Arguments

`arguments`	An argument data.frame, as built by `rcell2.cellid::arguments`.
`cell.command`	By default `NULL`, to use the built-in binary. Otherwise a path to a CellID binary executable (get if from https://github.com/darksideoftheshmoo/cellID-linux).
`n_cores`	Number of cores to use for position-wise parallelization,internally capped to number of positions in `arguments`. Set to 1 to disable parallelization. If NULL, defaults to available cores - 1.
`debug_flag`	Set to 0 to disable CellID printf messages (builtin CellID only).
`dry`	Do everything without actually running CellID, print the commands that would have been issued.
`encode_cellID_in_pixels`	Set to TRUE to write cell interior and boundary pixels with intensity-encoded CellIDs and blank the rest of the image (CellID option '-s').
`fill_interior_pixels`	Set to TRUE to fill each cell interior area in the output image file with intensity-labeled pixels (CellID option '-i').
`label_cells_in_bf`	Set to TRUE to enable labeling of cells with their CellID in the BF output image using number characters (CellID option '-l', default FALSE).
`output_coords_to_tsv`	Set to TRUE to write cell interior masks and boundary pixels data to a .tsv file in the output directory (CellID option '-m').
`save.logs`	Set to TRUE to save CellID logs to text files, into the output directory of their corresponding position.

Value

A dataframe with one column indicating the issued commands and exit codes (in the command.output column). If the execution was sucessful, now you may run `rcell2::load_cell_data` or `rcell2.cellid::cell.load.` to get the results from the CellID output, typically located at the images path.

`cellid_output_descriptions`
Cell-ID output descriptions

Description

Cell-ID output descriptions

Usage

```
cellid_output_descriptions(list.output = T)
```

Arguments

`list.output`	Return the descriptions as a named list (TRUE), or as a data.frame (FALSE).

`cellid_parameter_descriptions`
Cell-ID parameter descriptions

Description

Cell-ID parameter descriptions

Usage

```
cellid_parameter_descriptions(list_format = T)
```

Arguments

`list_format`	If TRUE then format the dataframe into a named list

get_workflow_template_cellid

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_cellid(
  file_name = "rcell2.cellid_workflow_template.Rmd",
  open.template = T
)
```

Arguments

file_name	File name for the downloaded wokflow template.
open.template	Try using file.edit to open the file in RStudio after getting it.

imagej.fft.filter

Run ImageJ FFT filter macro from R

Description

Passes an ImageJ FFT filter on files matching `BF.*.tif` in the target directory.

Usage

```
imagej.fft.filter(
  pic.path,
  script.path = system.file("imagej_macros/FFT_filter_on_BFs_R.txt", package =
    "rcell2.cellid"),
  ...
)
```

Arguments

pic.path	Path to the directory containing the image files (passed as extra.args to imagej.macro.run).
script.path	Path to the ImageJ macro. Defaults to built-in macro.
...	Arguments passed on to `imagej.macro.run` `imagej.path` Path to the ImageJ binary (a path to "ImageJ-linux64" or equivalent). `extra.args` A string with extra arguments to the ImageJ command, pasted at the end. `headless` Wether ImageJ should be run headlessly (no GUI).

`wait` a logical (not `NA`) indicating whether the R interpreter should wait for the command to finish, or run it asynchronously. This will be ignored (and the interpreter will always wait) if `intern = TRUE`. When running the command asynchronously, no output will be displayed on the `Rgui` console in Windows (it will be dropped, instead).

Details

The modified images are saved to a new subdirectory with default name: "filtered/" (hardcoded in the macro).

`imagej.macro.run` *Run headless ImageJ Macro file*

Description

Run headless ImageJ Macro file

Usage

```
imagej.macro.run(
  script.path,
  imagej.path = "~/Software/Fiji.app/ImageJ-linux64",
  wait = T,
  headless = T,
  extra.args = ""
)
```

Arguments

`script.path`	Path to the ImageJ macro. Defaults to built-in macro.
`imagej.path`	Path to the ImageJ binary (a path to "ImageJ-linux64" or equivalent).
`wait`	a logical (not `NA`) indicating whether the R interpreter should wait for the command to finish, or run it asynchronously. This will be ignored (and the interpreter will always wait) if `intern = TRUE`. When running the command asynchronously, no output will be displayed on the `Rgui` console in Windows (it will be dropped, instead).
`headless`	Wether ImageJ should be run headlessly (no GUI).
`extra.args`	A string with extra arguments to the ImageJ command, pasted at the end.

parameters_default *Default parameters list for Cell-ID*

Description

Returns a list of key-value pairs, for the default Cell-ID parameters. It's output will tipically be used by parameters_write.

Usage

```
parameters_default(
  max_split_over_minor = 0.5,
  max_dist_over_waist = 8,
  max_pixels_per_cell = 2000,
  min_pixels_per_cell = 75,
  background_reject_factor = 0.75,
  tracking_comparison = 0.2,
  align_individual_cells = F,
  align_fl_to_bf = F,
  align_fl_to_first = F,
  image_type = "brightfield",
  bf_fl_mapping = "list",
  treat_brightfield_as_fluorescence_also = F
)
```

Arguments

max_split_over_minor
: Default: 0.50 For every combination of two pixels on the boundary, Cell-ID calculates the distance along the boundary path divided by the Euclidean distance between them. The maximum value of this ratio is larger for cells with a "figure-eight" shape that were pinched in some part than for circular cells. If the maximum value is above a user-defined threshold (which defaults to max_dist_over_waist=6), then the cell is split into two cells at the location of the pinch. After a split, if the Euclidean distance divided by the length of the minor axis of either of the new cells is greater than a user-defined value (which defaults to max_split_over_minor=0.5), then the two cells are re-grouped as a single cell. Thus, to perform the split we require that the two new cells have a generally circular shape and are not too elongated, as would be the case if the previous split was not over two cells, but over a cell and its mating projection.

max_dist_over_waist
: Default: 8.00 For every combination of two pixels on the boundary, Cell-ID calculates the distance along the boundary path divided by the Euclidean distance between them. The maximum value of this ratio is larger for cells with a "figure-eight" shape that were pinched in some part than for circular cells. If the maximum value is above a user-defined threshold (which defaults to max_dist_over_waist=6), then the cell is split into two cells at the location of the pinch. After a split, if the Euclidean distance divided by the length of the minor axis of either of the new cells is greater than a user-defined value (which defaults to max_split_over_minor=0.5), then the two cells are re-grouped as a single cell. Thus, to perform the split we require that the two new cells have a

generally circular shape and are not too elongated, as would be the case if the previous split was not over two cells, but over a cell and its mating projection.

`max_pixels_per_cell`
: Default: `2000` Area limits per cell (upper bound, in pixels).

`min_pixels_per_cell`
: Default: `75` Area limits per cell (lower bound, in pixels).

`background_reject_factor`
: Default: `0.75` CellID's code makes an initial decision about the graylevels of the boundary pixels. To do this it takes the mean position of all the graylevels in the images and subtracts Z standard deviations. It then starts by considering all gray levels below this value as being parts of the cell borders. This value Z is the parameter background_reject_factor. Brightfield images taken slightly out of focus may do better with with higher values (ie, higher values will better avoid spurious cells), but if the cell boundaries in the image are too narrow, a smaller value may be necessary–which might increase the level of background.

`tracking_comparison`
: Default: `0.20` Cell-ID attempts to track cells over time. The value of this parameter is the minimal fractional overlap between two cells in consecutive time points for them to be considered the same cell. The default value is 0.2. Also named \"I_over_U_for_match\" in CellID's cell.c and segment.c files.

`align_individual_cells`
: Default: `F` Allow wiggling between the brightfield and fluorescence images.

`align_fl_to_bf`
: Default: `F` Frame alignment. Cell-ID can perform image registrations, moving the the frame in XY to align it to a reference image. If "align FL to BF" is selected the bright field image is used as reference. If "align FL to first" is selected the first fluorescence image is used as reference. These options are especially useful when sampling different positions, as repositioning of the microscope stage might introduce some displacement between consecutive images.

`align_fl_to_first`
: Default: `F` To-do: document or link to explanation.

`image_type`
: Default: `"brightfield"` To-do: document or link to explanation.

`bf_fl_mapping`
: Default: `"list"` Possible values: "list", "time". "bf_fl_mapping" option description (guessed from code, mask_mod branch). The mapping between brightfield and fluorescence images can be made by acquisition time, or derived from the order in the list of paths passed as command line options "-b" and "-f" to cell. If the order is by "list", then the paths must be grouped and ordered first by t.frame (ascending) and then by channel. If the order is by "time", cell derives the BF-FL mapping from the acquisition time in the TIFF metadata.

`treat_brightfield_as_fluorescence_also`
: Default: `F` Calculate all the fluorescence images variables on the bright field image as if it were a fluorescence image. This is potentially a good idea since it allows a good way to reject spurious cells. For example, the average value of the boundary pixels in good cells will be lower than the background level, but not so for spurious cells, etc.

Details

Documentation for each parameter can be found at: https://github.com/darksideoftheshmoo/cellID-linux#parameters

Boolean values are for "flag" type parameters which enable a feature when present (eg. "align_fl_to_bf"), or, if absent, indicate default behavior.

Other parameters have values which must end up separated from names by a space " " in the parameters.txt file format that Cell-ID uses:

```
max_split_over_minor 0.5
max_dist_over_waist 8
max_pixels_per_cell 2000
min_pixels_per_cell 75
background_reject_factor 0.75
tracking_comparison 0.2
align_fl_to_bf
image_type brightfield
bf_fl_mapping list
```

Value

A nice list of named parameters, input for `parameters_write`.

See Also

parameters_write, arguments

`parameters_write` *Write parameters to a [temporary] file*

Description

Parses a `parameters.list` object from `parameters_default`, and writes its contents to a Cell-ID friendly plain text file.

Usage

```
parameters_write(
  parameters.list = rcell2.cellid::parameters_default(),
  param.dir = base::tempdir(),
  param.file = NULL
)
```

Arguments

`parameters.list`	a parameters list for Cell-ID (like one from parameters_default)
`param.dir`	directory where parameter files will be written.
`param.file`	a file name for the parameters file.

Value

A path to the text file where parameters where written.

See Also

parameters_default, arguments

rcell2.cellid *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 cellMagick package provides three categories of important functions: cellMagick, tidyCell and shinyCell.

tidyCell functions

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

cellMagick functions

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. Requirement of the imagemagick library might bother some, so this awesome feature is optional.

shinyCell functions

R-Shiny based graphical interface to filter cells by arbitrary variables, and inspect and annotate cells. It should not be required that the data comes from images processed by CellID.

See Also

`magick`

rename_mda *Image file renamer for Metamorph MDA*

Description

MDA: "Multi dimensional acquisition" app in Metamorph.

Usage

```
rename_mda(
  images.path,
  rename.path = NULL,
  rename.function = file.symlink,
  identifier.pattern = ".*_w(\\d).*_s(\\d{1,2})_t(\\d{1,2}).TIF$",
  identifier.info = c("ch", Position = "pos", time = "t.frame"),
  channel.maping.df = data.frame(ch = 1:3, ch.name = c("BF", "YFP", "TFP")),
```

```
  file.ext = ".tif",
  skip.thumbs.pat = ".*thumb.*",
  cleanup.first = F,
  ...
)
```

Arguments

images.path	Path to the directory containing the images output by Meramorph MDA.
rename.path	Path to the target directory. If NULL (the default) images are sent to a "renamed" subdirectory of images.path.
rename.function	Either file.copy, file.symlink or a similar function.
identifier.pattern	Regex defining gropus for each part of the name.
identifier.info	Character vector with strings "pos", "t.frame", and "ch" (channel), in the same order in which they appear in the identifier.pattern. Names in this vector are prefixed to the identifier in the final file name (for example, by default, "Position" is prepended to the position number; but channel has no prefix).
channel.maping.df	A dataframe with two columns: "ch" holding the original channel names in the source files, and "ch.name" with the new names for each channel.
file.ext	File extension to use in the final file name, such as: ".tif".
skip.thumbs.pat	A regex pattern to filter files. Convenient if the MDA output thumbnails for each image. Set to NULL to disable.
cleanup.first	Set to TRUE to remove all files within the rename.path directory. FALSE by default.
...	Further arguments passed onto rename.function.

Details

Uses regex groups to extract channel, position and time information from file names, and uses it to stitch new and friendlyer names. These are used to copy or link image files to a target directory.

For example, far1_rtcc_exp16_thumb_w1LED-BF--YFPcube--cam_s17_t35.TIF can be converted to BF_Position17_time35.tif.

The identifier.pattern is a key parameter. There must be three groups, one for each of the three information types: channel, position and time. The defaults are useful for a file name such as far1_rtcc_exp16_thumb_w1LED-BF--YFPcube--cam_s17_t35.TIF, in which the channel is identified by a "w", position by an "s", and time by a "t".

The order in which this information appears in the file name is specified in identifier.info. If you wish to add a prefix to each field in the final file name, name the elements in this vector. For example, the default c("ch",Position="pos",time="t.frame") indicates that channel has no prefix, the "pos" field will be prefixed by "Position", and the "t.frame" field will be prefixed by "time". Then, for example, a new file name could look like this: BF_Position1_time3.tif.

Channel names will be translated according to the rows in channel.dict (see the parameter's description). These are easily adaptable to other use cases, for example you may change channel.dict to include more, less or other channels, in whatever order. Note that the values in the ch column

must exactly match the strings captured by the corresponding capture group in `identifier.pattern`. For example, the channel in the original file names may be integers from 1 to 3, which are captured and matched with dplyr's left_join to the `channel.dict` data frame. Then, the value in `ch.name` is used to build the final file name.

Limitations: In the original file names, the identifiers for each field can only be integers.

examples images.path <- "~/Projects/PhD/data/uscope/multidimensional_exp-20211126-Far1NG-wt_y_dKar4/" rename_mda(images.path, rename.function = file.copy)

Value

Invisibly returns a list with the rename.path (output directory), and a data.frame with the output from the renaming function (see the `rename.function` parameter's description) and name conversions.

Index