Package 'islify'

July 23, 2025

Title Automatic scoring and classification of cell-based assay images

Version 1.1.0

Date 2025-04-14

biocViews Software,CellBasedAssays,BiomedicalInformatics,FeatureExtraction, Visualization,Pathways,Classification

Description This software is meant to be used for classification of images of cell-based assays for neuronal surface autoantibody detection or similar techniques. It takes imaging files as input and creates a composite score from these, that for example can be used to classify samples as negative or positive for a certain antibody-specificity. The reason for its name is that I during its creation have thought about the individual picture as an archielago where we with different filters control the water level as well as ground characteristica, thereby finding islands of interest.

License GPL-3

URL https://github.com/Bioconductor/islify

BugReports https://github.com/Bioconductor/islify/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.5)

Imports autothresholdr (>= 1.4.2), Matrix (>= 1.6.1), RBioFormats (>= 1.0.0), tiff (>= 0.1.12), png (>= 0.1.8), dbscan (>= 1.1.12), abind (>= 1.4.8), methods (>= 4.3.3), stats (>= 4.3.3)

Suggests knitr, rmarkdown, testthat, BiocStyle

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/islify

git_branch devel

git_last_commit f430e18

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-23

Author Jakob Theorell [aut, cre, fnd] (ORCID: <https://orcid.org/0000-0001-8752-3151>, Funding provided by the Swedish Wenner-Gren Foundations)

Maintainer Jakob Theorell <jakob.theorell@ki.se>

Contents

`islify-package` *Automatic scoring and classification of cell-based assay images*

Description

This software is meant to be used for classification of images of cell-based assays for neuronal surface autoantibody detection or similar techniques. It takes imaging files as input and creates a composite score from these, that for example can be used to classify samples as negative or positive for a certain antibody-specificity. The reason for its name is that I during its creation have envisioned the individual picture as an archielago where we with different filters control the water level, and thereby define islands with differing sizes and characteristics.

Author(s)

Maintainer: Jakob Theorell <jakob.theorell@ki.se>

See Also

Useful links:

- https://github.com/Bioconductor/islify
- Report bugs at https://github.com/Bioconductor/islify/issues

---

`getIntensityCutoff` *Identification of an intensity cutoff for a whole experiment*

Description

With this function, the otherwise integrated intensity cutoff is externalised. This is only useful in situations where the original values from the imaging have not been perturbed, so that the individual frames have different max values, and are not all compressed to a range from 0 to 1. NB! This requirement is not fulfilled if the files are exported as RGB compliant, as the RGB format always ranges from 0 to 1.

Usage

```
getIntensityCutoff(
  imgDirs,
  frameNum,
  numOfImgs = "All",
  numPix = "All",
  threshold_method = "Triangle",
  ignore_white = FALSE
)
```

Arguments

imgDirs	A vector or list of pathways, including filenames, to the images to be analysed, e.g. "Raw_images/Positive_ctrl.nd2". Formats that are currently supported are nd2, czi, tiff, png or lists of images, in the form of three-dimensional arrays, where each layer in the third dimension represents a color. nd2, czi or non-normalised, integer TIFF are clearly preferable for memory and resolution purposes.
frameNum	This identifies which of the frames in the file that contains the information about the autoantibody binding.
numOfImgs	If the provided files are nd2 format, they can contain multiple files. In this case, this flag can be used to restrict the number of used images.
numPix	If the frames are very large, this can be used to reduce the computational burden. For reproducibility reasons, it might be clever to run this multiple times, in this case, to bootstrap, or alternatively to set a seed before starting.
threshold_method	The method used for thresholding. Available alternatives are the same as for the auto_thresh function. The default "Triangle" method is in no way the only option, but it seems to perform reasonably well under many circumstances.
ignore_white	This is passed on to the auto_thresh function. If a value, for example inherited from getQuantileIntensities, then the values above this level will not be considered when identifying the background threshold. Remedies some of the variance between negative and positive control samples.

Value

An intensity cutoff value for the chosen color.

See Also

auto_thresh

Examples

```
# Load example data and run the function:
data(negImage)
data(posImage)
getIntensityCutoff(imgDirs = list(negImage, posImage), frameNum = 1)
```

`getQuantileIntensities` *Get information about the quantiles for the intensity values for all colors in an image*

Description

With this function, the user can get an overview of how the values for the intensities in a file or set of files are distributed. This is useful mainly to set reasonable cutoff values for the saveImage function, but might also be good for sanity checking the getIntensityCutoff results.

Usage

```
getQuantileIntensities(imgDirs, quantiles = c(0.99))
```

Arguments

`imgDirs`	A vector or list of pathways, including filenames, to the images to be analysed, e.g. "Raw_images/Positive_ctrl.nd2". Formats that are currently supported are nd2, czi, tiff, png or lists of images, in the form of three-dimensional arrays, where each layer in the third dimension represents a color. nd2, czi or non-normalised, integer TIFF are clearly preferable for memory and resolution purposes.
`quantiles`	Which quantiles should be returned? Values between and including 0 and 1 are accepted.

Value

A list of matrices with one row per color in the data and with thirteen columns of intensities from percentile 0 through to percentile 100.

Examples

```
# Load example data and run the function:
data(negImage)
getQuantileIntensities(list(negImage))
```

`getSizeCutoff` *Identification of a suitable size cutoff for cells*

Description

This function is needed in situations where the cell size is not known. What it does is that it identifies the median cell nucleus size in a provided file. In this case, therefore, it is important that the frame number identifies a frame with nuclear staining, such as DAPI or similar.

Usage

```
getSizeCutoff(
  imgDirs,
  frameNum,
  noiseThreshold = 10,
  numPix = "All",
  numOfImgs = "All"
)
```

Arguments

imgDirs	A vector or list of pathways, including filenames, to the images to be analysed, e.g. "Raw_images/Positive_ctrl.nd2". Formats that are currently supported are nd2, czi, tiff, png or lists of images, in the form of three-dimensional arrays, where each layer in the third dimension represents a color. nd2, czi or non-normalised, integer TIFF are clearly preferable for memory and resolution purposes.
frameNum	This identifies which of the frames in the file that contains the information about the nuclear staining. In RGB-compliant files, this is often number 3.
noiseThreshold	This decides under what size in pixels that an islet is considered noise. Defaults to 10 pixels.
numPix	If the frames are very large, this can be used to reduce the computational burden. For reproducibility reasons, it might be clever to run this multiple times, in this case, to bootstrap, or alternatively to set a seed before starting.
numOfImgs	If the provided files are nd2 format, they can contain multiple files. In this case, this flag can be used to restrict the number of used images.

Value

A size cutoff value in pixels, which corresponds to the median of the island diameters above the noise threshold.

Examples

```
# Load example data and run the function:
data(negImage)
getSizeCutoff(imgDirs = list(negImage), frameNum = 3)
```

---

islify *Generation of islands and island derived statistics*

Description

This function is the central umbrella function that takes a list of image files or image file directories and generates a list of statistics for each of the files.

Usage

```
islify(
  imgDirs,
  imgNames,
  frameNumFocus,
  frameNumReference = FALSE,
  frameNumNuclei = FALSE,
  sizeCutoff,
  intensityCutoffFocus = TRUE,
  intensityCutoffReference = TRUE,
  intensityCutoffNuclei = TRUE,
  threshold_method = "Triangle",
  ignore_white = FALSE,
  ringFrac = 0,
  flatFrac = 0,
  truncLim = "max",
  reportIntensity = FALSE,
  diagnoImgs = TRUE,
  outDir = ".",
  highNoise = FALSE,
  numPix = "All",
  numOfImgs = "All"
)
```

Arguments

imgDirs	A vector or list of pathways, including filenames, to the images to be analysed, e.g. "Raw_images/Positive_ctrl.nd2". Formats that are currently supported are nd2, czi, tiff, png or lists of images, in the form of three-dimensional arrays, where each layer in the third dimension represents a color. nd2, czi or non-normalised, integer TIFF are clearly preferable for memory and resolution purposes.
imgNames	The names of the images. This argument is used both for naming of the rows in the output matrix, but also for naming of the images if such are generated using the diagnoImgs flag.
frameNumFocus	This identifies which of the frames in the files that contains the information that should be measured primarily, such as IgG binding in the case of autoantibody screening.
frameNumReference	Optionally, this identifies a frame in the files that can be used to focus the analysis to certain regions. It could be that only areas with GFP expression can contain meaningful expression of the marker of primary interest, and then only these areas will be considered.
frameNumNuclei	If no reference is present, but a nuclear staining is, this parameter can be used to identify which approximate surface area for the individual picture that could under optimal circumstances be covered with antibodies. As the nuclei are smaller than the cells, it is theoretically possible to get a report back of a fraction exceeding 1, but this is an unlikely scenario, as 100 percent transfection efficiency is seldomly obtained.
sizeCutoff	The size of a typical cell nucleus. This is used to identify structures of a size equal to or larger than this cell threshold. Smaller objects that are ring-shaped

will be excluded with this cutoff, which reduces noise considerably, but the threshold should not be set too high as it then risks leading to the exclusion of true positive cells. This value can be generated using the getSizeCutoff function.

intensityCutoffFocus
: If a series of images have been generated with the same settings and the original dynamic range of these images has been retained, it is preferred to calculate the intensity cutoff before and provide it here, but the value can also be calculated within the function. This parameter therefore takes either a value or TRUE, in which each individual frame gets a separate intensity cutoff. See getIntensityCutoff. This latter option is suitable when it is unclear if all samples in a series have been generated with the same imaging settings, or, more commonly, if RGB-compliant files are used as input, as these are always normalised to a range from 0 to 1, and thus, the background in a negative frame will have considerably higher values than in a positive sample, simply because the rate from the lowest to the highest signal is more stretched out in a case with noise only.

intensityCutoffReference
: Same as above but for the reference frame, if present.

intensityCutoffNuclei
: Same as above but for the nuclei frame, if present.

threshold_method
: The method used for thresholding. Available alternatives are the same as for the auto_thresh function. The default "Triangle" method is in no way the only option, but it seems to perform reasonably well under many circumstances.

ignore_white
: This is passed on to the auto_thresh function. If a value, for example inherited from getQuantileIntensities, then the values above this level will not be considered when identifying the background threshold. Remedies some of the variance between negative and positive control samples.

ringFrac
: The positive selection filter. After a crude minimal diameter criterion, checks if the island retains its width even if pixels below this fraction of the 99th to 1th percentile range of the intensity of the island are removed. The point is to actively seek truly poiitive islands as surface-stained cells have a heterogeneous, ring-shaped structure. This is not used in the default setting.

flatFrac
: The negative selection filter, meant to get rid of dead cells that generally have a very homogenous staining pattern. If the median absolute deviation is lower than the flatFrac of the range from the 99th to the 1st percentile of the island, then it is considered flat and excluded. Low values, in the range of 0.001 to 0.1 are recommended.

truncLim
: In extreme cases, such as with the NMDA-R assay, it might be helpful to truncated the most highly stained cells, as they counterintuirively tend to be part of the background. The level for this truncation limit is set here. THis also affects the diagnoImgs if these are present. It can be helpful to consult the getQuantileIntensities function to identify suitable values here. In most instances, it is however advisable to stick to the default max.

reportIntensity
: Should the sum of the intensities for all the surviving islands be returned? Default is FALSE.

diagnoImgs
: Should a images delimiting the islands that have been selected be returned?

outDir
: The directory that the diagnoImages should be saved in. Only used together with diagnoImgs = TRUE.

highNoise	Is the assay especially noisy, with very high and individual background? If so, a second run of noise reduction can be used to increase signal-to-noise ratio. This is often but not always needed for fixed cell-based assays, e.g.
numPix	If the frames are very large, this can be used to reduce the computational burden. NB! If this command is used, then the images will not be identical from round to round, and will not be comparable to the output of other functions, as a random subset of the pictures are used.
numOfImgs	If the provided files are nd2 format, they can contain multiple files. In this case, this flag can be used to restrict the number of used images.

Value

A data frame with statistics for the individual images: #'

intensityCutoff_focus The intensity cutoff for the individual file. If the reference is included, it will have a separate value.

fractionOfAll_focus The fraction of the total number of pixels that pass the filters. If the reference is included, it will have a separate value.

fractionOfRed_focus If a reference is included, then fractionOfAll_focus/fractionOfAll_ref is also saved.

secondIntensityCutoff_focus If highNoise is TRUE, then the second noise filter is included here. If the reference is included, it will have a separate value.

Optionally, if diagnoImgs = TRUE, a directory containing images identifying the regions that have been identified as cells of interest is exported.

See Also

auto_thresh

getIntensityCutoff, getSizeCutoff auto_thresh

Examples

```
# Retrieve the example data
data(posImage)
data(negImage)

# First, establish the average nuclear size. Here, tje nuclei from one image
# is generally enough.
sizeCutoff <- getSizeCutoff(imgDirs = list(negImage), frameNum = 3)

# We can run this algorithm in two ways: either only using the blue for size
# and red for the rest, or we can also incorporate the green color, saying
# e.g.that we only are interested in IgG expression that colocalizes with
# GFP expression. We start with the simpler case.

intensityCutoffRed <- getIntensityCutoff(
    imgDirs = list(negImage, posImage),
    frameNum = 1
)
result <- islify(
    imgDirs = list(negImage, posImage),
    imgNames = c("Neg", "Pos"),
    frameNumFocus = 1,
```

```
    sizeCutoff = sizeCutoff,
    intensityCutoffFocus =
        intensityCutoffRed,
    diagnoImgs = FALSE
)

# As can be noted above, diagnoImgs are set to FALSE, which means that the
# output will be restricted to statistics only. If set to TRUE, then an
# image showing the selected areas in red on a black-and-white background
# are generated. In that case, a directory, outDir, also needs to be
# specified.

# Now comes the more complex useage, where the reference color is also
# integrated. Here, to save time, only the negative image is used for
# reference as both are expected to have similar GFP intensities.

intensityCutoffGreen <- getIntensityCutoff(
    imgDirs = list(negImage),
    frameNum = 2
)

resultWithGreen <- islify(
    imgDirs = list(negImage, posImage),
    imgNames = c("Neg", "Pos"),
    frameNumFocus = 1,
    frameNumReference = 2,
    sizeCutoff = sizeCutoff,
    intensityCutoffFocus =
        intensityCutoffRed,
    intensityCutoffReference =
        intensityCutoffGreen,
    diagnoImgs = FALSE
)
```

`negImage` *A few CASPR2-negative cells*

Description

This is a negative control image, used for help functions and vignettes. It is a real-life image showing CASPR2 expressing HEK293-MSC cells co-incubated with CASPR2-IgG. It was generated by Dr Amanda Freitas Huhtamäki during her postdoctoral period in the Jakob Theorell team, Center for Infectious Medicine, Department of Medicine Huddinge, in the Live Cell Imaging core facility, Huddinge, both Karolinska Institutet, Sweden.

Usage

```
data("negImage")
```

Format

An a x b x c array representing red (IgG), green (GFP) and blue (DAPI) colors.

posImage *A few CASPR2-positive cells*

Description

This is a positive control image, used for help functions and vignettes. It is a real-life image showing CASPR2 expressing HEK293-MSC cells co-incubated with CASPR2-IgG. It was generated by Dr Amanda Freitas Huhtamäki during her postdoctoral period in the Jakob Theorell team, Center for Infectious Medicine, Department of Medicine Huddinge, in the Live Cell Imaging core facility, Huddinge, both Karolinska Institutet, Sweden.

Usage

```
data("posImage")
```

Format

An a x b x c array representing red (IgG), green (GFP) and blue (DAPI) colors.

saveImage *Saving individually coloured png images from raw image input*

Description

This function simply generates pngs from raw image files, such as .nd2.

Usage

```
saveImage(
  imgDirs,
  imgNames,
  frameNums = "All",
  frameCols,
  truncTo = "max",
  outDir = ".",
  numOfImgs = "All",
  numPix = "All",
  saveImages = TRUE
)
```

Arguments

imgDirs	The directory containing the image files. nd2, czi, tiff and png formats or a list of three-dimensional arrays, where each layer in the third dimension represents a color are currently supported. nd2, czi or non-normalised, integer TIFF are clearly preferable.
imgNames	The names of the images. This argument is used both for naming of the rows in the output matrix, but also for naming of the images if such are generated using the diagnoImgs flag.

frameNums	This identifies which of the frames in the files that should be plotted. Defaults to "All", which then also generates a merge.
frameCols	Which color does the frame/frames in question have? Strangely, in some file formats such as nd2, it varies - not all are "R-G-B", but "G-R-B" also occurs. Can take the values "R"/"Red", "G"/"Green", and "B"/"Blue". Should be a vector the same length as frameNums. If nothing is provided, it defaults to only saving the merge.
truncTo	Above which value should the data be truncated? This argument has two possible inputs: a numeric value or "max". Vectors of individual values, the same length as frameCols are also accepted. The value option is often the most suitable, but requires some knowledge of the range of values expected in the file. For this, the getQuantileIntensities function can be useful to provide insights to provide reasonable values. If this value is set to a lower value than the true max, the data will be truncated and a warning thrown.
outDir	The directory that the images should be saved in. Subfolders are created within this directory for each color.
numOfImgs	If the provided files are nd2 format, they can contain multiple files. In this case, this flag can be used to restrict the number of used images.
numPix	If the frames are very large, this can be used to reduce the computational burden. For reproducibility reasons, it might be clever to run this multiple times, in this case, to bootstrap, or alternatively to set a seed before starting. NB! If this command is used, then the images will not be identical from round to round, and will not be comparable to the output of other functions, as a random subset of the pictures are used.
saveImages	Should images be saved? This of course seems like a rather pointless command for a function meant to save pictures, but it is there for development reasons. Don't use it!

Value

Named png images that are scaled to the max allowed value in the raw image file and coloured according to frameCol.

See Also

getQuantileIntensities

Examples

```
# Retrieve the example data
data(posImage)
data(negImage)

# To use this function, the getQuantileIntensities function is very helpful,
# as it can be used to define reasonable truncTo values. It is of course
# important to use a strongly stained sample in this case, to make sure that
# the positive samples in the series are not overly truncated, reducing the
# visual signal-to-noise ratio.
posQuants <- getQuantileIntensities(list(posImage), quantiles = 0.99)
# We will use the 99th percentile
posQuants
#    Percent_99
```

```
# 1  0.3681333
# 2  0.2476340
# 3  0.3794010

# And now to the function. NB! In this case, we are using the last
# "saveImages = FALSE" flag, which means that nothing is produced! Remove it
# to use the function.
saveImage(list(posImage, negImage), c("pos", "neg"),
    frameNums = "All",
    frameCols = c("R", "G", "B"), truncTo = posQuants[[1]][, "Percent_99"],
    outDir = ".", saveImages = FALSE
)
```

`zprime` *Compute the Z'-factor quality score*

Description

This function is derived in its entirety from the deprecated imageHTS package maintained by Gregoire Pau <gregoire.pau at embl.de>. All the documentation below as well as the code is identical to the original text. Only the simplified example is new. So kudos to the original authors! This re-publication is just there as no alternative has seemingly been available in R for years and it is needed for the downstream analyses with islify data.

Usage

```
zprime(a, b, method = c("mahalanobis", "robust", "fixsd", "original"))
```

Arguments

`a`	A vector or matrix of control features. For example, positive controls.
`b`	A vector or matrix of different control values. For example negative controls.
`method`	A character vector, indicating which method should be used to compute the Z'-factor. Default is mahalanobis. See details.

Details

The Z'-factor is a popular metric measuring the separation of control features in high-throughput screens. The original paper describing the Z'-factor is Zhang, 1999, J Biomol Screen.

#' Several univariate Z'-factor scores exist. The original Z'-factor from Zhang, 1999 is computed by Z' = 1 - 3*(sd(a)+sd(b))/abs(mean(a)-mean(b)). A more rigorous definition of the score, implemented by the method fixsd is given by Z' = 1 - 3*sqrt(var(a)+var(b))/abs(mean(a)-mean(b)), where the pooled standard deviation is computed by the square root of the sum of the control variances. A robust method, less sensitive to outliers, is computed by the relation Z' = 1-3*(mad(a)+mad(b))/abs(median(a)-median(b)) where the control dispersions are computed with the mad and the control locations with the median.

A multivariate extension of the Z'-factor score can be designed by linearly transforming the multivariate data to one dimension and computing the standard (here, fixsd) Z'-factor. It can be shown that the linear transform that maximizes the score is the LDA. Moreover, one can demonstrate that the resulting Z'-factor score is equivalent of computing Z' = 1 - 3/dMaha(mu_a, mu_b, Sigma_a + Sigma_b) where dMaha is the Mahalanobis distance.

Value

The z-prime value for the data in question.

References

J. H. Zhang, T. D. Chung, K. R. Oldenburg. A Simple Statistical Parameter for Use in Evaluation and Validation of High Throughput Screening Assays. J Biomol Screening, 1999.

Examples

```
#Generate some data
a <- rnorm(100, mean = 0, sd = 1)
b <- rnorm(100, mean = 7, sd = 1)

zprime(a, b)
```

Index