Package 'IFC'

August 22, 2022

Encoding UTF-8

Type Package

Title Tools for Imaging Flow Cytometry

Version 0.1.6

Date 2022-08-22

Maintainer Yohann Demont <git.demont@gmail.com>

BugReports https://github.com/gitdemont/IFC/issues

Description Contains several tools to treat imaging flow cytometry data from 'ImageStream®' and 'FlowSight®' cytometers ('Amnis®', part of 'Luminex®'). Provides an easy and simple way to read and write .fcs, .rif, .cif and .daf files. Information such as masks, features, regions and populations set within these files can be retrieved for each single cell. In addition, raw data such as images stored can also be accessed. Users, may hopefully increase their productivity thanks to dedicated functions to extract, visualize, manipulate and export 'IFC' data. Toy data example can be installed through the 'IFCdata' package of approximately 32 MB, which is available in a 'drat' repository <https://gitdemont.github.io/IFCdata/>. See file 'COPYRIGHTS' and file 'AUTHORS' for a list of copyright holders and authors.

Copyright file inst/COPYRIGHTS

License GPL-3

Depends R (>= 3.4.0)

Imports Rcpp (>= 0.10.0), RcppProgress, xml2, png, tiff, jpeg, utils, grid, gridExtra, lattice, latticeExtra, KernSmooth, DT, visNetwork

Suggests IFCdata, shiny, reticulate

LinkingTo Rcpp, RcppProgress

Additional_repositories https://gitdemont.github.io/IFCdata/

RoxygenNote 7.1.1

NeedsCompilation yes

Author Yohann Demont [aut, cre],
Gautier Stoll [ctb],
Guido Kroemer [ldr],
Jean-Pierre Marolleau [ldr],
Loïc Garçon [ldr]

Repository CRAN

Date/Publication 2022-08-22 10:30:05 UTC

R topics documented:

`IFC-package` *Tools for Imaging Flow Cytometry*

Description

Contains several tools to treat Imaging Flow Cytometry data from ImageStream(R) and FlowSight(R) cytometers (Amnis(R), part of Luminex(R)). Provides an easy and simple way to read or write .fcs, .rif, .cif and .daf files. Information such as masks, features, regions and populations set within these files can be retrieved for each single cell. In addition, raw data such as images stored can also be accessed. Users, may hopefully increase their productivity thanks to dedicated functions to extract, visualize, manipulate and export IFC data.

Details

The IFC package provides several categories of functions:
- to read / write / export / visualize:
`readIFC`, `writeIFC`, `ExtractFromDAF`, `ExportToDAF`, `data_to_DAF`, `ExtractFromXIF`, `ExportToXIF`, `ExtractFromFCS`, `ExportToFCS`, `ExportToBATCH`, `BatchReport`, `ExportToReport`, `ExportToGallery`, `ExportToNumpy`, `DisplayGallery`, `ExtractImages_toBase64`, `ExtractImages_toFile`, `ExtractImages_toMatrix`, `ExtractMasks_toMatrix`, `objectExtract`, `popsNetwork`, `plotGraph`, `paletteIFC`, `autoplot`
- to transform features values
`smoothLinLog`, `inv_smoothLinLog`
- to deeply extract information from files:
`getInfo`, `getOffsets`, `getIFD`, `getFullTag`, `getAborted`
- dedicated to populations:
`popsCopy`, `popsGetObjectsIds`, `popsNetwork`, `popsRename`
- for adding / removing features, regions, populations:
`data_add_features`, `data_add_regions`, `data_add_pops`, `data_rm_features`, `data_rm_regions`, `data_rm_pops`

- to allow several coercion:
buildBatch, buildFeature, buildGraph, buildPopulation, buildRegion

Author(s)

Maintainer: Yohann Demont <git.demont@gmail.com>

---

autoplot *Automatic Parameters Detection for IFC Graphs*

Description

Function intended to generate IFC graphs with minimal inputs from users.
It is essentially based on automatic detection of graphical parameters thanks to 'shown_pops' argument.

Usage

```
autoplot(
  obj,
  shown_pops = NULL,
  subset = NULL,
  x = NULL,
  x_trans = NULL,
  y = NULL,
  y_trans = NULL,
  type = NULL,
  smoothingfactor = NULL,
  normalize = NULL,
  bin,
  viewport = "ideas",
  precision = c("light", "full")[1],
  color_mode = c("white", "black")[1],
  draw = TRUE,
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
shown_pops	one or several populations present in 'obj'. Default is NULL. If provided, autoplot will try to display these populations. See details when not provided. autoplot will try to determine x and y and their transformations based on

'shown_pops' parameter. If all populations provided in 'shown_pops' are siblings, region(s) from which 'shown_pops' were defined will be displayed.
In case 'shown_pops' are not siblings, they will be treated as populations and a graph will be generating with an overlay of these populations. Order of this overlay is given by order of 'shown_pops'.
Finally, changing any of the following arguments (x, x_trans, y, y_trans, type) to something else than the one detected from 'shown_pops' will prevent from displaying region(s) and 'shown_pops' populations will be displayed as overlay. However, please consider that if original type is 'histogram' changing x_trans transformation will have no impact on this.

`subset` a population present in 'obj'. Default is NULL. Background population that will be used to generate graph. This argument will not be used when graph is an histogram. If this argument is filled with a different population than what can be determined thanks to 'shown_pops', Then 'shown_pops' will be treated as overlay. However, 'shown_pops' argument can still be used to determine x, y axis and their transformation

`x` feature for x-axis. Default is NULL. When empty, `autoplot` will try to determine if automatically from 'shown_pops' argument. If provided, x feature has to be a name from 'obj' features. Note that providing x feature : - takes precedence on automatic x-axis detection. - will reset x-axis transformation to "P" except if 'x_trans' is filled.

`x_trans` parameter for x-axis transformation. Default is NULL. If not provided, transformation will be determined thanks to 'shown_pops'. It takes precedence when provided and if provided it has to be be either "P" or coercible to a positive numeric. "P" will leave x-axis as is but a positive numeric will be passed has hyper argument of `smoothLinLog` to transform x-axis.

`y` feature for y-axis. Default is NULL. When empty, `autoplot` will try to determine it automatically from 'shown_pops' argument. If provided, y feature has to be a name from obj features. Note that providing y feature - takes precedence on automatic y-axis detection. - will reset y-axis transformation to "P" except if 'y_trans' is filled.

`y_trans` parameter for y-axis transformation. Default is NULL. If not provided, transformation will be determined thanks to 'shown_pops'. It takes precedence when provided and and if provided it has to be be either "P" or coercible to a positive numeric. "P" will leave y-axis as is but a positive numeric will be passed has hyper argument of `smoothLinLog` to transform y-axis. Note that it is irrelevant for "histogram".

`type` type of plot. Default is NULL to allow `autoplot` to detemine 'type' automatically. If provided it has to be either "histogram", "scatter", "density". Note that when "histogram" is choosen, 'subset' parameter will not be used. Note that "density" will be possible only when 'subset' will be automatically determined or filled with only one population. Note that when `autoplot` has determined, thanks to 'shown_pops' that original plot is an "histogram", "Object Number" will be used as y-axis by default when 'type' is forced to "scatter" or "density".

`smoothingfactor`
when type of graph is "histogram", whether to smooth it or not. Default is NULL. Should be an integer [0:20] Note that 0 means no smoothing and other

	values will produce smoothing
normalize	when type of graph is "histogram", whether to normalize it or not. Default is NULL. Should be a logical.
bin	number of bins when graph's type is "histogram" / number of equally spaced grid points for density. Default is missing to allow autoplot to determine it by itself.
viewport	Either "ideas", "data" or "max" defining limits used for the graph. Default is "ideas". -"ideas" will use same limits as the one defined in ideas. -"data" will use data to define limits. -"max" will use data and regions drawn to define limits.
precision	when graphs is a 2D scatter with population overlay, this argument controls amount of information displayed. Default is "light". -"light", the default, will only display points of same coordinates that are amoung the other layers. -"full" will display all the layers.
color_mode	Whether to extract colors from obj in white or black mode. Default is "white".
draw	whether to draw plot. Default is TRUE.
...	Other arguments to be passed.

Details

when 'shown_pops' are not provided, autoplot can't determine anything.
So, if not provided default values will be used:
-'subset' = "All"
-'x' = "Object Number"
-'x_trans' = "P"
-'y' = "Object Number"
-'y_trans' = "P"
-'type' = "histogram"

Value

an **lattice** trellis object

BatchReport *Batch Generation of Graphical and Statistic Report*

Description

Batch creates graphical an statistical report.

Usage

```
BatchReport(
  fileName,
  obj,
  selection,
  write_to,
  overwrite = FALSE,
  gating,
  main,
  byrow = FALSE,
  times = 5,
  color_mode = c("white", "black")[1],
  add_key = "panel",
  precision = c("light", "full")[1],
  trunc_labels = 38,
  trans = "asinh",
  bin,
  viewport = "ideas",
  display_progress = TRUE,
  ...
)
```

Arguments

fileName, obj	either one or the other. Path to file(s) to read from for 'fileName' or list of 'IFC_data' objects for obj.
selection	indices of desired graphs. It can be provided as an integer vector or as a matrix. In such case, the layout of the matrix will reflect the layout of the extracted graphs for each 'fileName' or "obj'. NA value will result in an empty place. When missing, it will be determined by the whole layout of 1st 'fileName' or 'obj' with 'gating' applied when provided
write_to	pattern used to export file(s). Placeholders, like c("%d/%s_fromR.pdf", "%d/%s_fromR.csv"), will be substituted: -%d: with full path directory -%p: with first parent directory -%e: with extension (without leading .) -%s: with shortname (i.e. basename without extension). Exported file(s) extension(s) will be deduced from this pattern using either 1st 'fileName' or 'obj'. Note that has to be a .pdf and/or .csv.
overwrite	whether to overwrite file or not. Default is FALSE. Note that if TRUE, it will overwrite file. In addition a warning message will be sent.
gating	an 'IFC_gating' object as extracted by readGatingStrategy(). Default is missing. If not missing, each 'IFC_data' provided in 'obj' or read from 'fileName' will be passed to applyGatingStrategy() before creating the report.
main	the main title of the document. Default is missing.
byrow	whether to add selected graphs for each file by row or not. Default is FALSE.

times	number of files to add before starting a new row or column (depending on 'by-row').
color_mode	Whether to extract colors in white or black mode. Default is "white".
add_key	whether to draw a "global" key under title or in the first "panel" or "both". Default is "panel". Accepted values are either: FALSE, "panel", "global", "both" or c("panel", "global"). Note that it only applies when display is seen as overlaying populations.
precision	when graphs is a 2D scatter with population overlay, this argument controls amount of information displayed. Default is "light". -"light", the default, will only display points of same coordinates that are among the other layers. -"full" will display all the layers.
trunc_labels	maximum number of characters to display for labels. Default is 38.
trans	name of the transformation function for density graphs. If missing the default, the BasePop[[1]]$densitytrans, if any, will be retrieved, otherwise "asinh" will be used.
bin	default number of bin used for histogram. Default is missing.
viewport	Either "ideas", "data" or "max" defining limits used for the graph. Default is "ideas". -"ideas" will use same limits as the one defined in ideas. -"data" will use data to define limits. -"max" will use data and regions drawn to define limits.
display_progress	whether to display a progress bar. Default is TRUE.
...	other parameters to be passed.

Value

It invisibly returns full path of exported .pdf and/or .csv file(s).

`buildBatch` *Batch Builder*

Description

Prepares XML node for `ExportToBATCH`.

Usage

```
buildBatch(
  files,
  compensation,
  analysis,
  default_batch_dir,
```

```
  config_file,
  name = "Batch1",
  use_acquisition = FALSE,
  suffix = "",
  allow_channels_dissimilarity = FALSE,
  overwrite = TRUE,
  segment_rif = "None",
  options
)
```

Arguments

`files`	path of files to batch.
`compensation`	path to compensation file.
`analysis`	path to analysis file.
`default_batch_dir`	directory where batches are stored. It can be found in IDEAS(R) software, under Options -> Application Defaults -> Directories -> Default Batch Report Files Directory. If missing, the default, it will be deduced from IDEAS(R) config file, However, if it can't be deduced then tempdir(check = TRUE) from **base** will be used. This argument takes precedence over 'config_file' and filling 'default_batch_dir' prevents the use of 'config_file' argument.
`config_file`	path to IDEAS(R) config file. It may depends on IDEAS(R) software installation but one may use "C:/Users/%USER%/AppData/Roami Corporation/userconfig.xml".
`name`	name of batch. Default is "Batch1".
`use_acquisition`	whether to use acquisition as analysis template. Default is FALSE.
`suffix`	suffix to add to files when batched. Default is "".
`allow_channels_dissimilarity`	whether to allow building batch when all files were not acquired with same channels. Default is FALSE.
`overwrite`	whether to overwrite files or not. Default is TRUE.
`segment_rif`	size of file segmentation. Default is "None", for no segmentation. Allowed are "None", "100", "1K", "5K", "10K", "50K", "100K".
`options`	A list of arguments to be passed. If missing, the default, options will be set to: -"Brightfield compensation"=TRUE, -"EDF deconvolution"=TRUE, -"Camera background"=TRUE, -"Spatial alignment"=TRUE. Allowed are TRUE or FALSE for all, excepted for 'Spatial aligment' which can also be path to .rif file.

Value

a list containing batch information:
-xml, the xml object to be written,
-batch_dir, the directory where xml file is desired to be saved according to 'default_batch_dir' and 'config_file'.

`buildFeature` *IFC Feature Coercion*

Description

Helper to build a list to allow feature export.

Usage

```
buildFeature(
  name,
  type = c("single", "combined", "computed")[1],
  def = "Camera Line Number",
  val = NULL,
  ...
)
```

Arguments

`name`	feature's name. If missing, it will be determined thanks to def.
`type`	feature's type. Default is "single". Allowed are "single", "combined", "computed".
`def`	definition of the feature. Default is "Camera Line Number".
`val`	a coercible to numeric vector of feature values. Default is NULL. Note that although not mandatory for `buildFeature`it has to be provided to allow feature export in `ExportToDAF` and `data_add_features`.
`...`	Other arguments to be passed.

Value

a list containing all feature information.

buildGraph *IFC Graph Coercion*

Description

Helper to build a list to allow graph export.

Usage

```
buildGraph(
  type = c("histogram", "scatter", "density")[3],
  xlocation = 0,
  ylocation = 0,
  f1 = "Object Number",
  f2 = "Object Number",
  scaletype = 1,
  xmin = -1,
  xmax = 1,
  ymin = 0,
  ymax = 1,
 title = paste0(unlist(lapply(BasePop, FUN = function(x) x$name)), collapse = ", "),
  xlabel = f1,
  ylabel = f2,
  axislabelsfontsize = 10,
  axistickmarklabelsfontsize = 10,
  graphtitlefontsize = 12,
  regionlabelsfontsize = 10,
  bincount = 0,
  freq = c("T", "F")[1],
  histogramsmoothingfactor = 0,
  xlogrange = "P",
  ylogrange = "P",
  maxpoints = +Inf,
  stats = c("true", "false")[2],
  xsize = c(320, 480, 640)[1],
  ysize = xsize + ifelse(stats == "true", splitterdistance, 0),
  splitterdistance = 120,
  xstats = "Count|%Gated|Mean",
  ystats = xstats,
  order,
  xstatsorder,
  Legend,
  BasePop = list(list()),
  GraphRegion = list(list()),
  ShownPop = list(list()),
  ...
)
```

Arguments

type	Graph's type. Either "histogram", "scatter" or "density". Default is "density".
xlocation	Integer. Graph's x location. Default is 0.
ylocation	Integer. Graph's x location. Default is 0.
f1	Character. Graph x axis parameter. Default is "Object Number".
f2	Character. Graph y axis parameter. Default is "Object Number". Only used when 'type' is not "histogram".
scaletype	Integer. Graph scale. Either 0 (auto), 1 (manual). Default is 1.
xmin	Double. Graph's xmin. Default -1.
xmax	Double. Graph's xmax. Default 1.
ymin	Double. Graph's xmin. Default 0.
ymax	Double. Graph's xmax. Default 1.
title	Character. Graph title label. Default will use names of BasePop collapse with ', '.
xlabel	Character. Graph x axis label.
ylabel	Character. Graph y axis label.
axislabelsfontsize	Integer. Axis label font size. Default is 10. Allowed are: 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28. Checked but not yet implemented.
axistickmarklabelsfontsize	Integer. Axis tick font size. Default is 10. Allowed are: 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28. Checked but not yet implemented.
graphtitlefontsize	Integer. Axis title font size. Default is 12. Allowed are: 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28. Checked but not yet implemented.
regionlabelsfontsize	Integer. Axis region font size. Default is 10. Allowed are: 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28. Checked but not yet implemented.
bincount	Integer. Histogram bin count. Default is 0. Allowed are: 0, 8, 16, 32, 64, 128, 256, 512, 1024.
freq	Character. Histogram with frequency normalization of not. Default is "T", allowed are "T" or "F".
histogramsmoothingfactor	Integer. Histogram smoothing factor. Allowed are [0-20]. Only partly implemented, default is 0 for no smoothing other values will produce same smoothing.
xlogrange	determines transformation instruction for x-axis. Default is "P" for no transformation.
ylogrange	determines transformation instruction for y-axis. Default is "P" for no transformation.

`maxpoints` determines the maximum number of points to display. Default is +Inf to display all points.
If provided, values from]0,1] will be used as a proportion of the total number of points to show.
While values values superior to 1 will be interpreted as the maximal number of points to show.
It only applies to 2D graphs. When 'type' is "histogram", +Inf will be used whatever the value provided as input.

`stats` Character. Either "true" or "false" to display stats. Default is "false".

`xsize` Integer. Graph's x size. Default is 320 for small. Regular are: 320 (small), 480 (medium), 640 (big). Checked but not yet implemented.

`ysize` Integer. Graph's y size. Default is 'ysize' + 'splitterdistance' when 'stats' is set to "true". Checked but not yet implemented.

`splitterdistance`
Integer. Default is 120. Checked but not yet implemented.

`xstats` Character. x stats to be computed. Default is 'Count|%Gated|Mean'. It has to be a filled with the concatenation of 'Count', '%Total', '%Gated', '%Plotted', 'Objects/mL', 'Mean', 'Median', 'Std. Dev.', 'MAD', 'CV', 'Minimum', 'Maximum', 'Geo. Mean', 'Mode', 'Variance' and /or 'NaN', collapse with 'I'. Checked but not yet implemented.

`ystats` Character. y stats to be computed. Should be identical to 'xstats'. Default is xstats. Checked but not yet implemented.

`order` Character. Order to display populations. When 'type' is "density" it will be BasePop[[1]]$name. When 'type' is "histogram" or "density" 'ShownPop' are not allowed Otherwise, it will use each of 'GraphRegion', 'BasePop' and 'ShownPop' names, collapse with 'I'.

`xstatsorder` Character. Order of stat rows. It will use each of 'GraphRegion' names & each of 'BasePop' names, reverted and collapse with 'I'.

`Legend` Default is list(list(onoff='false',x='0',y='0',witdh='96',height='128')). Not yet implemented.

`BasePop` Default is list(list()). See details.

`GraphRegion` Default is list(list()). Only allowed member are sub-list(s) with only one character component named 'name'.

`ShownPop` Default is list(list()). Only allowed member are sub-list(s) with only one character component named 'name'.

`...` Other arguments to be passed.

Details

Many parameters are not used or are only partly implemented, but most are checked in order to be compatible for further export.
For 'BasePop', if left as is "All" will be used as default.
This parameter will be built / checked according to 'type' argument.
'BasePop' has to be a list of list(s) and each sub-list should can contain several elements, but only "name" is mandatory.

The sublist members are:
-"name", "linestyle", "fill",
and only when 'type' is "density"
-"densitybincount", "densitymin", "densitymax",
-"densitycolors", "densitycolorslightmode", "densitycolorsdarkmode",
-"densitylevel", "densitytrans".
Each sub-list will be created automatically with the following default values (except if explicitly provided):
-linestyle="Solid",
-fill="true",
-densitybincount="128",densitymin="0",densitymax="0",
-densitycolors="-16776961|-13447886|-256|-23296|-65536|",
-densitycolorslightmode="-16776961|-13447886|-256|-23296|-65536|",
-densitycolorsdarkmode="-16776961|-13447886|-256|-23296|-65536|",
-densitylevel="",
*when provided it has to be in a format of "fill[true,false]|lines[true,false]|nlevels[integer>1]|lowest[numeric[0-1[]|" *describing how the levelplot should be drawn.
*Besides, 'densitrans' will not be used. -densitytrans="asinh"
*it can take a function to be applied to the 2D local densities
*or a name of a feature within 'IFC_data' object to draw a gradient against this feature
Note that when 'type' is "density", 'BasePop' should be of length one.
and fill will be overwritten to "true".

Value

a list containing all graph information.

`buildPopulation` *IFC Population Coercion*

Description

Helper to build a list to allow population export.

Usage

```
buildPopulation(
  name,
  type,
  base = "All",
  color,
  lightModeColor,
  style,
  region,
  fx,
  fy,
  definition,
```

```
  obj,
  ...
)
```

Arguments

name	name of the population.
type	type of population. Either "B", "C", "G" or "T" for Base, Combined, Graphical or Tagged, respectively. If missing, the default, 'type' will be deduced from other parameters. If 'name' is "All" type will be "B". Otherwise, if 'fx' is given type will be "G". Otherwise, "T", if 'definition' is missing but not 'obj' or "C" if 'definition' is not missing.
base	which population is based on. It will be base="All", for 'type' "T" and "C" and base="", for 'type' "B". It is only needed when type = "G".
color	color of the population. See paletteIFC for allowed colors. If not provided, it will be sampled.
lightModeColor	lightModeColor of the population. See paletteIFC for allowed colors. If not provided, it will be sampled.
style	style of the population. Either 20, 4, 3, 1, 5, 0, 2, 18, 15, 17, respectively for: "Simple Dot", "Cross", "Plus", "Empty Circle", "Empty Diamond", "Empty Square", "Empty Triangle", "Solid Diamond", "Solid Square", "Solid Triangle".
region	Only if type="G". Name of the region defining the population.
fx	Only needed if type="G". Name of the x-feature defining the population.
fy	Only needed if type="G" and only if region is defined in 2D. Name of the y-feature defining the population.
definition	Only needed if type="C". Parameters defining the population.
obj	Only needed if type="T". Either a: -Logical vector of same length as "All" population indicating if a cell belongs to the population or not. -Integer vector of indices of cells that belongs to the population. Note that first object is 0.
...	Other arguments to be passed.

Value

a list containing all population information.

`buildRegion` *IFC Region Coercion*

Description

Helper to build a list to allow region export.

Usage

```
buildRegion(
  type,
  label,
  cx,
  cy,
  color,
  lightcolor,
  ismarker = "false",
  doesnotoverride = "false",
  xlogrange,
  ylogrange,
  x,
  y,
  ...
)
```

Arguments

type	Region's type. Either "line", "rect", "poly" or "oval".
label	label of the region.
cx	x label's position. If not provided x center will be used.
cy	y label's position. If not provided y center will be used.
color	color of the region. See paletteIFC for allowed colors.
lightcolor	lightcolor of the region. See paletteIFC for allowed colors.
ismarker	Default is 'false'. Allowed are 'true' or 'false'. Used for compatibility with amnis file but role remains unknown.
doesnotoverride	Default is 'false'. Allowed are 'true' or 'false'. Used for compatibility with amnis file but role remains unknown.
xlogrange	determines transformation instruction for x-axis. Default is "P" for no transformation.
ylogrange	determines transformation instruction for y-axis. Default is "P" for no transformation.
x	vector of x vertices values.
y	vector of y vertices values.
...	Other arguments to be passed.

Value

a list containing all region information.

checksumIFC *IFC Files Checksum*

Description

This function returns RIF/CIF checksum. Checksum is the sum of img IFDs (Image Field Directory) offsets of objects 0, 1, 2, 3 and 4.

Usage

```
checksumIFC(fileName, ...)
```

Arguments

fileName	path to file.
...	arguments to pass to checksumDAF or checksumXIF.

Details

if fileName is a DAF file, then CIF checksum is computed from images values found in DAF.

Value

an integer corresponding to IFC file checksum.

data_add_features *Add Feature to IFC_data Object*

Description

Adds features to an already existing 'IFC_data' object.

Usage

```
data_add_features(obj, features, ...)
```

Arguments

obj	an 'IFC_data' object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
features	a list of features to add to obj. Each element of this list will be coerced by buildFeature.
...	Other arguments to be passed.

Details

A warning will be thrown if a provided feature is already existing in obj.
In such a case this feature will not be added to obj.
If any input feature is not well defined and can't be created then an error will occur.

Value

an IFC_data object with features added.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  ## copy 1st feature found in daf
  feat_def <- daf$features_def[[1]]
  if(length(feat_def) != 0) {
    feat_def_copy <- feat_def
    ## modify name and value of copied features
    feat_def_copy$name <- "copied_feature"
    feat <- daf$features[, feat_def$name]
    feat_copy <- feat
    feat_copy <- feat_copy * 10
    ## create new object with this new feature
   dafnew <- data_add_features(obj = daf, features = list(c(feat_def_copy, list(val = feat_copy))))
  }
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`data_add_pops` *Add Population to IFC_data Object*

Description

Adds populations to an already existing 'IFC_data' object.

Usage

```
data_add_pops(
  obj,
  pops,
  pnt_in_poly_algorithm = 1,
  pnt_in_poly_epsilon = 1e-12,
  display_progress = TRUE,
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
pops	a list of population(s) to add to 'obj'. Each element of this list will be coerced by buildPopulation.
pnt_in_poly_algorithm	algorithm used to determine if object belongs to a polygon region or not. Default is 1. Note that for the moment only 1(Trigonometry) is available.
pnt_in_poly_epsilon	epsilon to determine if object belongs to a polygon region or not. It only applies when algorithm is 1. Default is 1e-12.
display_progress	whether to display a progress bar. Default is TRUE.
...	Other arguments to be passed.

Details

A warning will be thrown if a provided population is already existing in 'obj'.
In such a case this population will not be added to 'obj'.
If any input population is not well defined and can't be created then an error will occur.

Value

an IFC_data object with pops added.

Source

For pnt_in_poly_algorithm, Trigonometry, is an adaptation of Jeremy VanDerWal's code https://github.com/jjvanderwal/SDMTools

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  ## copy 1st population from existing daf
  pop <- daf$pops[[1]]
  if(length(pop) != 0) {
    pop_copy <- pop
    ## modify name, obj and type of copied population
    pop_copy$name <- paste0(pop_copy$name,"_copy")
    pop_copy$obj <- (which(pop_copy$obj)-1)[1]
    pop_copy$type <- "T"
    ## create new object with this new population
    dafnew <- data_add_pops(obj = daf, pops = list(pop_copy))
  }
} else {
```

```
  message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                  'https://gitdemont.github.io/IFCdata/',
                  'to install extra files required to run this example.'))
}
```

`data_add_regions` *Add Region to IFC_data Object*

Description

Adds regions to an already existing 'IFC_data' object.

Usage

```
data_add_regions(obj, regions, ...)
```

Arguments

`obj`	an 'IFC_data' object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
`regions`	a list of region(s) to add to obj. Each element of this list will be coerced by `buildRegion`.
`...`	Other arguments to be passed.

Details

A warning will be thrown if a provided region is already existing in 'obj'.
In such a case this region will not be added to 'obj'.
If any input population is not well defined and can't be created then an error will occur.

Value

an IFC_data object with regions added.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  ## copy 1st region found in daf
  reg <- daf$regions[[1]]
  if(length(reg) != 0) {
    reg_copy <- reg
    ## modify region label and x boundaries
    reg_copy$label <- paste0(reg_copy$label,"_copy")
    reg_copy$x <- reg_copy$x*0.9
    ## create new object with this new region
```

```
    dafnew <- data_add_regions(obj = daf, regions = list(reg_copy))
  }
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`data_rm_features` *Remove Features from an IFC_data Object*

Description

Removes regions from an already existing 'IFC_data' object.

Usage

```
data_rm_features(obj, features, list_only = TRUE, adjust_graph = TRUE, ...)
```

Arguments

`obj`	an 'IFC_data' object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
`features`	a character vector of features names to remove within 'obj'. Note that "Object Number" is not allowed and will be excluded from 'features' if present.
`list_only`	whether to return a list of elements that will be impacted by the removal. Default is TRUE. If FALSE then modified object will be returned.
`adjust_graph`	whether to try to adjust graph when possible. Default is TRUE.
`...`	Other arguments to be passed.

Value

an 'IFC_data' object or a list of elements impacted by removal depending on 'list_only' parameter.

`data_rm_pops` *Remove Population from an IFC_data Object*

Description

Removes populations from an already existing 'IFC_data' object.

Usage

```
data_rm_pops(obj, pops, list_only = TRUE, adjust_graph = TRUE, ...)
```

Arguments

obj	an ‘IFC_data‘ object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
pops	a character vector of population names to remove within ’obj’. Note that "All" and "" are not allowed and will be excluded from ’pops’ if present.
list_only	whether to return a list of elements that will be impacted by the removal. Default is TRUE. If FALSE then modified object will be returned.
adjust_graph	whether to try to adjust graph when possible. Default is TRUE.
...	Other arguments to be passed.

Value

an ‘IFC_data‘ object or a list of elements impacted by removal depending on ’list_only’ parameter.

---

data_rm_regions *Remove Region from an IFC_data Object*

Description

Removes regions from an already existing ‘IFC_data‘ object.

Usage

```
data_rm_regions(obj, regions, list_only = TRUE, adjust_graph = TRUE, ...)
```

Arguments

obj	an ‘IFC_data‘ object extracted by ExtractFromDAF(extract_features = TRUE) or ExtractFromXIF(extract_features = TRUE).
regions	a character vector of regions names to remove within ’obj’.
list_only	whether to return a list of elements that will be impacted by the removal. Default is TRUE. If FALSE then modified object will be returned.
adjust_graph	whether to try to adjust graph when possible. Default is TRUE.
...	Other arguments to be passed.

Value

an ‘IFC_data‘ object or a list of elements impacted by removal depending on ’list_only’ parameter.

data_to_DAF *DAF File Writer*

Description

Writes an 'IFC_data' object to a daf file

Usage

```
data_to_DAF(
  obj,
  write_to,
  viewing_pop = "All",
  overwrite = FALSE,
  binary = TRUE,
  endianness = .Platform$endian,
  display_progress = TRUE,
  verbose = FALSE,
  fullname = TRUE,
  cifdir = dirname(obj$fileName),
  ntry = +Inf,
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted with features extracted.
write_to	pattern used to export file. Placeholders, like "%d/%s_fromR.%e", will be substituted: -%d: with full path directory of 'obj$fileName' -%p: with first parent directory of 'obj$fileName' -%e: with extension of 'obj$fileName' (without leading .) -%s: with shortname from 'obj$fileName' (i.e. basename without extension). Exported file extension will be deduced from this pattern. Note that it has to be a .daf.
viewing_pop	Character String. Allow user to change displayed population. Default is 'All'.
overwrite	whether to overwrite file or not. Default is FALSE. Note that if TRUE, it will overwrite exported file if path of 'obj$fileName' and deduced from 'write_to' arguments are different. Otherwise, you will get an error saying that overwriting source file is not allowed. Note also that an original file, i.e. generated by IDEAS(R) or INSPIRE(R), will never be overwritten. Otherwise, you will get an error saying that overwriting original file is not allowed.

binary	whether to write object to file in binary mode or not. Default is TRUE. Note that it can represent a convenient way to make file written in binary mode back-compatible with former version of IDEAS software. /!\ However unexpected behaviour may happen if features, regions, pops, ... are depending on masks (e.g. AdaptiveErode, Component, LevelSet, Watershed) introduced in newer version of IDEAS software. /!\ Important please note that conversion from binary to non-binary and back to binary may create some rounding adjustment resulting in some features/image values changes. Finally, if data originate from FCS, 'binary' will be forced to FALSE.
endianness	The endian-ness ("big" or "little") of the target system for the file. Default is .Platform$endian. Endianness describes the bytes order of data stored within the files. This parameter may not be modified.
display_progress	whether to display a progress bar. Default is TRUE.
verbose	whether to display information (use for debugging purpose). Default is FALSE.
fullname	whether to export daf file with full name of its corresponding cif, if found. Default is TRUE. If cif can't be found, daf file will be exported with the original cif file name.
cifdir	the path of the directory to initially look to cif file. Default is dirname(obj$fileName). Only apply when 'fullname' is set to TRUE.
ntry	number of times data_to_DAF will be allowed to find corresponding cif file. Default is +Inf. Only apply when 'fullname' is set to TRUE.
...	other arguments to be passed.

Value

It invisibly returns full path of exported file.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  tmp <- tempdir(check = TRUE)
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  ## add a new population to daf
  dafnew <- data_add_pops(daf, list(buildPopulation(name = "test", type = "T", obj = 0)))
  ## export obj to file using binary mode
  data_to_DAF(obj = dafnew, write_to = paste0(tmp, "\\test_bin.daf"),
              overwrite = TRUE, binary = TRUE)
  ## exporting to non binary mode
  data_to_DAF(obj = dafnew, write_to = paste0(tmp, "\\test_notbin.daf"),
              overwrite = TRUE, binary = FALSE)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
```

```
                  'to install extra files required to run this example.'))
}
```

DisplayGallery *Gallery Display*

Description

Displays gallery of 'IFC_img' / 'IFC_msk' objects

Usage

```
DisplayGallery(
  ...,
  objects,
  offsets,
  image_type = "img",
  layout,
  name = "DisplayGallery",
  caption = FALSE,
  pageLength = 10L,
  pdf_pageSize = "A2",
  pdf_pageOrientation = "landscape",
  pdf_image_dpi = 96,
  extract_max = 10,
  sampling = FALSE,
  display_progress = TRUE,
  mode = c("rgb", "gray")[1]
)
```

Arguments

`...`	arguments to be passed to `objectExtract` with the exception of 'ifd' and 'bypass'(=TRUE). If 'param' is provided 'export'(="base64") and the above parameters will be overwritten. If 'offsets' are not provided extra arguments can also be passed with ... to `getOffsets`. /!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to `objectParam`.
`objects`	integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.
`offsets`	object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.
`image_type`	image_type of desired offsets. Either "img" or "msk". Default is "img".

layout	a character vector of [acquired channels + 'composite' images] members to export. Default is missing to export everything. Note that members can be missing to be removed from final display. Note that members not found will be automatically removed and a warning will be thrown.
name	id of the datatable container. Default is DisplayGallery.
caption	whether to display caption name or not. Default is FALSE.
pageLength	integer, number of objects to display per page. Default is 10.
pdf_pageSize	string, page dimension when exporting to pdf. Default is "A2".
pdf_pageOrientation	string, page orientation when exporting to pdf. Default is "landscape". Allowed are "landscape" or "portrait".
pdf_image_dpi	integer, desired image resolution. Default is 96, for full resolution.
extract_max	maximum number of objects to extract. Default is 10. Use +Inf to extract all.
sampling	whether to sample objects or not. Default is FALSE.
display_progress	whether to display a progress bar. Default is TRUE.
mode	(objectParam argument) color mode export. Either "rgb" or "gray". Default is "rgb".

Details

arguments of objectExtract will be deduced from DisplayGallery input arguments.
Please note that PDF export link will be available if 'write_to' will not result in a "bmp".
Please note that exporting to "tiff" may depend on browser capabilities.
Please note that a warning may be sent if gallery to display contains large amount of data. This is due to use of datatable() from **DT**.
Warning message:
In instance$preRenderHook(instance) :
It seems your data is too big for client-side DataTables. You may consider server-side processing: http://rstudio.github.io/DT/server.html
For these reasons, it may be better to use "png" extension to display images.

Value

it invisibly returns a list whose members are:
-data, data for DT::datatable(),
-args, associated arguments to pass to DT::datatable().

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  cif <- ExtractFromXIF(fileName = file_cif)
  info <- getInfo(fileName = file_cif, from = "analysis")
  ## randomly show at most 10 "img" objects from file
```

```
  DisplayGallery(info = info, image_type = "img", extract_max = 10,
                 sampling = TRUE, write_to = "example.png")
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`ExportToBATCH` *Batch File Writer*

Description

Writes an XML file to batch files

Usage

```
ExportToBATCH(batch)
```

Arguments

`batch`	list of batch nodes as created by `buildBatch`.

Value

It invisibly returns full path of xml batch file.

`ExportToDAF` *DAF File Writer*

Description

Writes a new DAF file based on another one and exports new region(s), pop(s), feature(s), graph(s) and / or mask(s).

Usage

```
ExportToDAF(
  fileName,
  write_to,
  pops = list(),
  regions = list(),
  features = list(),
  graphs = list(),
  masks = list(),
  viewing_pop = "All",
```

```
  endianness = .Platform$endian,
  verbose = FALSE,
  overwrite = FALSE,
  fullname = TRUE,
  cifdir = dirname(fileName),
  ntry = +Inf,
  ...
)
```

Arguments

fileName	path of file to read data from.
write_to	pattern used to export file. Placeholders, like "%d/%s_fromR.%e", will be substituted: -%d: with full path directory of 'fileName' -%p: with first parent directory of 'fileName' -%e: with extension of 'fileName' (without leading .) -%s: with shortname from 'fileName' (i.e. basename without extension). Exported file extension will be deduced from this pattern. Note that has to be a .daf.
pops	list of population(s) to export. Will be coerced to exportable format by buildPopulation.
regions	list of region(s) to export. Will be coerced to exportable format by buildRegion.
features	list of feature(s) to export.
graphs	list of graph(s) to export. Not yet implemented.
masks	list of mask(s) to export. Not yet implemented.
viewing_pop	Character String. Allow user to change displayed population. Default is 'All'.
endianness	The endian-ness ("big" or "little") of the target system for the file. Default is .Platform$endian. Endianness describes the bytes order of data stored within the files. This parameter may not be modified.
verbose	whether to display information (use for debugging purpose). Default is FALSE.
overwrite	whether to overwrite file or not. Default is FALSE. Note that if TRUE, it will overwrite exported file if path of 'fileName' and deduced from 'write_to' arguments are different. Otherwise, you will get an error saying that overwriting source file is not allowed. Note also that an original file, i.e. generated by IDEAS(R) or INSPIRE(R), will never be overwritten. Otherwise, you will get an error saying that overwriting original file is not allowed.
fullname	whether to export daf file with full name of its corresponding cif, if found. Default is TRUE. If cif can't be found, daf file will be exported with the original cif file name.
cifdir	the path of the directory to initially look to cif file. Default is dirname(fileName). Only apply when 'fullname' is set to TRUE.

ntry	number of times ExportToDAF will be allowed to find corresponding cif file. Default is +Inf. Only apply when 'fullname' is set to TRUE.
...	other arguments to be passed.

Value

It invisibly returns full path of exported file.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  tmp <- tempdir(check = TRUE)
  ## create a tagged population named test with 1st object
  pop <- buildPopulation(name = "test", type = "T", obj = 0)
  ExportToDAF(file_daf, write_to = paste0(tmp, "\\test.daf"),
              overwrite = TRUE, pops = list(pop))
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

ExportToFCS *FCS File Writer*

Description

Writes an 'IFC_data' object to a Flow Cytometry Standard (FCS) file.

Usage

```
ExportToFCS(
  obj,
  write_to,
  overwrite = FALSE,
  delimiter = "/",
  cytometer = "Image Stream",
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted with features extracted.

write_to	pattern used to export file. Placeholders, like "%d/%s_fromR.%e", will be substituted: -%d: with full path directory of 'obj$fileName' -%p: with first parent directory of 'obj$fileName' -%e: with extension of 'obj$fileName' (without leading .) -%s: with shortname from 'obj$fileName' (i.e. basename without extension). Exported file extension will be deduced from this pattern. Note that it has to be a .fcs.
overwrite	whether to overwrite file or not. Default is FALSE. Note that if TRUE, it will overwrite exported file if path of 'fileName' and deduced from 'write_to' arguments are different. Otherwise, you will get an error saying that overwriting source file is not allowed. Note also that an original file will never be overwritten.
delimiter	an ASCII character to separate the FCS keyword-value pairs. Default is : "/".
cytometer	string, if provided it will be used to fill $CYT keyword. However, when missing $CYT will be filled with obj$description$FCS$instrument if found, or "Image Stream" otherwise.
...	other arguments to be passed. keyword-value pairs can be passed here.

Value

invisibly returns full path of exported file.

`ExportToGallery` *Gallery Export*

Description

Exports gallery of 'IFC_img' / 'IFC_msk' objects

Usage

```
ExportToGallery(
  ...,
  objects,
  offsets,
  image_type = "img",
  layout,
  export = c("file", "matrix", "base64")[2],
  write_to,
  base64_id = FALSE,
  base64_att = "",
  overwrite = FALSE,
  main = "",
  add_channels = TRUE,
  add_ids = 1,
```

```
  add_lines = 2,
  bg_color = "grey20",
  dpi = 300,
  scale = list(),
  extract_max = 10,
  sampling = FALSE,
  display_progress = TRUE
)
```

Arguments

...	arguments to be passed to objectExtract with the exception of 'ifd' and 'bypass'(=TRUE). If 'param' is provided 'mode'(="rgb") and the above parameters will be overwritten. If 'offsets' are not provided extra arguments can also be passed with ... getOffsets. /!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.
objects	integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.
offsets	object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.
image_type	image_type of desired offsets. Either "img" or "msk". Default is "img".
layout	a character vector of [acquired channels + 'composite' images] members to export. Default is missing to export everything. Note that members can be missing to be removed from final gallery export. Note that members not found will be automatically removed and a warning will be thrown.
export	export format. Either "file", "matrix", "base64". Default is "matrix".
write_to	used when 'export' is "file" or "base64" to compute respectively filename or base64 id attribute. Exported type will be deduced from this pattern. Allowed export are '.bmp', '.jpg', '.jpeg', '.png', '.tif', '.tiff'. Note that '.bmp' are faster but not compressed producing bigger data. Placeholders, if found, will be substituted: -%d: with full path directory -%p: with first parent directory -%e: with extension (without leading .) -%s: with shortname (i.e. basename without extension) -%o: with objects (at most 10, will be collapse with "_", if more than one). -%c: with channel_id (will be collapse with "_", if more than one, composite in any will be bracketed). A good trick is to use: -"%d/%s_gallery_Obj[%o]_Ch[%c].tiff", when 'export' is "file" -"%s_gallery.bmp", when 'export' is "base64". Note that if missing and 'export' is not "file", 'write_to' will be set to "%s_gallery.bmp".
base64_id	whether to add id attribute to base64 exported object. Default is TRUE. Only applied when 'export' is "base64".

`base64_att` attributes to add to base64 exported object. Default is "".
Only applied when export is "base64". For example, use "class=draggable".
Note that id (if base64_id is TRUE) and width and height are already used.

`overwrite` whether to overwrite file or not. Default is FALSE.

`main` main title that will be displayed on top center of the image. If too large it will be clipped.

`add_channels` whether to add channels names. Default is TRUE.

`add_ids` integer, indice of column to mark objects ids number. Default is 1. If add_ids < 1, no ids are added.

`add_lines` integer, size of separating lines between objects. Default is 1. If add_lines < 1, no separating lines are added.

`bg_color` background color for main, channels and separating lines. Default is "grey20".

`dpi` integer, the resolution of the image in DPI (dots per inch). Default is 300.
Please note that whetever this parameter is final resolution will be 96 dpi.
However image will be scaled according this parameter and magnification factor will be equal to this parameter divided by 96.

`scale` a named list whose members are 'size', 'style', 'color', 'xoff', 'yoff'. Default is list() to draw no scale. Otherwise,
-'size' positive integer. Scale's bar size in micro-meter. Default is '7'.
This parameter can't be lesser than 6px nor higher than image width + scale text.
-'style' a character string. Scale's bar style, either "dash" or "line". Default is "dash".
-'color' a character string. color of the scale. Default is "white".
-'xoff' positive integer. x offset in image to draw scale, starting from bottom left corner.
-'yoff' positive integer. y offset in image to draw scale, starting from bottom left corner.

`extract_max` maximum number of objects to extract. Default is 10. Use +Inf to extract all.

`sampling` whether to sample objects or not. Default is FALSE.

`display_progress` whether to display a progress bar. Default is TRUE.

Details

arguments of `objectExtract` will be deduced from `ExportToGallery` input arguments. TRICK: for exporting only ONE 'objects', set 'add_channels' = FALSE, 'add_ids' >= 1, 'force_width' = FALSE, 'dpi' = 96; this allows generating image with its original size incrusted with its id number.

Value

Depending on 'export':
-"matrix", a rgb array,
-"base64", a data-uri string,
-"file", an invisible vector of ids corresponding to the objects exported.

ExportToNumpy *Numpy Export*

Description

Exports IFC objects to numpy array [objects,height,width,channels]

Usage

```
ExportToNumpy(
  ...,
  objects,
  offsets,
  image_type = "img",
  size = c(64, 64),
  force_width = FALSE,
  display_progress = TRUE,
  python = Sys.getenv("RETICULATE_PYTHON"),
  dtype = c("uint8", "int16", "uint16", "double")[3],
  mode = c("raw", "gray")[1],
  export = c("file", "matrix")[2],
  write_to,
  overwrite = FALSE
)
```

Arguments

...	arguments to be passed to objectExtract with the exception of 'ifd' and 'bypass'(=TRUE). If 'param' is provided the above parameters will be overwritten. If 'offsets' are not provided extra arguments can also be passed with ... getOffsets. /!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.
objects	integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.
offsets	object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.
image_type	image_type of desired offsets. Either "img" or "msk". Default is "img".
size	a length 2 integer vector of final dimensions of the image, height 1st and width 2nd. Default is c(64,64).
force_width	whether to use information in 'info' to fill size. Default is FALSE. When set to TRUE, width of 'size' argument will be overwritten.
display_progress	whether to display a progress bar. Default is TRUE.

python	path to python. Default is Sys.getenv("RETICULATE_PYTHON"). Note that this numpy should be available in this python to be able to export to numpy array file, otherwise 'export' will be forced to "matrix".
dtype	desired array?s data-type. Default is "double". Allowed are "uint8", "int16", "uint16" or "double". If 'mode' is "raw", this parameter will be forced to "int16".
mode	(objectParam argument) color mode export. Either "raw", "gray" . Default is "raw".
export	export format. Either "file", "matrix". Default is "matrix". Note that you will need 'reticulate' package installed to be able to export to numpy array file, otherwise 'export' will be forced to "matrix".
write_to	used when 'export' is "file" to compute respectively filename. Exported type will be deduced from this pattern. Allowed export are '.npy'. Placeholders, if found, will be substituted: -%d: with full path directory -%p: with first parent directory -%e: with extension of (without leading .) -%s: with shortname (i.e. basename without extension) -%o: with objects (at most 10, will be collapse with "_", if more than one). -%c: with channel_id (will be collapse with "_", if more than one, composite in any will be bracketed). A good trick is to use: -"%d/%s_Obj[%o]_Ch[%c].npy", when 'export' is "file"
overwrite	whether to overwrite file or not. Default is FALSE.

Details

arguments of objectExtract will be deduced from ExportToNumpy input arguments.
Please note that size parameter has to be supplied and could not be set to (0,) when 'object' length is not equal to one
ExportToNumpy requires reticulate package, python and numpy installed to create npy file.
If one of these is missing, 'export' will be set to "matrix".

Value

Depending on 'export':
-"matrix", an array whose dimensions are [object, height, width, channel].
-"file", it invisibly returns path of .npy exported file.

ExportToReport *Graphical and Statistic Report Generation*

Description

Generates report from 'IFC_data' object.

Usage

```
ExportToReport(
  obj,
  selection,
  write_to,
  overwrite = FALSE,
  onepage = TRUE,
  color_mode = c("white", "black")[1],
  add_key = "panel",
  precision = c("light", "full")[1],
  trunc_labels = 38,
  trans = "asinh",
  bin,
  viewport = "ideas",
  display_progress = TRUE,
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted with features extracted.
selection	indices of desired graphs. It can be provided as an integer vector or as a matrix. In such case, the layout of the matrix will reflect the layout of the extracted graphs. NA value will result in an empty place. Otherwise, when 'selection' is provided as a vector not identical to seq_along(obj$graphs), 'onepage' parameter will be set to FALSE. Note that indices are read from left to right, from top to bottom. Default is missing for extracting all graphs.
write_to	pattern used to export file(s). Placeholders, like c("%d/%s_fromR.pdf", "%d/%s_fromR.csv"), will be substituted: -%d: with full path directory of 'obj$fileName' -%p: with first parent directory of 'obj$fileName' -%e: with extension of 'obj$fileName' (without leading .) -%s: with shortname from 'obj$fileName' (i.e. basename without extension). Exported file(s) extension(s) will be deduced from this pattern. Note that has to be a .pdf and/or .csv.
overwrite	whether to overwrite file or not. Default is FALSE. Note that if TRUE, it will overwrite file. In addition a warning message will be sent.
onepage	whether to generate a pdf with all graphs on one page or not. Default is TRUE.
color_mode	Whether to extract colors from obj in white or black mode. Default is "white".
add_key	whether to draw a "global" key under title or in the first "panel" or "both". Default is "panel". Accepted values are either: FALSE, "panel", "global", "both" or c("panel", "global"). Note that it only applies when display is seen as overlaying populations.

precision	when graphs is a 2D scatter with population overlay, this argument controls amount of information displayed. Default is "light". -"light", the default, will only display points of same coordinates that are amoung the other layers. -"full" will display all the layers.
trunc_labels	maximum number of characters to display for labels. Default is 38.
trans	name of the transformation function for density graphs. If missing the default, the BasePop[[1]]$densitytrans, if any, will be retrieved, otherwise "asinh" will be used.
bin	default number of bin used for histogram. Default is missing.
viewport	Either "ideas", "data" or "max" defining limits used for the graph. Default is "ideas". -"ideas" will use same limits as the one defined in ideas. -"data" will use data to define limits. -"max" will use data and regions drawn to define limits.
display_progress	whether to display a progress bar. Default is TRUE.
...	other parameters to be passed.

Details

depending on 'write_to', function will create .pdf and/or .csv file(s) report with according to graphs found in 'obj'.
- csv file if created will contain "Min.","1st Qu.","Median","Mean","3rd Qu.","Max." for each graph found for x and y (if not histogram) for drawn populations and regions.
- pdf file if created will contain graphs and to a certain extent some stats "Min.", "Median", "Mean", "Max." (no more than 7 rows).
Note that only graphs will be exported (no images, features values, population stats, ...) in the same layout they were created and without sizing.

Value

It invisibly returns full path of exported .pdf and/or .csv file(s).

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  tmp <- tempdir(check = TRUE)
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf, extract_images = FALSE,
                        extract_offsets = FALSE, display_progress = FALSE)
  L = length(daf$graphs)
  if(L > 0) {
    ## randomly export at most 5 graphs from daf
    sel = sample(1:L, min(5, L))
    ExportToReport(obj = daf, selection = sel,
                   write_to = paste0(tmp, "\\test.pdf"), overwrite = TRUE)
```

```
  }
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

ExportToXIF *RIF/CIF File Writer*

Description

Subsets or merges RIF or CIF files.

Usage

```
ExportToXIF(
  fileName,
  write_to,
  objects,
  offsets,
  fast = TRUE,
  extract_features = FALSE,
  endianness = .Platform$endian,
  verbose = FALSE,
  verbosity = 1,
  overwrite = FALSE,
  display_progress = TRUE,
  add_tracking = TRUE,
  ...
)
```

Arguments

fileName	path(s) of file(s) to subset or merge. If multiple files are provided they will be merged. Otherwise, if only one file is input it will be subsetted. All files have to be either '.rif' or '.cif' files. All files should have same channels.
write_to	pattern used to export file. Placeholders, like "%d/%s_fromR.%e", will be substituted: -%d: with full path directory of first element of 'fileName' -%p: with first parent directory of first element of 'fileName' -%e: with extension of first element of 'fileName' (without leading .) -%s: with shortname from of first element of 'fileName' (i.e. basename without extension). Exported file extension will be deduced from this pattern. It has to be the same as 'fileName', i.e. .cif or .rif.

`objects` integer vector, IDEAS objects ids numbers to use. If missing, the default, all objects will be used. Only apply for subsetting.

`offsets` object of class 'IFC_offset'. If missing, the default, offsets will be extracted from 'fileName'.
This param is not mandatory but it may allow to save time for repeated XIF export on same file. Only apply for subsetting.

`fast` whether to fast extract 'objects' or not. Default is TRUE. Meaning that 'objects' will be extracting expecting that objects are stored in ascending order.
Note that a warning will be sent if an 'object' is found at an unexpected order. In such a case you may need to rerun function with 'fast' = FALSE. If set to FALSE, all object_ids will be scanned from 'fileName' to ensure extraction of desired 'objects'.
IMPORTANT: whatever this argument is, features are extracted assuming an ascending order of storage in file.
Only apply for subsetting.

`extract_features` whether to try to extract features. Default is FALSE. IMPORTANT: it is not clear if how features are stored and which objects they rely to when input file is already a merge or a subset. For this reason it should be carefully checked. Note that features extraction is not implemented for merging.

`endianness` the endian-ness ("big" or "little") of the target system for the file. Default is .Platform$endian.
Endianness describes the bytes order of data stored within the files. This parameter may not be modified.

`verbose` whether to display information (use for debugging purpose). Default is FALSE.

`verbosity` quantity of information displayed when verbose is TRUE; 1: normal, 2: rich. Default is 1.

`overwrite` whether to overwrite file or not. Default is FALSE.
Note that if TRUE, it will overwrite exported file if path(s) of file(s) in 'fileName' and deduced from 'write_to' arguments are different. Otherwise, you will get an error saying that overwriting source file is not allowed.
Note also that an original file, i.e. generated by IDEAS(R) or INSPIRE(R), will never be overwritten. Otherwise, you will get an error saying that overwriting original file is not allowed.

`display_progress` whether to display a progress bar. Default is TRUE.

`add_tracking` whether to register files' paths and objects' ids in the exported file. Default is TRUE.

`...` other arguments to be passed.

Details

when 'extract_features' is set TRUE, only features stored in binary format will be extracted if found. If the input 'fileName' is a merged of several files then features will be extracted from these files. If these files can't be found, Warning(s) will be thrown and input 'fileName' will be extracted without features values.

Value

It invisibly returns full path of exported file.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  tmp <- tempdir(check = TRUE)
  ## use a cif file, but you can also subset rif
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  ## subset objects 0,1 and 4 from file
  exported <- ExportToXIF(fileName = file_cif, write_to = paste0(tmp, "\\test.cif"),
                          overwrite = TRUE, objects = c(0,1,4))
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

ExtractFromDAF *DAF File Reader*

Description

Extracts data from DAF Files.

Usage

```
ExtractFromDAF(
  fileName,
  extract_features = TRUE,
  extract_images = TRUE,
  extract_offsets = TRUE,
  extract_stats = TRUE,
  endianness = .Platform$endian,
  pnt_in_poly_algorithm = 1,
  pnt_in_poly_epsilon = 1e-12,
  display_progress = TRUE,
  ...
)
```

Arguments

fileName	path to file.
extract_features	whether to extract features (and graphs, pops and regions) from file. Default is TRUE.
extract_images	whether to extract images information from file. Default is TRUE.

`extract_offsets`
: whether to extract IFDs offsets from corresponding. Default is TRUE.
See `getOffsets` for further details.

`extract_stats`
: whether to extract population statistics. Default is TRUE.

`endianness`
: The endian-ness ("big" or "little") of the target system for the file. Default is .Platform$endian.
Endianness describes the bytes order of data stored within the files. This parameter may not be modified.

`pnt_in_poly_algorithm`
: algorithm used to determine if object belongs to a polygon region or not. Default is 1.
Note that for the moment only 1(Trigonometry) is available.

`pnt_in_poly_epsilon`
: epsilon to determine if object belongs to a polygon region or not. It only applies when algorithm is 1. Default is 1e-12.

`display_progress`
: whether to display a progress bar. Default is TRUE.

`...`
: Other arguments to be passed.

Details

When extract_features is TRUE it allows features, graphs, pops, regions to be extracted.
If extract_features is TRUE, extract_stats will be automatically forced to TRUE.
If extract_stats is TRUE, extract_features will be automatically forced to TRUE.
If extract_offsets is TRUE, extract_images will be automatically forced to TRUE.
If extract_images is TRUE, information about images will be extracted.

Value

A named list of class 'IFC_data', whose members are:
-description, a list of descriptive information,
-fileName, path of fileName input,
-fileName_image, path of .cif image fileName is refering to,
-features, a data.frame of features,
-features_def, a describing how features are defined,
-graphs, a list of graphical elements found,
-pops, a list describing populations found,
-regions, a list describing how regions are defined,
-images, a data.frame describing information about images,
-offsets, an integer vector of images and masks IFDs offsets,
-stats, a data.frame describing populations count and percentage to parent and total population,
-checksum, checksum of .cif image fileName is refering to computed from images values found in current daf.

Source

For pnt_in_poly_algorithm, Trigonometry, is an adaptation of Jeremy VanDerWal's code https://github.com/jjvanderwal/SDMTools

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`ExtractFromFCS` *FCS File Reader*

Description

Extracts data from Flow Cytometry Standard (FCS) Files.

Usage

```
ExtractFromFCS(fileName, ...)
```

Arguments

`fileName`	path(s) of file(s). If multiple files are provided they will be merged and populations will be created to identify each single file within returned 'IFC_data' object.
`...`	other arguments to be passed to readFCS function, with the exception of 'options$text_only'.

Value

A named list of class 'IFC_data', whose members are:
-description, a list of descriptive information,
-Merged_fcs, character vector of path of files used to create fcs, if input was a merged,
-fileName, path of fileName input,
-fileName_image, path of .cif image fileName is referring to,
-features, a data.frame of features,
-features_def, a describing how features are defined,
-graphs, a list of graphical elements found,
-pops, a list describing populations found,
-regions, a list describing how regions are defined,
-images, a data.frame describing information about images,
-offsets, an integer vector of images and masks IFDs offsets,
-stats, a data.frame describing populations count and percentage to parent and total population,
-checksum, a checksum integer.

Source

Data File Standard for Flow Cytometry, version FCS 3.1 from Spidlen J. et al. available at doi: 10.1002/cyto.a.20825.

`ExtractFromXIF` *RIF/CIF File Reader*

Description

Extracts data from RIF or CIF Files.

Usage

```
ExtractFromXIF(
  fileName,
  extract_features = TRUE,
  extract_images = FALSE,
  extract_offsets = FALSE,
  extract_stats = TRUE,
  pnt_in_poly_algorithm = 1,
  pnt_in_poly_epsilon = 1e-12,
  force_default = TRUE,
  verbose = FALSE,
  verbosity = 1,
  display_progress = TRUE,
  fast = TRUE,
  recursive = FALSE,
  ...
)
```

Arguments

`fileName`	path to file.
`extract_features`	whether to extract features from file. Default is TRUE. If TRUE, `ExtractFromXIF` will try to export features. It it fails a message will be sent. Otherwise, graphs, pops and regions will be also extracted.
`extract_images`	whether to extract images information from file. Default is FALSE.
`extract_offsets`	whether to extract IFDs offsets from corresponding. Default is FALSE. See `getOffsets` for further details.
`extract_stats`	whether to extract population statistics. Default is TRUE.
`pnt_in_poly_algorithm`	algorithm used to determine if object belongs to a polygon region or not. Default is 1. Note that for the moment only 1(Trigonometry) is available.

`pnt_in_poly_epsilon`
: epsilon to determine if object belongs to a polygon region or not. It only applies when algorithm is 1. Default is 1e-12.

`force_default`
: when display information can't be retrieved whether to use default values. Default is TRUE.

`verbose`
: whether to display information (use for debugging purpose). Default is FALSE.

`verbosity`
: quantity of information displayed when verbose is TRUE; 1: normal, 2: rich. Default is 1.

`display_progress`
: whether to display a progress bar. Default is TRUE.

`fast`
: whether to fast 'extract_offsets' or not. Default is TRUE.
Meaning that offsets will be extracting expecting that raw object are stored in ascending order. if extract_images is FALSE, a message will be thrown since extraction method does not ensure correct mapping between objects and offsets. if extract_images is TRUE, a warning will be sent if an object is found at an unexpected order.

`recursive`
: whether to recursively apply `ExtractFromXIF` on files defining input fileName when it is a merged. Default is FALSE.

`...`
: Other arguments to be passed.

Details

If extract_stats is TRUE, extract_features will be automatically forced to TRUE.
If extract_images is TRUE, extract_offsets will be automatically forced to TRUE.
If extract_offsets is TRUE, offsets of images and masks IFDs will be extracted.
If extract_images is TRUE, information about images will be extracted.
If the input fileName is a merged of several files and recursive is set to TRUE, then ExtractFromXIF will be applied recursively on these files.
/!\ Note that features extraction is mandatory to correctly extract graphs, pops, regions and statistics values.

Value

A named list of class 'IFC_data', whose members are:
-description, a list of descriptive information,
-fileName, path of fileName input,
-fileName_image, same as fileName,
-features, a data.frame of features,
-features_def, a describing how features are defined,
-graphs, a list of graphical elements found,
-pops, a list describing populations found,
-regions, a list describing how regions are defined,
-images, a data.frame describing information about images,
-offsets, an integer vector of images and masks IFDs offsets,
-stats, a data.frame describing populations count and percentage to parent and total population,
-checksum, current file checksum.

If fileName is a merged of several files returned object will be of class ‘IFC_data‘ and ‘Merged‘. If recursive is set to "TRUE", ExtractFromXIF will be applied recursively on files defining the merged. and the returned object will be a list of the above-mentionned list for each of these files.

Source

For pnt_in_poly_algorithm, Trigonometry, is an adaptation of Jeremy VanDerWal's code https://github.com/jjvanderwal/SDMTools

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file, but you can also read rif
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  cif <- ExtractFromXIF(fileName = file_cif)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

ExtractImages_toBase64

Shorcut for Batch Images Extraction to Base64

Description

Function to shortcut extraction, normalization and eventually colorization of images to matrix ! excludes mask.

Usage

```
ExtractImages_toBase64(
  ...,
  objects,
  offsets,
  display_progress = TRUE,
  mode = c("rgb", "gray")[1]
)
```

Arguments

...	arguments to be passed to objectExtract with the exception of 'ifd' and 'bypass'(=TRUE). If 'param' is provided 'export'(="base64") and 'mode' will be overwritten. If 'offsets' are not provided extra arguments can also be passed with ... getOffsets. /!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.

objects	integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.
offsets	object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.
display_progress	whether to display a progress bar. Default is TRUE.
mode	(objectParam argument) color mode export. Either "rgb", "gray". Default is "rgb".

Details

arguments of objectExtract will be deduced from ExtractImages_toBase64 input arguments.

Value

A list of base64 encoded images corresponding to objects extracted.

ExtractImages_toFile *Shorcut for Batch Images Extraction to Files*

Description

Function to shortcut extraction, normalization and eventually colorization of images to matrix ! excludes mask.

Usage

```
ExtractImages_toFile(
  ...,
  objects,
  offsets,
  display_progress = TRUE,
  mode = c("rgb", "gray")[1],
  write_to
)
```

Arguments

...	arguments to be passed to objectExtract with the exception of 'ifd' and 'bypass'(=TRUE). If 'param' is provided 'export'(="file"), 'write_to' and 'mode' will be overwritten. If 'offsets' are not provided extra arguments can also be passed with ... getOffsets. /!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.
objects	integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.

`offsets` object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.

`display_progress` whether to display a progress bar. Default is TRUE.

`mode` (`objectParam` argument) color mode export. Either "rgb", "gray" . Default is "rgb".

`write_to` (`objectParam` argument) used to compute exported file name.
Exported "file" extension will be deduced from this pattern. Allowed export are '.bmp', '.jpg', '.jpeg', '.png', '.tif', '.tiff'. Note that '.bmp' are faster but not compressed producing bigger data.
Placeholders, if found, will be substituted:
-%d: with full path directory
-%p: with first parent directory
-%e: with extension (without leading .)
-%s: with shortname (i.e. basename without extension)
-%o: with object_id
-%c: with channel_id
A good trick is to use "%d/%s/%s_%o_%c.tiff".

Details

arguments of `objectExtract` will be deduced from `ExtractImages_toFile` input arguments.

Value

It invisibly returns a list of exported file path of corresponding to objects extracted.

ExtractImages_toMatrix

Shorcut for Batch Images Extraction to Matrices/Arrays

Description

Function to shortcut extraction, normalization and eventually colorization of images to matrix ! excludes mask.

Usage

```
ExtractImages_toMatrix(..., objects, offsets, display_progress = TRUE)
```

Arguments

`...` arguments to be passed to `objectExtract` with the exception of 'ifd' and 'bypass'(=TRUE).
If 'param' is provided 'export'(="matrix") will be overwritten.
If 'offsets' are not provided extra arguments can also be passed with ... `getOffsets`.
/!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to `objectParam`.

objects — integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.

offsets — object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.

display_progress — whether to display a progress bar. Default is TRUE.

Details

arguments of objectExtract will be deduced from ExtractImages_toMatrix input arguments.

Value

A list of matrices/arrays of images corresponding to objects extracted.

ExtractMasks_toMatrix *Shorcut for Batch Masks Extraction to Matrices/Arrays*

Description

Function to shortcut extraction, normalization and eventually colorization of masks to matrix ! excludes image.

Usage

```
ExtractMasks_toMatrix(..., objects, offsets, display_progress = TRUE)
```

Arguments

... — arguments to be passed to objectExtract with the exception of 'ifd' and 'bypass'(=TRUE).
If 'param' is provided 'export'(="matrix") will be overwritten.
If 'offsets' are not provided extra arguments can also be passed with ... getOffsets.
/!\ If not any of 'fileName', 'info' and 'param' can be found in ... then attr(offsets, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.

objects — integer vector, IDEAS objects ids numbers to use. This argument is not mandatory, if missing, the default, all objects will be used.

offsets — object of class 'IFC_offset'. This argument is not mandatory but it may allow to save time for repeated image export on same file.

display_progress — whether to display a progress bar. Default is TRUE.

Details

arguments of objectExtract will be deduced from ExtractMasks_toMatrix input arguments.

Value

A list of matrices/arrays of masks corresponding to objects extracted.

`getAborted` *Aborted Batch Files Retrieval*

Description

Try to retrieve files whose processing failed during batch. This is a very beta version

Usage

```
getAborted(aborted, default_batch_dir, config_file)
```

Arguments

`aborted`	path to file containing aborted information. If missing, the default, a dialog box will be displayed to choose this file. Note, that if provided 'default_batch_dir' and 'config_file' will not be used.
`default_batch_dir`	directory where batches are stored. It can be found in IDEAS(R) software, under Options -> Application Defaults -> Directories -> Default Batch Report Files Directory. If missing, the default, it will be deduced from IDEAS(R) config file, However, if it can't be deduced then current working directory will be used. This argument takes precedence over 'config_file' and filling 'default_batch_dir' prevents the use of 'config_file' argument.
`config_file`	path to IDEAS(R) config file. It may depends on IDEAS(R) software installation but one may use "C:/Users/%USER%/AppData/Roami Corporation/userconfig.xml".

Value

a list of 4 elements:
-not_existing: a list of files paths that caused failure because they were not found during batch,
-not_handled: a list of failed files and the retrieved error message.
-failed_found: a list of failed files and their unique corresponding paths,
-failed_match: a list of failed files and their all paths that could match.

getFullTag *Image Field Directory Full Tag Retrieval*

Description

Retrieves full tag value from IFDs (Image Field Directory) extracted by `getIFD`.

Usage

```
getFullTag(IFD, which = 1, tag = "256", raw = FALSE)
```

Arguments

`IFD`	an object of class 'IFC_ifd_list' extracted by `getIFD`.
`which`	scalar, integer (index) or the name of 'IFD' sub-element to extract 'tag' from. Default is 1 to extract 'tag' from the first member of 'IFD'.
`tag`	scalar, integer (index) or the name of the IFD[[which]] of the desired 'tag'.
`raw`	whether to return tag as a raw vector. Default is FALSE.

Details

It may be usefull to extract all information contained in a specific 'tag' since `getIFD` is designed to be run with argument trunc_bytes so as to only extract essential bytes to run faster and save memory. Nonetheless, thanks to `getFullTag` users will still be able to get full extraction of specific tag.

Value

the full value of the corresponding IFD tag.

Source

TIFF 6.0 specifications archived from web https://web.archive.org/web/20211209104854/https://www.adobe.io/open/standards/TIFF.html

getIFD *RIF/CIF Image Field Directories Extraction*

Description

Extracts IFDs (Image File Directory) in RIF or CIF files.
IFDs contain information about images or masks of objects stored within XIF files.
The first IFD is special in that it does not contain image of mask information but general information about the file.
Users are highly encouraged to read TIFF specifications to have a better understanding about what IFDs are.

Usage

```
getIFD(
  fileName,
  offsets = "first",
  trunc_bytes = 12,
  force_trunc = FALSE,
  verbose = FALSE,
  verbosity = 1,
  display_progress = FALSE,
  bypass = FALSE,
  ...
)
```

Arguments

fileName	path to file.
offsets	either "all", "first" or an object of class 'IFC_offset'. Default is "first".
trunc_bytes	a positive integer maximal number of individual scalar to extract BYTE/ASCII/SBYTE/UNDIFINED for TAGS (1, 2, 6 or 7). Default is 12. However, if less is found, less is returned in map. Note that, if 0 is provided, it will be automatically set to 1.
force_trunc	whether to force truncation for all TAGS types. Default is FALSE. If TRUE, 'trunc_bytes' will be used for TAGS (3, 4, 5, 8, 9, 10, 11 and 12) to extract desired number of individual scalar corresponding to each types.
verbose	whether to display information (use for debugging purpose). Default is FALSE.
verbosity	quantity of information displayed when verbose is TRUE; 1: normal, 2: rich. Default is 1.
display_progress	whether to display a progress bar. Default is FALSE.
bypass	whether to bypass checks on 'trunc_bytes', 'force_trunc', 'verbose', 'verbosity' and 'display_progress'. Default is FALSE.
...	other arguments to be passed.

Details

Function will return IFDs (image, mask or first) from the file using provided offsets argument.
IFDs contain several tags that can be viewed as descriptive meta-information of raw data stored within RIF or CIF file. For more details see TIFF specifications.
If 'offsets' == "first" only first IFD will be returned.
If 'offsets' == "all" all images and masks IFDs will be returned but not "first" one. Be aware that errors may occur if offsets are not extracted with `getOffsets` or `subsetOffsets`.

Value

A list of named lists, each containing:
-tags, a named list whose names are tags found, where each tag is a list of tag, typ, siz, val, byt, len,

off, map information.
-infos, a named list containing essential information about IFDs, IMAGE_LENGTH, IMAGE_WIDTH, OBJECT_ID, COMPRESSION, TYPE, STRIP_OFFSETS, STRIP_BYTE_COUNTS, BG_MEAN, BG_STD
-curr_IFD_offset, the position of current IFD offset
-next_IFD_offset, the position of next IFD offset

Source

TIFF 6.0 specifications archived from web https://web.archive.org/web/20211209104854/https://www.adobe.io/open/standards/TIFF.html

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  ## read 1st IFD
  IFD_first <- getIFD(fileName = file_cif, offsets = "first")
  ## show information contained in 1st IFD
  print(sapply(IFD_first[[1]]$tags, FUN=function(x) x))
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

---

getInfo *IFC File Information Extraction*

Description

Retrieves rich information from RIF, CIF and DAF files.

Usage

```
getInfo(
  fileName,
  from = c("acquisition", "analysis")[2],
  verbose = FALSE,
  verbosity = 1,
  warn = TRUE,
  force_default = TRUE,
  cifdir = dirname(fileName),
  ntry = +Inf,
  ...
)
```

Arguments

fileName	path to file..
from	whether to extract information from 'acquisition' or 'analysis'. Default is 'analysis'.
verbose	whether to display information (use for debugging purpose). Default is FALSE.
verbosity	quantity of information print to console when verbose is TRUE; 1: normal, 2: rich. Default is 1.
warn	whether to send warning message when trying to read 'analysis' information from a 'rif' file. Default is TRUE.
force_default	when display information can't be retrieved whether to use default values. Default is TRUE.
cifdir	the path of the directory to initially look to cif file. Default is dirname(fileName). Only apply when 'fileName' is a .daf file.
ntry	number of times getInfo will be allowed to find corresponding cif file. Default is +Inf. Only apply when 'fileName' is a .daf file. If cif can't be found, but 'ntry' is reached, then an error will be thrown.
...	other arguments to be passed.

Value

a list of information (open .daf file in an text editor for more details) about input fileName of class 'IFC_info' and 'acquistion' or 'analysis', whose members are:
-objcount, number of object in file,
-date, date of file creation,
-instrument, instrument identification,
-sw_raw, version of software for raw data,
-sw_processed, version of software for processed data,
-channelwidth, default channel width in pixel,
-in_use, channels used,
-brightfield, whether brightfield is applied on channels and its intensity,
-illumination, laser illumination parameters,
-collectionmode, the collection mode,
-magnification, magnification used,
-coremode, the core mode,
-evmode, the high gain mode,
-CrossTalkMatrix. compensation matrix applied,
-ChannelPresets, channel preset,
-ImageDisplaySettings, image display settings,
-Images, information about colors, range and channels,
-masks, masks defined,
-ViewingModes, modes of visualization,
-checksum, checksum computed,
-Merged_rif, character vector of path of files used to create rif, if input file was a merged,
-Merged_cif, character vector of path of files used to create cif, if input file was a merged,
-XIF_test, integer defining XIF type,
-checksum, integer corresponding to file checksum,

-fileName, path of fileName input,
-fileName_image, path of fileName_image.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  info <- getInfo(fileName = file_daf, from = "analysis")
  ## show some information
  print(info$Images)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`getOffsets` *RIF/CIF File Image Field Directories Offsets Extraction*

Description

Extracts offsets of the IFDs (Image Field Directories) within a XIF file. Users are highly encouraged to read TIFF specifications to have a better understanding about what offsets and IFDs are.

Usage

```
getOffsets(fileName, fast = TRUE, display_progress = TRUE, verbose = FALSE)
```

Arguments

`fileName`	path to file.
`fast`	whether to fast extract objects or not. Default is TRUE. Meaning that offsets will be extracting expecting that objects are stored in ascending order. A message will be thrown since fast extraction method does not ensure correct mapping between objects and offsets. If set to FALSE, all object_ids will be scanned from 'fileName' to ensure extraction of desired offsets.
`display_progress`	whether to display a progress bar. Default is TRUE.
`verbose`	whether to display information (use for debugging purpose). Default is FALSE.

Details

Offsets are byte positions of IFDs found within RIF or CIF file. For more details see TIFF specifications.

Value

an integer vector of class 'IFC_offset' of IFDs offsets found in XIF file.

Source

TIFF 6.0 specifications archived from web https://web.archive.org/web/20211209104854/https://www.adobe.io/open/standards/TIFF.html

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  system.time(offsets_fast <- getOffsets(fileName = file_cif, fast = TRUE))
  system.time(offsets_slow <- getOffsets(fileName = file_cif, fast = FALSE))
  identical(offsets_fast, offsets_slow)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

---

inv_smoothLinLog *Inverse Smooth LinLog Transformation*

Description

Gets values back just to their original values before applying smoothLinLog.

Usage

```
inv_smoothLinLog(x, hyper = 1000, base = 10, lin_comp = log(base))
```

Arguments

x	A numeric vector.
hyper	value where transition between Lin/Log is applied.
base	base of Log scale.
lin_comp	value that is used to smooth transition between Lin/Log. Default is log(base).

Value

the inverse smoothLinLog transformation of the input.

objectCleanse *Object Cleanser*

Description

Removes abnormalities (clipped/debris) from image.

Usage

```
objectCleanse(mat, msk, add_noise = TRUE, random_seed = NULL, bg = 0, sd = 0)
```

Arguments

mat	a numeric matrix (image).
msk	a numeric matrix (mask identifying abnormalities).
add_noise	if TRUE adds normal noise to background using rnorm(), from **Rcpp**. Default is TRUE.
random_seed	a single value, interpreted as an integer, or NULL to be used with set.seed() from **base** when 'add_noise' is set to TRUE. Default is NULL.
bg	mean value of the background added if add_noise is TRUE. Default is 0.
sd	standard deviation of the background added if add_noise is TRUE. Default is 0.

Value

According to msk, pixel values in mat are substituted by either bg [add_noise == FALSE] or rnorm(n = prod(dim(mat), mean=bg, sd=sd)) [add_noise == TRUE].

objectDisplay *Object Display*

Description

This function is intended to display object extracted by objectExtract.

Usage

```
objectDisplay(
  image,
  input_range = c(0, 4095),
  full_range = FALSE,
  force_range = FALSE,
  gamma = 1,
  color = "Green",
  dpi = 300
)
```

Arguments

image	An object extracted by objectExtract of class 'IFC_img' or 'IFC_msk'. Note that a matrix with finite values can also be used.
input_range	a finite numeric vector of 2 values, sets the range of the input intensity values. Values exceeding this range are clipped. Default is 'c(0, 4095)'.
full_range	if 'full_range' is TRUE, then 'input_range' will be set to 'c(0, 4095)' and 'gamma' forced to 1. Default is FALSE.
force_range	if 'force_range' is TRUE, then 'input_range' will be adjusted to object range in [-4095, +inf] and 'gamma' forced to 1. Default is FALSE. Note that this parameter takes the precedence over 'input_range' and 'full_range'.
gamma	gamma correction. Default is 1, for no correction.
color	a color. Default is "Green".
dpi	display resolution. Default is 300.

Details

If input 'image' is of class 'IFC_img' or 'IFC_msk', then if 'input_range', 'full_range', 'force_range', 'gamma' and / or 'color' parameters is/are missing, it/they will be extracted from 'image' attributes.
If input 'image' is not of one of class 'IFC_img' or 'IFC_msk', then force_range will be forced to TRUE.
An error will be thrown if input image contains non finite values.

Value

it invisibly returns NULL

---

objectExtract *Object Extraction*

Description

Extracts / Decompress objects stored in RIF or CIF Files.

Usage

```
objectExtract(ifd, param, verbose = FALSE, bypass = FALSE, ...)
```

Arguments

ifd	list of sub elements of IFD data information extracted by getIFD. This parameter can't be missing.
param	object of class 'IFC_param', containing extraction parameters defined by objectParam. This argument is not mandatory but it may allow to save time for repeated image export on same file. If this parameter is missing, objectExtract will use extra ... to pass arguments to objectParam to control object extraction. However, if 'param' is provided, '...' will be ignored.

verbose	whether to display information (use for debugging purpose). Default is FALSE.
bypass	whether to bypass checks on 'ifd' and 'param'. Default is FALSE.
...	other arguments to be passed to objectParam. If 'param' is not provided then '...' will be used to compute 'param'. /!\ If not any of 'fileName', 'info' can be found in '...' then attr(ifd, "fileName_image") will be used as 'fileName' input parameter to pass to objectParam.

Value

A list (for every extracted objects) of list (for every exported channels) depending on "export" parameter:
-"matrix", a matrix when 'mode' is set to "raw" or "gray" OR an array when 'mode' == "rgb",
-"base64", a data-uri string,
-"file", an invisible file path corresponding to the location of exported file(s).

Source

For image decompression, Lee Kamentsky's code porting from https://github.com/ome/bioformats/blob/4146b9a1797501f0fec7d6cfe69124959bff96ee/components/formats-bsd/src/loci/formats/in/FlowSightReader.java
cited in https://linkinghub.elsevier.com/retrieve/pii/S1046-2023(16)30291-2

BSD implementations of Bio-Formats readers and writers

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  cif_offs <- getOffsets(fileName = file_cif, fast = TRUE)
  ## extract infomation
  info <- getInfo(fileName = file_cif, from = "analysis")
  ## retrieve number of objects stored
  nobj <- as.integer(info$objcount)
  ## randomly subset the offsets of at most 5 "img" objects
  sel = sample(0:(nobj-1), min(5, nobj))
  sub_offs <- subsetOffsets(cif_offs, objects = sel, image_type = "img")
  ## read IFDs from these "img" objects
  IFDs <- getIFD(fileName = file_cif, offsets = sub_offs)
  ## extract raw data of these"img" objects to matrix
  raw = objectExtract(ifd = IFDs, info = info, mode = "raw",
                      export = "matrix")
  ## extract base64 "rgb" colorized version of these "img" objects to base64
  b64 = objectExtract(ifd = IFDs, info = info, mode = "rgb",
                      export = "base64", base64_id = TRUE,
                      write_to = "example_%o_%c.png")
  ## use DisplayGallery to show the first "img" objects and play with ... extra parameters
  ## force_range, add_noise, selection, composite, see objectParam
  DisplayGallery(info = info, offsets = cif_offs, objects = sel,
                 base64_id = TRUE, write_to = "example_%o_%c.png",
                 force_range = c(FALSE,TRUE,FALSE,TRUE), add_noise = FALSE,
                 selection = c(1,2,4,6), composite = "1.7/4.3")
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

`objectParam` *Object Extraction Parameters Definition*

Description

Defines 'IFC_object' object extraction parameters.

Usage

```
objectParam(
  ...,
  info,
  mode = c("rgb", "gray", "raw")[3],
  export = c("file", "matrix", "base64")[2],
  write_to,
  base64_id = FALSE,
```

```
  base64_att = "",
  overwrite = FALSE,
  composite = "",
  selection = "all",
  size = c(0, 0),
  force_width = TRUE,
  random_seed = NULL,
  removal = "none",
  add_noise = TRUE,
  full_range = FALSE,
  force_range = FALSE,
  spatial_correction = FALSE
)
```

Arguments

...	arguments to be passed to getInfo, only if 'info' is not provided.
info	object of class 'IFC_info', rich information extracted by getInfo. This argument is not mandatory but it may allow to save time for repeated image export on same file. If missing, the default, 'info' will be extracted thanks to '...'.
mode	color mode export. Either "rgb", "gray" or "raw". Default is "raw". Note that "raw" is only possible when 'export' is "matrix".
export	format mode export. Either "file", "matrix", "base64". Default is "matrix".
write_to	used when export is "file" or "base64" to compute respectively exported file name or base64 id attribute. Exported "file" extension and "base64" MIME type will be deduced from this pattern. Allowed export are ".bmp", ".jpg", ".jpeg", ".png", ".tif", ".tiff". Note that '.bmp' are faster but not compressed producing bigger data. Placeholders, if found, will be substituted: -%d: with full path directory -%p: with first parent directory -%e: with extension (without leading .) -%s: with shortname (i.e. basename without extension) -%o: with object_id -%c: with channel_id A good trick is to use: -"%d/%s/%s_%o_%c.tiff", when 'export' is "file" -"%o_%c.bmp", when 'export' is "base64". Note that if missing and 'export' is not "file", 'write_to' will be set to "%o_%c.bmp".
base64_id	whether to add id attribute to base64 exported object. Default is FALSE. Only applied when export is "base64".
base64_att	attributes to add to base64 exported object. Default is "". Only applied when export is "base64". For example, use "class=draggable". Note that id (if base64_id is TRUE) and width and height are already used.
overwrite	only apply when 'export' is "file" whether to overwrite file or not. Default is FALSE.

`composite` character vector of image composite. Default is "", for no image composite.
Should be like "1.05/2.4/4.55" for a composition of 5 perc. of channel 1, 40 perc. of channel 2 and 50 perc. of channel 55.
Note that channels should have been acquired and final image composition should be 100 perc., otherwise an error is thrown.
Note that each composite will be appended after 'selection'.

`selection` physical channels to extract.
Note that this parameter will be ordered.
Default is "all" to extract all acquired channels.
Use "none" to only extract composite.

`size` a length 2 integer vector of final dimensions of the image, height 1st and width 2nd. Default is c(0,0) for no change.

`force_width` whether to use information in 'info' to fill size. Default is TRUE. When set to TRUE, width of 'size' argument will be overwritten.

`random_seed` a single value, interpreted as an integer, or NULL to be used with set.seed() from **base** when 'add_noise' is set to TRUE. Default is NULL.

`removal` removal method: Either "none", "raw", "clipped", "masked", "MC".
-"none", to keep image as is
-"raw", to keep image as is, it provides a convinient way to retrieve "raw" value for the mask.
-"clipped", to remove clipped object from image.
-"masked", to only keep masked object from image.
-"MC", to only keep MC masked object from image. This parameter will be repeated with rep_len() from **base** for every physical channel that needs to be extracted according to 'selection' and 'composite' parameters.

`add_noise` if TRUE adds normal noise to background using rnorm(), from **Rcpp**. Default is TRUE.
Note that it is better to set it to FALSE when 'removal' is "masked" or "MC". Doing so will allow to place masked object in a zero filled background, otherwise background will still be filled with noise. This parameter will be repeated with rep_len() from **base** for every physical channel that needs to be extracted according to 'selection' and 'composite' parameters.

`full_range` only apply when mode is not "raw", if full_range is TRUE, then [0,4095] range will be kept. Default is FALSE.
It is like "raw" mode but allowing normalization to [0,1]. This parameter will be repeated with rep_len() from **base** for every physical channel that needs to be extracted according to 'selection' and 'composite' parameters.

`force_range` only apply when mode is not "raw", if force_range is TRUE, then range will be adjusted to object range in [-4095, +inf] resulting in normalization. Default is FALSE.
Note that this parameter takes the precedence over 'full_range'.
This parameter will be repeated with rep_len() from **base** for every physical channel that needs to be extracted according to 'selection' and 'composite' parameters.

`spatial_correction` only apply on RIF file, whether to apply spatial correction. Default is FALSE.

Details

when a mask is detected, 'add_noise', 'full_range' and 'force_range' are set to FALSE.

Value

an object of class 'IFC_param'.

---

paletteIFC *R/IDEAS Color Palette Mapping*

Description

Maps colors between IDEAS and R.

Usage

```
paletteIFC(
  x = c("", "palette", "palette_R", "to_light", "to_dark")[1],
  col = "White"
)
```

Arguments

x	either "", "palette","palette_R", to_light, to_dark. Default is "".
col	a compatible color to transform to color or lightModeColor. Default is "White". if 'x' == to_light, function will convert 'col' to lightModeColor. if 'x' == to_dark, function will convert 'col' to color. if 'col' is not found or 'x' is anything else then a data.frame of compatible colors is returned.

Value

IFC palette of available colors.

---

plotGraph *Plot and Stats Computation for IFC Graph*

Description

Computes plot and stats from a IFC graph

Usage

```
plotGraph(
  obj,
  graph,
  draw = FALSE,
  stats_print = draw,
  color_mode = c("white", "black")[1],
  add_key = "panel",
  precision = c("light", "full")[1],
  trunc_labels = 38,
  trans = "asinh",
  bin,
  viewport = "ideas",
  ...
)
```

Arguments

obj	an 'IFC_data' object extracted with features extracted.
graph	a graph from 'obj' or a list that can be coerced by buildGraph.
draw	whether to draw plot or not. Default is FALSE.
stats_print	whether to print stats or not. Default is given by 'draw' argument.
color_mode	whether to extract colors from 'obj' in white or black mode. Default is "white".
add_key	whether to draw a "global" key under title or in the first "panel" or "both". Default is "panel". Accepted values are either: FALSE, "panel", "global", "both" or c("panel", "global"). Note that it only applies when display is seen as overlaying populations.
precision	when graphs is a 2D scatter with population overlay, this argument controls amount of information displayed. Default is "light". -"light", the default, will only display points of same coordinates that are among the other layers. -"full" will display all the layers.
trunc_labels	maximum number of characters to display for labels. Default is 38.
trans	the name of a transformation function for density graphs. If missing the default, the BasePop[[1]]$densitytrans, if any, will be retrieved, otherwise "asinh" will be used.
bin	number of bin used for histogram / density. Default is missing.
viewport	either "ideas", "data" or "max" defining limits used for the graph. Default is "ideas". -"ideas" will use same limits as the one defined in ideas. -"data" will use data to define limits. -"max" will use data and regions drawn to define limits.
...	other arguments to be passed.

Value

it invisibly returns a list whose members are:
-plot, "trellis" object that can be displayed using plot, if 'draw' was TRUE,
-stats, a table of statistics computed for the graph, if 'stats_print' was TRUE,
-input, a list with input parameters.

`popsCompute` *IFC_pops Computation*

Description

Function used to compute 'IFC_pops' object
It requires pops, regions and features.

Usage

```
popsCompute(
  pops,
  regions,
  features,
  pnt_in_poly_algorithm = 1,
  pnt_in_poly_epsilon = 1e-12,
  display_progress = TRUE,
  title_progress = "",
  ...
)
```

Arguments

`pops`	list of populations that will be coerced by `buildPopulation`.
`regions`	an object of class 'IFC_regions', list of regions.
`features`	an object of class 'IFC_features', data.frame of features.
`pnt_in_poly_algorithm`	algorithm used to determine if object belongs to a polygon region or not. Default is 1. Note that for the moment only 1(Trigonometry) is available.
`pnt_in_poly_epsilon`	epsilon to determine if object belongs to a polygon region or not. It only applies when algorithm is 1. Default is 1e-12.
`display_progress`	whether to display a progress bar. Default is TRUE.
`title_progress`	character string, giving the title of the progress bar. Default is "".
`...`	other arguments to be passed.

Value

an object of class 'IFC_pops'.

Source

For pnt_in_poly_algorithm, Trigonometry, is an adaptation of Jeremy VanDerWal's code https://github.com/jjvanderwal/SDMTools

---

popsCopy *Copy Populations from One File to Another File*

Description

Copies populations from a DAF file into a copy of another DAF file. Only creates new file with copied population.

Usage

```
popsCopy(
  from,
  into,
  write_to,
  pops,
  use_regex = FALSE,
  overwrite = FALSE,
  append_name = TRUE,
  offset = 0,
  endianness = .Platform$endian,
  verbose = FALSE,
  ...
)
```

Arguments

from	path to file to copy populations from.
into	path to file that will be used as a template to copy population into. Caution, it is mandatory that 'into' contains 'from' starting at 'offset'.
write_to	pattern used to export file. Placeholders, like "%d/%s_fromR.%e", will be substituted: -%d: with full path directory of 'into' -%p: with first parent directory of 'into' -%e: with extension of 'into' (without leading .) -%s: with shortname from 'into' (i.e. basename without extension). Exported file extension will be deduced from this pattern. Note that it has to be a .daf.

pops	regular expression or vector of desired populations present in 'from'. If missing, the default, all populations found will be copied. If given but not found, a warning will be sent.
use_regex	whether to use regex to pick up population into 'from'. Default is FALSE.
overwrite	whether to overwrite existing file or not. Default is FALSE. Note that if TRUE, it will overwrite exported file if path of 'into' and deduced from 'write_to' arguments are different. Otherwise, you will get an error saying that overwriting source file is not allowed. Note also that an original file, i.e. generated by IDEAS(R) or INSPIRE(R), will never be overwritten. Otherwise, you will get an error saying that overwriting original file is not allowed.
append_name	whether to append_name basename(from) to exported populations. Default is TRUE.
offset	Object number of 1st object of 'from' in 'into'. Default is 0.
endianness	The endian-ness ("big" or "little") of the target system for the file. Default is .Platform$endian. Endianness describes the bytes order of data stored within the files. This parameter may not be modified.
verbose	whether to display information (use for debugging purpose). Default is FALSE.
...	Other arguments to be passed.

Details

Populations are exported as tagged populations.

Value

a new file is created containing exported populations.
It invisibly returns full path of exported file.

`popsGetObjectsIds` *IFC_pops Object Numbers*

Description

Retrieves objects ids belonging to a population.

Usage

```
popsGetObjectsIds(obj, pop = "")
```

Arguments

obj	an 'IFC_data' object extracted with features extracted.
pop	a population name from 'obj'. Default is "". If left as is or not found an error is thrown displaying all available population in 'obj'.

Value

An integer vector is returned

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  obj <- popsGetObjectsIds(obj = daf, pop = names(daf$pops)[length(daf$pops)])
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

---

popsNetwork *IFC_pops Network Display*

Description

Builds and displays populations network.

Usage

```
popsNetwork(
  obj,
  hierarchical = TRUE,
  color_mode = "white",
  highlight = NULL,
  seed = NULL,
  direction = "LR",
  weighted = TRUE,
  ...
)
```

Arguments

obj	an ‘IFC_data‘ object extracted with features extracted.
hierarchical	whether to display network using a hierarchical layout or not. Default is TRUE.
color_mode	Whether to extract colors from ’obj’ in "white" or "black" mode. Default is "white".
highlight	population to permanently highlight. If found in ’obj’, this population will be displayed with its color. Default is NULL.
seed	If you provide a seed manually, the layout will be the same every time. Default is NULL.

direction	The direction of the hierarchical layout. Default is 'LR'. The available options are: 'UD', 'DU', 'LR', 'RL'. To simplify: up-down, down-up, left-right, right-left.
weighted	whether to scale population's node size according to count. Default is TRUE.
...	other argument to be passed.

Value

a **visNetwork** object.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  daf <- ExtractFromDAF(fileName = file_daf)
  popsNetwork(obj = daf)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

---

popsRename *Populations Renaming*

Description

Renames populations in an 'IFC_data' object

Usage

```
popsRename(
  obj,
  old_names = character(),
  new_names = character(),
  loops = 10L,
  verbose = TRUE,
  ...
)
```

Arguments

obj	an 'IFC_data'.
old_names	character vector of name(s) of population(s) to rename inside 'obj'. Default is character().
new_names	character vector of desired new population(s) name(s). Default is character().

loops	a positive integer specifying the maximum number of recursive loops before raising an error. Default is 10L.
verbose	whether to show a final message about the renaming. Default is TRUE.
...	other arguments to be passed.

Value

an object of class 'IFC_data'.

---

readFCS *FCS File Parser*

Description

Parse data from Flow Cytometry Standard (FCS) compliant files.

Usage

```
readFCS(
  fileName,
 options = list(header = list(start = list(at = 0, n = 6), text_beg = list(at = 10, n
  = 8), text_end = list(at = 18, n = 8), data_beg = list(at = 26, n = 8), data_end =
  list(at = 34, n = 8)), apply_scale = TRUE, first_only = TRUE, force_header = FALSE,
    text_only = FALSE),
 display_progress = TRUE,
  ...
)
```

Arguments

fileName	path to file.
options	list of options used to parse FCS file. It should contain: - header, a list whose members define the "at" offset from header$start$at and the "n" number of bytes to extract: – start: where start reading FCS dataset. Default is list(at = 0, n = 6), – text_beg: where to retrieve file text segment beginning. Default is list(at = 10, n = 8), – text_end: where to retrieve file text segment end. Default is list(at = 18, n = 8), – data_beg: where to retrieve file text segment beginning. Default is list(at = 26, n = 8), – data_end: where to retrieve file text segment end. Default is list(at = 34, n = 8), - apply_scale, whether to apply data scaling. It only applies when fcs file is stored as DATATYPE "I". Default is TRUE. - first_only, whether to extract only the first dataset when several. Default is TRUE.

- force_header, whether to force the use of header to determine the position of data segment. Default is FALSE, for using positions found in "$BEGINDATA" and "$ENDDATA" keywords.
- text_only, whether to only extract text segment. Default is FALSE.

`display_progress`
: whether to display a progress bar. Default is TRUE.

`...`
: other arguments to be passed.

Details

'options' may be tweaked according to file type, instrument and software used to generate it.
Default 'options' should allow to read most files.
'apply_scale', 'force_header', 'first_only', and 'textt_only' can also be passed to 'options' thanks to ...

Value

a list whose elements are lists for each dataset stored within the file.
each sub-list contains:
- header, list of header information corresponding to 'options'
- delimiter, unique character used to separate keyword - values
- text, list of keywords values,
- data, data.frame of values.

Source

Data File Standard for Flow Cytometry, version FCS 3.1 from Spidlen J. et al. available at doi: 10.1002/cyto.a.20825.

`readIFC` *IFC Files Generic Reader*

Description

Reads IFC data from IFC files no matter if they are FCS, DAF, RIF or CIF.

Usage

```
readIFC(fileName, ...)
```

Arguments

`fileName`
: path to file.

`...`
: arguments to pass to `ExtractFromDAF` or `ExtractFromXIF` or `ExtractFromFCS`.

Details

If input 'fileName' is a DAF file ExtractFromDAF will be used to read the file.
If it is a CIF or RIF file readIFC will use ExtractFromXIF.
Finally, if 'fileName' is not a DAF, nor a CIF, nor a RIF file readIFC will use ExtractFromFCS.

Value

an object of class 'IFC_data'.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a rif file, but you can also read daf or cif
  file_rif <- system.file("extdata", "example.rif", package = "IFCdata")
  rif <- readIFC(fileName = file_rif)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

---

smoothLinLog *Smooth LinLog Transformation*

Description

Transforms values in lin-log

Usage

```
smoothLinLog(x, hyper = 1000, base = 10, lin_comp = log(base))
```

Arguments

x	A numeric vector.
hyper	value where transition between Lin/Log is applied.
base	base of Log scale.
lin_comp	value that is used to smooth transition between Lin/Log. Default is log(base).

Value

the smoothLinLog transformation of the input.

subsetOffsets *IFC_offset Subsetting*

Description

Subsets 'IFC_offset'

Usage

```
subsetOffsets(offsets, objects, image_type = c("img", "msk"))
```

Arguments

offsets	object of class 'IFC_offset' to subset.
objects	integer vector, IDEAS objects ids numbers to extract.
image_type	image_type of desired offsets. Default is c("img", "msk"). Allowed are "img" and/or "msk".

Value

a class 'IFC_offset' integer vector or empty integer() if objects are outside of offsets.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  ## use a cif file
  file_cif <- system.file("extdata", "example.cif", package = "IFCdata")
  ## extract offsets
  offsets <- getOffsets(fileName = file_cif)
  ## subset offsets of the 4 first "img" objects
  sub_offs <- subsetOffsets(offsets = offsets, objects = 0:3, image_type = "img")
  ## show subsetted offsets' structure
  str(sub_offs)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

writeIFC *IFC Files Generic Writer*

Description

Writes IFC data to DAF and subsets or merges RIF/CIF Files.

Usage

```
writeIFC(fileName, ...)
```

Arguments

fileName	path to file.
...	arguments to pass to ExportToDAF, ExportToXIF, ExportToFCS, or data_to_DAF.

Details

If an 'IFC_data' object is provided as 'fileName' or in '...' ExportToFCS or data_to_DAF will be used to export object. Otherwise, if 'fileName' is a DAF file ExportToDAF will be used to write file whereas if it is RIF or CIF file(s) writeIFC will use ExportToXIF.

Value

it invisible returns the path of exported file.

Examples

```
if(requireNamespace("IFCdata", quietly = TRUE)) {
  tmp <- tempdir(check = TRUE)
  ## use a daf file
  file_daf <- system.file("extdata", "example.daf", package = "IFCdata")
  ## create a tagged population named test with 1st object
  pop <- buildPopulation(name = "test", type = "T", obj = 0)
  writeIFC(file_daf, write_to = paste0(tmp, "\\test_write.daf"),
           overwrite = TRUE, pops = list(pop))
  ## use a rif file, but you can also use a cif
  file_rif <- system.file("extdata", "example.rif", package = "IFCdata")
  writeIFC(fileName = file_rif, write_to = paste0(tmp, "\\test_write.rif"),
           overwrite = TRUE, objects = 0)
} else {
 message(sprintf('Please run `install.packages("IFCdata", repos = "%s", type = "source")` %s',
                 'https://gitdemont.github.io/IFCdata/',
                 'to install extra files required to run this example.'))
}
```

Index