imageHTS

October 25, 2011

`collectCellFeatures` *Collect cell features.*

Description

Collect cell features from a given set of cells.

Usage

```
collectCellFeatures(x, uname, spot=NULL, id=NULL, access='cache')
```

Arguments

`x`	An imageHTS object.
`uname`	a character vector, containing the well names from where to collect the cell features. See `getUnames` for details.
`spot`	An optional numeric vector indicating the spots from where to collect the cell features. If missing, all spots are considered. The length of the vector must be the same as `uname`.
`id`	An optional numeric vector indicating the cell ids. If missing, all cells are considered. The length of the vector must be the same as `uname`.
`access`	A character string indicating how to access the data. Valid values are `local`, `server` and `cache`, the default. See `fileHTS` for details.

Details

Contrary to `readHTS`, `collectCellFeatures` collects cell features through multiple wells. Output data frame contains the columns `uname`, `spot` and `id`.

Value

A data frame containing the cell features.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

extractFeatures

Examples

```
## see extractFeatures for an example of collectCellFeatures
## example(extractFeatures)
```

---

countObjects *Count the number of objects in a segmented image*

Description

Count the number of objects in a segmented image.

Usage

```
countObjects(cseg)
```

Arguments

cseg	An image containing the objects.

Value

An integer indicating the number of objects in the cell mask.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

segmentWells

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')

## segment one well
uname = getUnames(x, content='rluc')[1]
z = segmentWells(x, uname=uname, segmentationPar='conf/segmentationpar.txt', writeData=FA
n = countObjects(z$cseg)
cat('number of cells=', n, '\n')
```

`extractFeatures` *Extract features from segmented images.*

Description

Extract features from segmented images.

Usage

```
extractFeatures(x, uname, featurePar, access='cache')
```

Arguments

`x`	An imageHTS object.
`uname`	a character vector, containing the well names to segment. See `getUnames` for details.
`featurePar`	a character string, indicating the filename containing the feature extraction parameters.
`access`	A character string indicating how to access the data. Valid values are `local`, `server` and `cache`, the default. See `fileHTS` for details.

Details

`extractFeatures` reads the DCF segmentation parameters file pointed by `featurePar`. The file must contain the field `extractfeatures.method` that indicates which core feature extraction function to use. The function takes two arguments: `cal`, the calibrated image to extract the features from, and `seg`, a list of two images, `cseg` and `nseg`, which contains the cell and nucleus segmentation masks, respectively. The function returns a matrix of features. See `getCellFtrsATH` for example.

For each well, `extractFeatures` writes a `ftrs` segmentation data file that contains the cell features. Use `readHTS` or `collectCellFeatures` to get the cell features after extraction.

Value

None.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

`getCellFtrsATH`, `fileHTS`, `collectCellFeatures`

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')

## segment one well
uname = getUnames(x, content='rluc')[1]
segmentWells(x, uname=uname, segmentationPar='conf/segmentationpar.txt')

## extract features from the well
extractFeatures(x, uname=uname, 'conf/featurepar.txt')

## read the feature file using readHTS
y = readHTS(x, type='ftrs', uname=uname)

## get features using collectCellFeatures
y = collectCellFeatures(x, uname=uname)
```

`fileHTS` *Get access to screen data files*

Description

`fileHTS` builds the path or URL pointing to a screen data file. `readHTS` reads the file pointed by `fileHTS`.

Usage

```
fileHTS(x, type, ..., createPath=FALSE, access='cache')
readHTS(x, type, ..., access='cache', format=NULL)
```

Arguments

`x`	An imageHTS object.
`type`	A character vector, indicating the requested file type. See Details.
`...`	Optional arguments. See Details.
`createPath`	A logical value specifying if the directories along the path should be created. Default is `FALSE`.
`access`	A character string indicating how to access the data. Valid values are `cache` (the default), `local` and `server`. See Details.
`format`	An optional character string indicating the format of the designated file. Valid formats are `tab`, a tab-separated file with headers; `rda`, a R data file; `dcf`, an imageHTS DCF configuration file.

Details

In imageHTS, all screen data files are accessed through the function `fileHTS`. Screen data can be accessed at two locations: the `localPath`, which is local writable directory, and `serverURL`, which is a server URL. `localPath` and `serverURL` are set during the instantiation of the `imageHTS` object with `parseImageConf`.

If `access` equals `local`, `fileHTS` returns a local path pointing to the requested file. If `server`, `fileHTS` returns a server URL pointing to the requested file. If `cache` (the default), `fileHTS` tests if the file is present at the local path. If not, `fileHTS` tries to download the file from the server and copies it in the local path. `fileHTS` always returns a local path if `access` equals `cache`.

A file is designated by its `type` and optional arguments. Known file types are:

- `file`: general-purpose file. The character string `filename` indicates its path, relative to the project directory. File format is unspecified.
- `source`: source image, designated by the character string `uname` and the numeric `channel`. File format is unspecified.
- `cal`: calibrated image, designated by the character string `uname`. The file is a R data file which contains an `EBImage` image object.
- `seg`: segmentation data, designated by the character string `uname`. The file is a R data file which contains a list of two `EBImage` image objects: `cseg` containing the cell mask and `nseg` the nucleus mask.
- `ftrs`: cell features, designated by the character string `uname`. The file is a tab-separated file which contains the cell features.
- `clabels`: cell labels, designated by the character string `uname`. The file is a tab-separated file which contains the cell labels.
- `viewfull`: calibrated JPEG image, designated by the character string `uname`. The file is a JPEG image.
- `viewunmonted`: spot-untiled calibrated JPEG image, designated by the character string `uname` and the spot number `spot`. The file is a JPEG image.
- `viewseg`: segmented JPEG image, designated by the character string `uname`. The file is a JPEG image.
- `viewthumb`: thumbnail JPEG image, designated by the character string `uname`. The file is a JPEG image.

`readHTS` reads and returns the corresponding file. `format` must be specified if `type` is `file` or `source`.

Value

`fileHTS` returns a character vector containing the path or URL of the requested file. `readHTS` returns the content of the requested file.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

`parseImageConf`, `getUnames`

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)

## fileHTS and readHTS examples
fileHTS(x, 'file', filename='conf/imageconf.txt')
fileHTS(x, 'source', uname='001-01-C05', channel=1)
readHTS(x, 'file', filename='conf/featurepar.txt', format='dcf')

## initialize imageHTS object using the remote kimorph screen
local = tempdir()
server = 'http://www.ebi.ac.uk/~gpau/imageHTS/screens/kimorph'
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)

## get cell features for well '002-02-D06'
f = readHTS(x, 'ftrs', uname='002-02-D06')
cat('nb cells=', nrow(f), '\n')
```

---

getImageConf *Get the imageHTS configuration*

Description

Get the imageHTS configuration from an imageHTS object.

Usage

```
getImageConf(x)
```

Arguments

x	An imageHTS object.

Details

See the documentation of parseImageConf for details.

Value

A list containing the imageHTS configuration attributes.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

parseImageConf

Examples

```
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
getImageConf(x)
```

---

getUnames *Get well unique names*

Description

Build valid well unique names, according to the screen layout and to input filter arguments. Each well in the screen has an unique name, based on its plate, replicate, row and column indices.

Usage

```
getUnames(x, plate, replicate, row, col, content)
```

Arguments

x	An imageHTS object.
plate	An optional numeric vector indicating the plate indices to keep.
replicate	An optional numeric vector indicating the replicate indices to keep.
row	An optional numeric vector indicating the row indices to keep.
col	An optional numeric vector indicating the column indices to keep.
content	An optional character string indicating the well content type to keep.

Details

Well content types are defined in the plateconf.txt configuration file. See getWellFeatures for details.

Well unique names can be also built and manipulated using prw2uname and uname2prw.

Value

A character vector containing a set of well unique names that are compatible with the input filter arguments.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

prw2uname, uname2prw, getWellFeatures

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt',
'conf/screenlog.txt')

getUnames(x, col=2)
getUnames(x, replicate=1, col=3)
getUnames(x, content='sample') ## get 'sample' wells
setdiff(getUnames(x), getUnames(x, content='empty')) ## get non-empty wells
```

`getWellFeatures` *Get well metadata, features and annotation information*

Description

Get well metadata, features and annotation information.

Usage

```
getWellFeatures(x, uname, feature=TRUE)
```

Arguments

`x`	An imageHTS object.
`uname`	a character vector, containing well names. See `getUnames` for details.
`feature`	A character vectors containing the requested features. Default is `TRUE`, returning all well features.

Details

`getWellFeatures` return `fData`, the well features loaded during the `configure` and `annotate` steps. Features include: `controlStatus`, the well content status derived from \'plateconf.txt\'.

Value

Returns a data frame containing well features.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

`fData`, `configure`, `annotate`

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')
x = annotate(x, 'conf/annotation.txt')

## select non-empty wells
unames = setdiff(getUnames(x), getUnames(x, content='empty'))
getWellFeatures(x, unames)
```

---

highlightSegmentation

Highlight segmented objects in an image

Description

Highlight segmented objects in an image.

Usage

```
highlightSegmentation(cal, nseg=NULL, cseg=NULL, thick=FALSE)
```

Arguments

cal	An EBImage image object containing the original image.
nseg	An optional EBImage image object containing the nucleus mask.
cseg	An optional EBImage image object containing the cell mask.
thick	A logical indicating whether thick borders (useful for print) are required. Default is FALSE.

Details

highlightSegmentation highlights nuclei and cells by outlining them in yellow and magenta.

Value

An EBImage image containing the annotated image.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

segmentWells

Examples

```
## see segmentWells for an example of highlightSegmentation
## example(segmentWells)
```

`imageHTS` *Package overview*

Description

imageHTS is an R package dedicated to the analysis of high-throughput microscopy-based screens. The package provides a modular and extensible framework to segment cells, extract quantitative cell features, predict cell types and browse screen data through web interfaces. Designed to operate in distributed environments, imageHTS provides a standardized access to remote screen data, facilitating the dissemination of high-throughput microscopy-based screens.

Package content

The following function instantiates the imageHTS object.

- parseImageConf: instantiate an imageHTS object from a local or remote screen data repository

The following functions process, segment, quantify, summarize the well images.

- segmentWells: segment cells in well images
- extractFeatures: extract cell features from segmented images
- readLearnTS: train a cell classifier
- predictCellLabels: predict cell labels
- summarizeWells: summarize cell populations

The following functions provides means to display and inspect the screen data.

- installWebQuery: install the webQuery module
- popWebQuery: pop the webQuery module
- installCellPicker: install the cellPicker module
- popCellPicker: pop the cellPicker module
- segmentATH: segment cells stained for DNA, actin and tubulin
- getCellFtrsATH: extract features from cells stained for DNA, actin and tubulin

The following functions give access to the screen data.

- fileHTS: build the path to a screen data file
- readHTS: read a screen data file
- parseDCF: read a DCF configuration file
- collectCellFeatures: collect cell features
- getWellFeatures: get well metadata, features and annotation information

The following manipulate well unique names.

- getUnames: get well unique names

- prw2uname: convert a (plate, replicate, well) data frame in well unique names
- uname2prw: convert well unique names in a (plate, replicate, well) data frame
- rowcol2well: convert a (row, col) data frame in well names
- well2rowcol: convert well names in a (row, col) data frame
- well2wellid: convert well coordinates in numerical well identifiers

Miscellaneous functions.

- zprime: compute the Z'-factor quality score
- highlightSegmentation: highlight segmented objects in an image
- countObjects: count the number of objects in a segmented image
- getImageConf: get the imageHTS configuration

Authors

Gregoire Pau, <gregoire.pau@embl.de>, 2010

`installCellPicker` *Set up imageHTS web modules*

Description

`installCellPicker` and `installWebQuery` install the cellPicker and webQuery modules in the local project directory.

`cellPicker` is a web application that allows the interactive selection/annotation of cells within images using a point-and-click web interface.

`webQuery` is a web application that allows to query information from the screen by well and well annotation. The webQuery module requires a web server and PHP to be installed.

Usage

```
installCellPicker(x)
installWebQuery(x)
```

Arguments

`x`	An imageHTS object.

Details

After installation, the cellPicker module can be used with the function `popCellPicker`.

`x` must be annotated using `annotate` before using `installWebQuery`. After installation, the webQuery module can be used with the function `popWebQuery`.

Value

None.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

popCellPicker, popWebQuery

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = annotate(x, 'conf/annotation.txt')

installCellPicker(x)
installWebQuery(x)
```

---

makeCellHTS *Segmentation of yeast cells and ring-shaped objects.*

Description

makeCellHTS creates a cellHTS2 object.

Usage

```
makeCellHTS(x, profiles, measurementNames, name)
```

Arguments

x	An imageHTS object.
profiles	A data frame containing the phenotypic profiles. See Details.
measurementNames	An optional character vector containing the measurement names. If missing, column names of profiles are used.
name	An optional character string containing the name of the assay.

Details

profiles is a data frame containing the phenotypic profiles, usually returned by summarizeWells or readHTS. Since cellHTS2 cannot handle large report, the dimension of the profiles must be lower than 10. This is usually done by subsetting columns or by dimension reduction.

Value

Returns a cellHTS2 object.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

summarizeWells, installWebQuery

Examples

```
## Not run:
  ## initialize kimorph object
  localPath = file.path(tempdir(), 'kimorph')
  serverURL = 'http://www.ebi.ac.uk/~gpau/imageHTS/screens/kimorph'
  x = parseImageConf('conf/imageconf.txt', localPath=localPath, serverURL=serverURL)
  x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')
  x = annotate(x, 'conf/annotation.txt')

  ## get profiles
  profiles = readHTS(x, type='file', filename='data/profiles.tab', format='tab')

  ## prepare cellHTS2 report
  ft = c('med.c.t.m.int', 'med.c.g.ss', 'med.c.g.ec', 'med.n.h.m.int', 'med.c.a.m.int')
  measurementNames = c('tubulin intensity', 'cell size', 'cell eccentricity', 'dna inten
  y = makeCellHTS(x, profiles[,c('uname', ft)], measurementNames=measurementNames, name=
  pathConf = file.path(localPath, 'conf')
  y = configure(y, 'description.txt', 'plateconf.txt', 'screenlog.txt', path=pathConf)
  y = annotate(y, 'annotation.txt', path=pathConf)
  yn = normalizePlates(y, scale='multiplicative', log=FALSE,
  method='median', varianceAdjust='none')

  ## write cellHTS2 report
  se = getSettings()
  se$plateList$intensities$include = TRUE
  setSettings(se)
  writeReport(raw=y, normalized=yn, outdir='report-cellHTS2', force=TRUE)

## End(Not run)
```

---

parseDCF *Parse a DCF file*

Description

Parse a DCF file.

Usage

```
parseDCF(filename)
```

Arguments

filename	A character string containing the path to a DCF file.

Details

The DCF format is a simple text format where each line is a field of the form \'a: a_0, a_1, ..., a_n\': `a` is the field name and `ak` the k-th value of `a`. Field name is separated from values by a colon \':\'. Field values are separated from each other by a comma \',\'.

`readHTS` with format `dcf` is a higher-level function that parses a specific DCF file from a given imageHTS project.

Value

A list of character vector, containing the DCF read fields.

Author(s)

Gregoire Pau, `<gregoire.pau@embl.de>`, 2010

See Also

`readHTS`

Examples

```
imageconf = system.file('submorph/conf/imageconf.txt', package='imageHTS')
parseDCF(imageconf)
```

---

parseImageConf *Instantiate an imageHTS object*

Description

Instantiate an imageHTS object from a local or remote screen data repository.

Usage

```
parseImageConf(filename, localPath='myscreen', serverURL, access='cache')
```

Arguments

`filename`	A character string containing the name of the imageHTS configuration file. See Details.
`localPath`	A character string indicating the path to the project directory that contains (or will contain) the screen data. The directory must be writable and will be created if missing. All intermediate files created by imageHTS will be stored in this directory. Default is `myscreen`.
`serverURL`	An optional character string indicating the URL or path to the server from where to download the data, if not available in the local path.
`access`	A character string indicating how to access the data. Valid values are `cache` (the default), `local` and `server`. See `fileHTS` for details.

Details

parseImageConf gets access to the configuration file depending on the value of the access argument. If local, the file is loaded from the local path; if server, from the server and if cache parseImageConf tries to first load the file from the local path, and if not present, downloads it from the server. This dual repository feature is useful when screen images are stored in a different location from where they are analysed. See fileHTS for details.

The imageHTS configuration file is a DCF file which contains the following fields:

- AssayName: a character string containing the name of the assay
- SourceFilenamePattern: a character string containing the source image pattern path, relative to the local path. Special fields: {plate}, {replicate}, {row}, {col} and {channel} will be replaced by elements of the corresponding fields PlateNames, ReplicateNames, RowNames, ColNames and ChannelNames.
- PlateNames: a comma-separated character vector, containing the plate names, to replaced in the field {plate} of SourceFilenamePattern.
- ReplicateNames: same as PlateNames for the field {replicate}.
- RowNames: same as PlateNames for the field {row}.
- ColNames: same as PlateNames for the field {col}.
- ChannelNames: same as PlateNames for the field {channel}.
- Montage: an optional comma-separated vector of two integers. If source images contain assembled different spot images of the well, this vector contains the dimension of the montage.

Use the command getImageConf to retrieve the configuration file from an imageHTS object.

Value

An imageHTS object.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

imageHTS, fileHTS, configure, getImageConf

Examples

```
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
getImageConf(x)
```

popCellPicker *Pop up imageHTS web modules*

Description

Pop up the imageHTS web modules cellPicker and webQuery, using the web browser.

Usage

```
popCellPicker(x, uname, spot=NULL, access='server', browse=TRUE)
popWebQuery(x, access='server', browse=TRUE)
```

Arguments

x	An imageHTS object.
uname	A character vector, containing the well names to annotate. See getUnames for details.
spot	An optional numeric vector, containing the spot indexes of the wells to annotate. If missing, all spots are used.
access	A character string indicating how to access the data. Valid values are local, cache and server, the default. See fileHTS for details.
browse	A logical indicating whether the web browser should be loaded. Default is TRUE.

Details

cellPicker must be installed using installCellPicker before using popCellPicker on the local project directory. If present, the numeric vector spot must have the same length as uname.

webQuery must be installed using installWebQuery before using popWebQuery on the local project directory.

Value

A character string containing the URL to access the cellPicker or the webQuery web module.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

installCellPicker, installWebQuery, getUnames

Examples

```
## initialize imageHTS object using the remote kimorph screen
local = tempdir()
server = 'http://www.ebi.ac.uk/~gpau/imageHTS/screens/kimorph'
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)

if (interactive()) {
```

```
  popCellPicker(x, uname=c('002-02-A11', '001-01-C17'))
  popWebQuery(x)
}
```

`prw2uname` *Convert and parse well unique names*

Description

Functions to convert and parse well unique names.

Usage

```
prw2uname(plate, replicate, row, col, well)
uname2prw(uname)
rowcol2well(row, col)
well2rowcol(well)
well2wellid(row, col, direction='row', dim)
```

Arguments

`plate`	An numeric vector of plate indices or a list containing the numeric vectors `plate`, `replicate`, `row` and `col`.
`replicate`	A numeric vector of replicate indices.
`row`	A numeric vector of row indices.
`col`	A numeric vector of column indices
`well`	A character vector of well names.
`uname`	A character vector of well unique names.
`direction`	A character string containing the direction of the mapping. Valid values are `row` and `col`. Default is `row`, where the well identifier 2 points is mapped to well (1, 2).
`dim`	A numeric vector of length two, containing the dimensions (number of rows, number of columns) of a plate.

Details

In `prw2uname`, wells can be specified using the arguments `row` and `col` or using the argument `well`.

Value

`prw2uname` returns a character vector of well unique names. `uname2prw` returns a data frame containing the columns `plate`, `replicate`, `row` and `col`. `rowcol2well` returns a character vector of well names. `well2rowcol` returns a data.frame containing the numeric vectors `row` and `col`. `well2wellid` returns a numeric vector containing the well identifiers.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

getWellFeatures

Examples

```
## prw2uname and uname2prw
u = prw2uname(plate=1:2, replicate=1, row=2, col=3:4)
print(u)
prw = uname2prw(u)
print(prw)
prw2uname(prw)

## rowcol2well and well2rowcol
w = rowcol2well(row=1:3, col=5:7)
print(w)
rc = well2rowcol(w)
print(rc)
```

---

readLearnTS *Learn, classify and predict cell labels.*

Description

readLearnTS trains an SVM classifier using cell features and a training cell set. predictCellLabels predicts cell labels.

Usage

```
readLearnTS(x, featurePar, trainingSet, access='cache', cost, gamma)
predictCellLabels(x, uname, access='cache')
```

Arguments

x	An imageHTS object.
uname	A character vector, containing the well names to segment. See getUnames for details.
featurePar	A character string, indicating the filename containing the feature parameters.
trainingSet	A character string, indicating the filename containing the training cell set. See Details.
access	A character string indicating how to access the data. Valid values are local, server and cache, the default. See fileHTS for details.
cost	An optional numeric vector containing the SVM costs to be explored during the cross-validation parameter grid-search. Default is c(0.1, 1, 10, 20).
gamma	An optional numeric vector containing the radial kernel gamma parameters to be explored during the cross-validation parameter grid-search. Default is c(0.0001, 0.001, 0.01, 0.1).

Details

readLearnTS trains an SVM classifier using cell features and a training cell set. Features enumerated in the remove.classification.features field of the feature parameters are not considered for classification. The training set, pointed by trainingSet, is a tab-separated file containing the rows uname, spot, id and label. Each row designates a cell. This file is constructed by using the output of the cellPicker module, see popCellPicker. After completion, readLearnTS writes the a RDA file \'data/classifier.rda\' in the local project directory. This file contains the list returned by readLearnTS.

predictCellLabels uses the trained classifier located in the file \'data/classifier.rda\' and cell features to predict cell labels of wells indicated by uname. For each well, the function writes the file clabels, which contains the predicted cell labels.

If present, popCellPicker shows the predicted cell labels. Several iterations of readLearnTS, predictCellLabels and popCellPicker calls are useful to build an efficient cell classifier.

Value

Returns an invisible list which contains: classifier, the trained classifier obtained by tune.svm and cft, the features that were used to train the classifier.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

popCellPicker

Examples

```
## see vignette for details
```

---

segmentATH *Segmentation and quantification of cells stained for DNA, actin and*

Description

segmentATH and getCellFtrsATH are segmentation and feature extraction functions designed for cell images stained for DNA, actin and tubulin.

Usage

```
segmentATH(x, uname, p, access)
getCellFtrsATH(cal, seg)
```

Arguments

x	An imageHTS object.
uname	A character string, containing the well name to segment.
p	A list of character vectors, containing the segmentation parameters. This is the output of parseDCF, given an input segmentation configuration file.
access	A character string indicating how to access the data. Valid values are local, server and cache, the default. See fileHTS for details.
cal	An EBImage image object containing the calibrated image.
seg	A list of two EBImage image objects: cseg, the cell segmentation mask and nseg, the nucleus segmentation mask.

Details

segmentATH is a segmentation function that can be specified in the seg.method field of a segmentation configuration file, called by the higher-level function segmentWells.

getCellFtrsATH is a feature extraction function that can be specified in the extractfeatures.method field of a feature configuration file, called by the higher-level function extractFeatures.

Value

segmentATH returns a list containing three EBImage images: cal, the calibrated image; nseg, the nucleus mask and cseg, the cell mask.

getCellFtrsATH returns a data frame containing the cell features.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

parseDCF, segmentWells, extractFeatures

Examples

```
## see segmentWells and extractFeatures
```

---

segmentWells *Segment cells in well images*

Description

Segment cells in well images.

Usage

```
segmentWells(x, uname, segmentationPar, access='cache', writeData=TRUE)
```

Arguments

x	An imageHTS object.
uname	A character vector, containing the well names to segment. See getUnames for details.
segmentationPar	A character string, indicating the filename containing the segmentation parameters.
access	A character string indicating how to access the data. Valid values are local, server and cache, the default. See fileHTS for details.
writeData	A boolean indicating whether the segmentation data should be written to the project directory. Default is TRUE.

Details

segmentWells reads the DCF segmentation parameters file pointed by segmentationPar. The file must contain the core segmentation function name indicated in the seg.method field. For each well indicated by uname, segmentWells calls the core segmentation function which returns a list containing a list of three EBImage images: cal, the calibrated image; nseg, the nucleus mask and cseg, the cell mask. See segmentATH for an example of a core segmentation function.

If writeData is TRUE, segmentWells writes for each well indicated by uname: the calibrated image data cal, the segmentation data seg, a calibrated JPEG image viewfull, untiled JPEG images viewunmonted, a calibrated JPEG image with segmentation annotation viewseg, a thumbnail JPEG image viewthumb and Javascript segmentation contour information contour. See fileHTS for details about these files.

Value

If uname is of length 1, returns an invisible list containing: cal, the calibrated image; nseg, the nucleus mask and cseg, the cell mask.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

segmentATH, getUnames, fileHTS

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')

## segment one well
uname = getUnames(x, content='rluc')[1]
z = segmentWells(x, uname=uname, segmentationPar='conf/segmentationpar.txt', writeData=FA
if (interactive()) {
  seg = highlightSegmentation(z$cal, z$nseg, z$cseg)
  display(seg)
```

```
}
```

`segmentYeastBF` *Segmentation of yeast cells and ring-shaped objects.*

Description

`segmentYeastBF` segments yeast cells from bright field microscopy images. `segmentRing` segments ring-shape objects in images.

Usage

```
segmentYeastBF(x, uname, p, access)
segmentRing(a, p)
```

Arguments

`x`	An imageHTS object.
`uname`	A character string, containing the well name to segment.
`p`	A list of character vectors, containing the segmentation parameters. This is the output of `parseDCF`, given an input segmentation configuration file. See details.
`access`	A character string indicating how to access the data. Valid values are `local`, `server` and `cache`, the default. See `fileHTS` for details.
`a`	An EBImage image object or a matrix containing the image to segment.

Details

`segmentYeastBF` is a high-level segmentation function that can be specified in the `seg.method` field of a segmentation configuration file, called by the higher-level function `segmentWells`.

`segmentRing` is used by `segmentYeastBF` and segments an image containing ring-shaped objects. The list of parameters `p` should contain:

- `edge.threshold`: a threshold parameter giving the cell edges
- `max.membrane.thickness`: the maximum membrane thickness, in pixels
- `crown.thickness`: the membrane thickness, in pixels
- `crown.steps`: a vector of 3 values, containing the minimum cell diameter, the maximum cell diameter, and the step between all possible diameters
- `locmin.threshold.width`: the adaptive threshold window width to compute the local minima, to call cell centers
- `locmin.threshold.offset`: the adaptive threshold window offset
- `locmin.erode.size`: the size of the erode paramter cleaning up the local minima map
- `cell.max.overlap`: the maximum cell overlap size, in pixels
- `nucleus.radius.offset`: the nucleus radius negative offset
- `cell.radius.offset`: the cell radius negative offset

Value

segmentYeastBF returns a list containing three EBImage images: cal, the calibrated image; nseg, the nucleus mask and cseg, the cell mask.

segmentRing returns a list containing two EBImage images: nseg, the nucleus mask and cseg, the cell mask.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

segmentWells, segmentATH

Examples

```
filename = system.file('yeast.jpeg', package='imageHTS')
a = readImage(filename)
if (interactive()) display(a)
p = list(edge.threshold=0.05, max.membrane.thickness=5, crown.thickness=8,
    crown.steps=c(31, 61, 4), locmin.threshold.width=9, locmin.threshold.offset=0.15,
    locmin.erode.size=3, cell.max.overlap=2, nucleus.radius.offset=10,
    cell.radius.offset=4)
seg = segmentRing(a, p)
hseg = highlightSegmentation(EBImage::channel(a, 'rgb'), cseg=seg$cseg, thick=TRUE)
if (interactive()) display(hseg)
```

---

summarizeWells *Summarize cell features*

Description

Compute phenotypic profiles by summarizing cell population features.

Usage

```
summarizeWells(x, uname, featurePar, access='cache')
```

Arguments

x	An imageHTS object.
uname	A character vector, containing the well names to summarize. See getUnames for details.
featurePar	A character string, indicating the filename containing the feature extraction parameters.
access	A character string indicating how to access the data. Valid values are local, server and cache, the default. See fileHTS for details.

Details

summarizeWells computes for each well, summary statistics about cell features. Currently, cell number n and median cell feature med.* (for each feature) are computed.

Moreover, if the DCF segmentation parameters file pointed by featurePar includes the field cell.classes, containing a list of comma-separated cell classes, cell classes ratio are computed and included in the phenotypic profiles.

summarizeWells creates the file data/profiles.tab which contains the phenotypic profiles. Use readHTS to read this file.

Value

A data frame, containing the phenotypic profiles.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

See Also

extractFeatures, readHTS

Examples

```
## Not run:
  ## initialize imageHTS object using the local submorph screen
  local = tempdir()
  server = system.file('submorph', package='imageHTS')
  x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
  x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')

  ## segment non-empty wells
  unames = setdiff(getUnames(x), getUnames(x, content='empty'))
  segmentWells(x, uname=unames, segmentationPar='conf/segmentationpar.txt')

  ## extract features
  extractFeatures(x, uname=unames, 'conf/featurepar.txt')

  ## cell classification
  readLearnTS(x, 'conf/featurepar.txt', 'conf/trainingset.txt')
  predictCellLabels(x, unames)

  ## summarize features
  summarizeWells(x, unames, 'conf/featurepar.txt')

  ## get profiles
  profiles = readHTS(x, type='file', filename='data/profiles.tab', format='tab')

## End(Not run)
```

zprime *Compute the Z'-factor quality score*

Description

Compute the Z'-factor quality score.

Usage

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

Arguments

a, b	Matrices of control features.
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'-factor, a numeric ranging from -infinity to 1.

Author(s)

Gregoire Pau, <gregoire.pau@embl.de>, 2010

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.

See Also

readHTS

Examples

```
## initialize imageHTS object using the local submorph screen
local = tempdir()
server = system.file('submorph', package='imageHTS')
x = parseImageConf('conf/imageconf.txt', localPath=local, serverURL=server)
x = configure(x, 'conf/description.txt', 'conf/plateconf.txt', 'conf/screenlog.txt')

## get profiles
profiles = readHTS(x, type='file', filename='data/profiles.tab', format='tab')
a = profiles[match(getUnames(x, content='rluc'), profiles$uname),]
b = profiles[match(getUnames(x, content='ubc'), profiles$uname),]

## compute Z'-factor scores on some features
ft = c('med.c.m.m.ss')
cat('Z\'-factor original=',  zprime(a[,ft], b[,ft], 'original'), 'fixsd=',  zprime(a[,f

## multivariate Z'-factor
ft = c('med.c.m.m.ss', 'med.n.h.m.ss', 'med.c.g.ec')
cat('Z\'-factor mahalanobis=',  zprime(a[,ft], b[,ft], 'mahalanobis'), '\n')
```

Index