Package: VoltRon (via r-universe)

May 22, 2026

Type Package

Title VoltRon for Spatial Omics Data Integration and Analysis

Version 0.2.6

Depends R (>= 4.6.0)

Author@R person(``Artür", ``Manukyan", role=c(``aut", ``cre"), email=``artur-man@hotmail.com", comment=c(ORCID=``0000-0002-0441-9517")), person(``Ella", ``Bahry", role=c(``aut")), person(``Raj Prateek", ``Rai", role=c(``aut")), person(``Wei-che", ``Ko", role=c(``aut")), person(``Markus", ``Landthaler", role=c(``aut")), person(``Altuna", ``Akalin", role=c(``aut"))

Author Artür Manukyan, Ella Bahry, Raj Prateek Rai, Wei-Che Ko, Markus Landthaler, Altuna Akalin

Maintainer Artür Manukyan <artur-man@hotmail.com>

Description VoltRon is a spatial omic analysis toolbox for multi-omics integration using spatial image registration. VoltRon is capable of analyzing multiple types and modalities of spatially-aware datasets. VoltRon visualizes and analyzes regions of interests (ROIs), spots, cells, molecules and event tiles.

License MIT + file LICENSE

SystemRequirements OpenCV 4.8 (or higher): libopencv-dev (Debian, Ubuntu) or opencv-devel (Fedora)

Encoding UTF-8

RoxygenNote 7.3.3

Imports utils, methods, grDevices, data.table, Matrix, S4Arrays, S4Vectors, ids, RcppAnnoy, RANN, igraph, dplyr, ggplot2, ggrepel, ggpubr, rjson, magick, EBImage, sp, rlang, shiny, shinyjs, stringr, BiocSingular, uwot, RCDT

LinkingTo Rcpp, RcppArmadillo (>= 0.4)

Collate 'RcppExports.R' 'zzz.R' 'allgenerics.R' 'allclasses.R' 'annotation.R' 'assay.R' 'auxiliary.R' 'clustering.R' 'conversion.R' 'data.R' 'deconvolution.R' 'differentialexpression.R' 'image.R' 'import.R' 'integration.R' 'interactive.R' 'metadata.R' 'objects.R' 'ondisk.R' 'processing.R' 'registration.R' 'sample.R' 'spatial.R' 'visualization.R'

Suggests testthat (>= 3.0.0), DelayedArray, DelayedMatrixStats, BiocParallel, rhdf5, Rarr, ZarrArray, basilisk, reticulate, RBioFormats, VoltRonStore, BPCells, HDF5Array, HDF5DataFrame, ZarrDataFrame, ImageArray, viridisLite, SpatialExperiment, SingleCellExperiment, SummarizedExperiment, Seurat, SeuratObject, Giotto, DESeq2, MuSiC, spacexr, XML, sf, ComplexHeatmap, xlsx, reshape2, arrow, vitessceR, geojsonR, circlize, rstudioapi, ggforce, ggnewscale, anndataR, anndata, SimpleITK

Remotes stla/RCDT, BIMSBbioinfo/VoltRonStore, bnprks/BPCells/r@v0.3.0, BIMSBbioinfo/VoltRon, BIMSBbioinfo/HDF5DataFrame, BIMSBbioinfo/ZarrDataFrame

Config/testthat/edition 3

LazyData true

LazyDataCompression gzip

Config/pak/sysreqs cmake libfftw3-dev libfreetype6-dev libglpk-dev libglu1-mesa-dev make libmagick++-dev gsfonts texlive libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libxml2-dev libopencv-dev libgl1-mesa-dev libssl-dev zlib1g-dev

Repository https://bimsbbioinfo.r-universe.dev

Date/Publication 2026-04-28 21:11:39 UTC

RemoteUrl https://github.com/BIMSBbioinfo/VoltRon

RemoteRef HEAD

RemoteSha 0d063637ce999bed298b6668e4a14a3b9918eb14

Contents

`VoltRon-package` *VoltRon: VoltRon for Spatial Omics Data Integration and Analysis*

Description

VoltRon is a spatial omic analysis toolbox for multi-omics integration using spatial image registration. VoltRon is capable of analyzing multiple types and modalities of spatially-aware datasets. VoltRon visualizes and analyzes regions of interests (ROIs), spots, cells, molecules and event tiles.

addAssay *Add Assay*

Description

add a new assay (vrAssay object) to a VoltRon object

Usage

```
addAssay(object, ...)

## S4 method for signature 'vrMetadata'
addAssay(
  object,
  metadata = NULL,
  assay,
  assay_name,
  sample = "Sample1",
  layer = "Section1"
)

## S4 method for signature 'VoltRon'
addAssay(
  object,
  assay,
  metadata = NULL,
  assay_name,
  sample = "Sample1",
  layer = "Section1"
)
```

Arguments

object	a VoltRon object.
...	arguments passed to other methods.
metadata	a predefined metadata
assay	a vrAssay object
assay_name	assay name of the new added assay
sample	sample name
layer	layer name

addBlockConnectivity *addBlockConnectivity*

Description

add connectivity information to the layers (vrLayer) of the same block (Block)

Usage

```
addBlockConnectivity(object, connectivity, zlocation = NULL, sample)
```

Arguments

object	a VoltRon object
connectivity	a metadata of edges representing connected layers within a block
zlocation	zlocation
sample	sample name

addFeature *addFeature*

Description

add a new feature set to a vrAssay (or VoltRon) object

Usage

```
addFeature(object, ...)

## S4 method for signature 'vrAssayV2'
addFeature(object, data, feature_name)

## S4 method for signature 'VoltRon'
addFeature(object, assay = NULL, data, feature_name)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
data	new data matrix for new feature set
feature_name	the name of the new feature set
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay. If given as "all", then provides a summary of spatial systems across all assays.

addMetadata *addMetadata*

Description

adding new columns or updating the values of the existing columns

Usage

```
addMetadata(object, assay = NULL, type = NULL, value, label)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
type	the assay type: ROI, spot or cell, or all for the entire metadata object
value	the new values of the metadata column
label	the label of the new column, either a new column or an existing one

addSpatialLayer *addSpatialLayer*

Description

adding additional layers of spatial plots to an existing vrSpatialPlot.

Usage

```
addSpatialLayer(
  g,
  object,
  assay,
  group.by = "Sample",
  plot.segments = FALSE,
  group.ids = NULL,
  reg = FALSE,
  colors = NULL,
  alpha = 1,
  n.tile = NULL,
  pt.size = 2,
  cell.shape = 21,
  graph = NULL,
  graph.edge.color = "orange",
  spatial = NULL,
  combine.groups = FALSE
)
```

Arguments

g	ggplot object
object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
group.by	a column of metadata from Metadata used as grouping label for the spatial entities
plot.segments	plot segments from vrSegments instead of points
group.ids	a subset of categories defined in metadata column from group.by
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
colors	the color set for group.by. Should be of the same size of group.id (if specified) or unique elements in group.by
alpha	alpha level of colors of visualized points and segments
n.tile	The number of tiles on x-and y-axis for rasterizing points (see geom_tile). The rasterization is performed automatically for large number of points Only applicable to spots, cells and molecules. If n.tile = 0 will turn of automated rasterization.
pt.size	point size
cell.shape	the shape of the points representing cells, see geom_point
graph	if not NULL, the graph is added to the plot
graph.edge.color	the color of graph edges, if graph is not NULL.
spatial	the name of the main spatial system
combine.groups	if TRUE, tile colors will reflect relative abundance of either of two groups, strictly for visualizing two groups when assay is a molecule typed and tiled (see n.tile).

---

annotateSpatialData *annotateSpatialData*

Description

A mini shiny app to for annotating spatial points

Usage

```
annotateSpatialData(
  object,
  label = "annotation",
  assay = NULL,
  annotation_assay = "ROIAnnotation",
  use.image.only = FALSE,
```

```
  shiny.options = list(launch.browser = getOption("shiny.launch.browser", interactive())),
  image_name = NULL,
  channel = NULL,
  ...
)
```

Arguments

object	a VoltRon object
label	the name of the new metadata column (default: annotation) annotating spatial points by selected polygons
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
annotation_assay	name of the annotation assay ()
use.image.only	if TRUE, use only the image
shiny.options	a list of shiny options (launch.browser, host, port etc.) passed options arguement of shinyApp. For more information, see runApp
image_name	the name/key of the image
channel	the name of the main channel
...	additional parameters passed to vrSpatialPlot.

Examples

```
## Not run:
# Annotate based on images
visium_data <- annotateSpatialData(visium_data, use.image.only = TRUE)

# Annotate based on spatial plot
xenium_data <- annotateSpatialData(xenium_data, group.by = "clusters")

## End(Not run)
```

---

as.AnnData *as.AnnData*

Description

Converting a VoltRon object into a AnnData (.h5ad) object

Usage

```
as.AnnData(
  object,
  file,
  assay = NULL,
  flip_coordinates = FALSE,
  method = "anndata",
  create.ometiff = FALSE,
  python.path = NULL,
  ...
)
```

Arguments

object	a VoltRon object
file	the name of the h5ad file.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
flip_coordinates	if TRUE, the spatial coordinates (including segments) will be flipped.
method	the package to use for conversion: "anndataR" or "anndata".
create.ometiff	should an ometiff file be generated of default image of the object
python.path	the path to the python binary, otherwise either `basilisk` package is used or `getOption("voltron.python.path")` should be not NULL.
...	additional parameters passed to vrImages.

Details

This function converts a VoltRon object into an AnnData object (.h5ad file). It extracts assay data, spatial coordinates, and optionally flips coordinates. Images associated with the assay can be included in the resulting AnnData file, with additional customization parameters like channel, scale.perc.

`as.Giotto` *as.Giotto*

Description

Converting a VoltRon object into a Giotto object

Usage

```
as.Giotto(object, assay = NULL, reg = FALSE)
```

Arguments

object	a VoltRon object
assay	the name of the assay to be converted
reg	if TRUE, registered coordinates will be used

---

as.OmeTiff *as.OmeTiff*

Description

Converting VoltRon (magick) images to ome.tiff

Usage

```
as.OmeTiff(object, out_path, image_id = "image_1", python.path = NULL)
```

Arguments

object	a magick-image object
out_path	output path to ome.tiff file
image_id	image name
python.path	the path to the python binary, otherwise either basilisk package is used or getOption("voltron.python.path") should be not NULL.

---

as.OmeZarr *as.OmeZarr*

Description

Converting VoltRon (magick) images to ome.tiff

Usage

```
as.OmeZarr(object, out_path, image_id = "image_1")
```

Arguments

object	a magick-image object
out_path	output path to ome.tiff file
image_id	image name

as.Seurat *as.Seurat*

Description

Converting a VoltRon object into a Seurat object

Usage

```
as.Seurat(
  object,
  cell.assay = NULL,
  molecule.assay = NULL,
  image_key = "fov",
  type = c("image", "spatial"),
  reg = FALSE
)
```

Arguments

object	a VoltRon object
cell.assay	the name(type) of the cell assay to be converted
molecule.assay	the name(type) of the molecule assay to be added to the cell assay in Seurat object
image_key	the name (or prefix) of the image(s)
type	the spatial data type of Seurat object: "image" or "spatial"
reg	if TRUE, registered coordinates will be used

as.SpatialExperiment *as.SpatialExperiment*

Description

Converting a VoltRon object into a SpatialExperiment object

Usage

```
as.SpatialExperiment(object, assay = NULL, reg = FALSE)
```

Arguments

object	a VoltRon object
assay	the name of the assay to be converted
reg	if TRUE, registered coordinates will be used

as.VoltRon *as.VoltRon*

Description

Generic methods for conversion into a VoltRon object

Usage

```
as.VoltRon(object, ...)

## S3 method for class 'Seurat'
as.VoltRon(
  object,
  type = c("image", "spatial"),
  assay_name = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'SpatialExperiment'
as.VoltRon(
  object,
  assay_type = "cell",
  assay_name = NULL,
  image_id = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	a Seurat or SpatialExperiment object
...	Additional parameter passed to formVoltRon
type	the spatial data type of Seurat object: "image" or "spatial"
assay_name	the assay name of the voltron assays (e.g. Visium, Xenium etc.)
verbose	verbose
assay_type	one of two types, 'cell' or 'spot' etc.
image_id	select image_id names if needed.

as.Zarr *as.Zarr*

Description

Generic methods to save VoltRon or magick-image objects as zarr files

Usage

```
as.Zarr(object, out_path, image_id)

## S3 method for class '`magick-image`'
as.Zarr(object, out_path, image_id = "image_1")
```

Arguments

object	a VoltRon or magick-image object
out_path	output path to zarr file
image_id	image name

combineChannels *combineChannels*

Description

Combining channels into novel channels of the same image

Usage

```
combineChannels(object, ...)

## S4 method for signature 'VoltRon'
combineChannels(
  object,
  assay = NULL,
  name = NULL,
  reg = FALSE,
  channels = NULL,
  colors = NULL,
  channel_key = "combined"
)

## S4 method for signature 'vrAssay'
combineChannels(
  object,
```

```
  name = NULL,
  reg = FALSE,
  channels = NULL,
  colors = NULL,
  channel_key = "combined"
)

## S4 method for signature 'vrAssayV2'
combineChannels(
  object,
  name = NULL,
  reg = FALSE,
  channels = NULL,
  colors = NULL,
  channel_key = "combined"
)

## S4 method for signature 'vrImage'
combineChannels(
  object,
  channels = NULL,
  colors = NULL,
  channel_key = "combined"
)

## S4 method for signature 'vrSpatial'
combineChannels(
  object,
  channels = NULL,
  colors = NULL,
  channel_key = "combined"
)
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
name	the name of the image
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
channels	the name of the channel associated with the image
colors	the colors associated with each channel
channel_key	the name of the new channel name

combineGraphs *combineGraphs*

Description

Combining the edges of multiple graphs

Usage

```
combineGraphs(
  object,
  graph.names = NULL,
  graph.weights = NULL,
  graph.key = "combined"
)
```

Arguments

object	a VoltRon Object
graph.names	a vector of graph names
graph.weights	the weights for edges of each graph.
graph.key	the name of the combined graph

convertAnnDataToVoltRon *convertAnnDataToVoltRon*

Description

converting AnnData h5ad files to VoltRon objects

Usage

```
convertAnnDataToVoltRon(file, AssayID = NULL, ...)
```

Arguments

file	h5ad file
AssayID	the ID assays in the h5ad file
...	additional parameters passed to formVoltRon

demuxVoltRon *demuxVoltRon*

Description

Subsetting/demultiplexing of the VoltRon Object using interactive shiny app

Usage

```
demuxVoltRon(
  object,
  max.pixel.size = 1200,
  use.points.only = FALSE,
 shiny.options = list(launch.browser = getOption("shiny.launch.browser", interactive()))
)
```

Arguments

object	a VoltRon object
max.pixel.size	the initial width of the object image
use.points.only	use spatial points instead of the reference image
shiny.options	a list of shiny options (launch.browser, host, port etc.) passed options argument of shinyApp. For more information, see runApp

dummy_cols *Fast creation of dummy variables*

Description

Quickly create dummy (binary) columns from character and factor type columns in the inputted data (and numeric columns if specified.) This function is useful for statistical analysis when you want binary columns rather than character columns. Adapted from the fastDummies package (https://jacobkap.github.io/fastDummies/)

Usage

```
dummy_cols(
  .data,
  select_columns = NULL,
  remove_first_dummy = FALSE,
  remove_most_frequent_dummy = FALSE,
  ignore_na = FALSE,
  split = NULL,
  remove_selected_columns = FALSE,
  omit_colname_prefix = FALSE
)
```

Arguments

`.data`	An object with the data set you want to make dummy columns from.
`select_columns`	Vector of column names that you want to create dummy variables from. If NULL (default), uses all character and factor columns.
`remove_first_dummy`	Removes the first dummy of every variable such that only n-1 dummies remain. This avoids multicollinearity issues in models.
`remove_most_frequent_dummy`	Removes the most frequently observed category such that only n-1 dummies remain. If there is a tie for most frequent, will remove the first (by alphabetical order) category that is tied for most frequent.
`ignore_na`	If TRUE, ignores any NA values in the column. If FALSE (default), then it will make a dummy column for value_NA and give a 1 in any row which has a NA value.
`split`	A string to split a column when multiple categories are in the cell. For example, if a variable is Pets and the rows are "cat", "dog", and "turtle", each of these pets would become its own dummy column. If one row is "cat, dog", then a split value of "," this row would have a value of 1 for both the cat and dog dummy columns.
`remove_selected_columns`	If TRUE (not default), removes the columns used to generate the dummy columns.
`omit_colname_prefix`	If TRUE (not default) and ‘length(select_columns) == 1‘, omit pre-pending the name of ‘select_columns‘ to the names of the newly generated dummy columns

Value

A data.frame (or tibble or data.table, depending on input data type) with same number of rows as inputted data and original columns plus the newly created dummy columns.

`fixVoltRon` *fixVoltRon*

Description

fixVoltRon

Usage

```
fixVoltRon(object)
```

Arguments

`object`	a VoltRon object

flipCoordinates *flipCoordinates*

Description

Flip the coordinates of spatial points in the y axis direction.

Usage

```
flipCoordinates(object, ...)

## S4 method for signature 'VoltRon'
flipCoordinates(
  object,
  assay = NULL,
  image_name = NULL,
  spatial_name = NULL,
  ...
)

## S4 method for signature 'vrAssay'
flipCoordinates(object, image_name = NULL, spatial_name = NULL, ...)

## S4 method for signature 'vrAssayV2'
flipCoordinates(object, image_name = NULL, spatial_name = NULL, ...)
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	additional parameters passed to vrCoordinates and vrSegments
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
image_name	(deprecated, use spatial_name) the name/key of the image
spatial_name	the name/key of the spatial system associated with the coordinates

formAssay *formAssay*

Description

Create a vrAssay (VoltRon assay) object

Usage

```
formAssay(
  data = NULL,
  coords,
  segments = list(),
  image = NULL,
  params = list(),
  type = "ROI",
  name = "Assay1",
  main_image = "image_1",
  main_featureset = NULL,
  assay_version = "v2",
  ...
)
```

Arguments

data	the feature matrix of spatialpoints
coords	the coordinates of the spatial points
segments	the list of segments each associated with a spatial point (optional)
image	a singelton or list of images as magick-image objects
params	additional parameters of the object
type	the type of the assay (tile, molecule, cell, spot or ROI)
name	the name of the assay
main_image	the name of the main_image
main_featureset	the name of the main_featureset
assay_version	the assay version
...	additional arguements passed to formImage

---

formImage *formImage*

Description

Create a vrImage (VoltRon image) object

Usage

```
formImage(coords, segments = list(), image = NULL, main_channel = NULL)
```

Arguments

coords	the coordinates of the spatial points
segments	the list of segments each associated with a spatial point
image	a singelton or list of images as magick-image objects
main_channel	the key of the main channel of vrImage object

---

formVoltRon *formVoltRon*

Description

Create a VoltRon object

Usage

```
formVoltRon(
  data = NULL,
  metadata = NULL,
  image = NULL,
  coords,
  segments = list(),
  sample.metadata = NULL,
  main.assay = NULL,
  assay.type = "cell",
  params = list(),
  sample_name = NULL,
  layer_name = NULL,
  image_name = NULL,
  feature_name = NULL,
  project = NULL,
  version = "v2",
  ...
)
```

Arguments

data	the feature matrix of spatialpoints
metadata	a metadata object of class vrMetadata
image	a singelton or list of images as magick-image objects
coords	the coordinates of the spatial points
segments	the list of segments each associated with a spatial point
sample.metadata	a data frame of the sample metadata, see SampleMetadata
main.assay	the name of the main assay

assay.type	the type of the assay (tile, molecule, cell, spot or ROI)
params	additional parameters
sample_name	the name of the sample
layer_name	the name of the layer
image_name	the name/key of the image
feature_name	the name/key of the feature set
project	project name
version	the assay version, V1 or V2
...	additional parameters passed to formAssay

FromSegmentToCrop *FromSegmentToCrop*

Description

get magick crop information from coordinates of a segment

Usage

```
FromSegmentToCrop(segment, imageinfo)
```

Arguments

segment	coordinates of a segment
imageinfo	info of the image

---

generateCosMxImage *generateCosMxImage*

Description

Generates a low resolution Morphology image of the CosMx experiment

Usage

```
generateCosMxImage(
  dir.path,
  fov.position.file,
  increase.contrast = FALSE,
  output.path = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`dir.path`	CosMx folder of images
`fov.position.file`	the file providing FOV positions
`increase.contrast`	increase the contrast of the image before writing
`output.path`	The path to the new morphology image created if the image should be saved to a location other than Xenium output folder.
`verbose`	verbose
`...`	additional parameters passed to the writeImage function

`generateGeoJSON` *generateGeoJSON*

Description

Generating geojson files from segments

Usage

```
generateGeoJSON(segments, file)
```

Arguments

`segments`	the segments, typically from vrSegments.
`file`	the GeoJSON file, typically to be used by QuPath software.

`generateSegments` *generateSegments*

Description

The function to import segments from a geojson file

Usage

```
generateSegments(geojson.file)
```

Arguments

`geojson.file`	the GeoJSON file, typically generated by QuPath software

generateTileData *generateTileData*

Description

Generating data matrices for tile-based VoltRon objects from images

Usage

```
generateTileData(object, ...)

## S4 method for signature 'VoltRon'
generateTileData(object, assay = NULL, ...)

## S4 method for signature 'vrAssay'
generateTileData(object, name = NULL, reg = FALSE, channel = NULL)

## S4 method for signature 'vrAssayV2'
generateTileData(object, name = NULL, reg = FALSE, channel = NULL)
```

Arguments

object	a VoltRon or vrAssay object.
...	additional parameters passed to vrAssay.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
name	the name of the main spatial system
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
channel	the name of the channel associated with the image

generateXeniumImage *generateXeniumImage*

Description

Generate a low resolution DAPI image of the Xenium experiment

Usage

```
generateXeniumImage(
  dir.path,
  increase.contrast = TRUE,
  resolution_level = 7,
  overwrite_resolution = FALSE,
  output.path = NULL,
  file.name = "morphology_lowres.tif",
  verbose = TRUE,
  ...
)
```

Arguments

dir.path	Xenium output folder
increase.contrast	increase the contrast of the image before writing
resolution_level	the level of resolution within Xenium OME-TIFF image. Default: 7 (553x402)
overwrite_resolution	if TRUE, the image "file.name" will be generated again although it exists at "dir.path"
output.path	The path to the new morphology image created if the image should be saved to a location other than Xenium output folder.
file.name	the name of the lowred morphology image. Default: morphology_lowres.tif
verbose	verbose
...	additional parameters passed to the writeImage function

Details

The Xenium morphology_mip.ome.tif file that is found under the outs folder comes is an hyperstack of different resolutions of the DAPI image. generateXeniumImage allows extracting only one of these layers by specifying the `resolution` parameter (Default: 7 for 553x402) among 1 to 8. Lower incides of resolutions have higher higher resolutions, e.g. 1 for 35416x25778. Note that you may need to allocate larger memory of Java to import higher resolution images.

`getBasilisk` *get the Python Basilisk environment*

Description

Defines a conda environment via Basilisk, which is used to convert R objects to Zarr stores.

Usage

```
getBasilisk()
```

`getClusters` *getClusters*

Description

Get clustering of the VoltRon object

Usage

```
getClusters(
  object,
  assay = NULL,
  label = "clusters",
  method = "leiden",
  resolution = 1,
  graph = "kNN",
  data.type = "norm",
  dims = 1:30,
  nclus = integer(0),
  distance_measure = "euclidean",
  abundance_limit = 2,
  seed = 1
)
```

Arguments

`object`	a VoltRon object.
`assay`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
`label`	the name for the newly created clustering column in the metadata.
`method`	The method of clustering. Use : (i) 'leiden' to perform graph clustering and (ii) 'kmeans' for K-means based clustering (iii) 'hierarchical' for hierarchical clustering.
`resolution`	the resolution parameter for leiden clustering.
`graph`	the graph type to be used.
`data.type`	the type of data used to cluster spatial points: "norm" (default), "raw" or an existing embeddings vrEmbeddingNames.
`dims`	the number of dimensions extracted from the embedding if data.type is not NULL
`nclus`	The number of cluster centers for K-means or hierarchical clustering.
`distance_measure`	the distance measure used by hierarchical clustering. See `method` for a list of distance measures in dist.

abundance_limit
: the minimum number of points for a cluster, hence clusters with abundance lower than this limit will be appointed to other nearby clusters.

seed
: seed.

---

getDeconvolution *getDeconvolution*

Description

Calculate deconvolution of spots and ROIs

Usage

```
getDeconvolution(
  object,
  assay = NULL,
  features = NULL,
  sc.object,
  sc.assay = "RNA",
  sc.cluster = "seurat_clusters",
  method = "RCTD",
  ...
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
features	features
sc.object	Seurat Object
sc.assay	assay of the Seurat Object used for the single cell data reference
sc.cluster	metadata column variable used for the single cell data reference
method	Deconvolution method, RCTD (spot), SPOTlight (spot), MuSiC (ROI)
...	additional parameters passed to method specific functions, e.g. RCTD, MuSiC.

getDiffExp *getDiffExp*

Description

Get differential expression with DESeq2

Usage

```
getDiffExp(
  object,
  assay = NULL,
  group.by,
  group.base = NULL,
  covariates = NULL,
  method = "DESeq2"
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
group.by	the categorical variable from metadata to get differentially expressed features across
group.base	Optional, the base category in group.by which is used as control group
covariates	the covariate variable for the design formula
method	the method for DE analysis, e.g. DESeq2

getFeatures *getFeatures*

Description

Get variable features of assays

Usage

```
getFeatures(object, ...)

## S4 method for signature 'VoltRon'
getFeatures(object, assay = NULL, max.count = 1, n = 3000)

## S4 method for signature 'vrAssay'
```

```
getFeatures(object, max.count = 1, n = 3000)

## S4 method for signature 'vrAssayV2'
getFeatures(object, max.count = 1, n = 3000)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
max.count	maximum count (across spatial points) for low count filtering
n	the top number of variable features

`getHotSpotAnalysis` *getHotSpotAnalysis*

Description

Conduct hot spot detection

Usage

```
getHotSpotAnalysis(
  object,
  assay = NULL,
  method = "Getis-Ord",
  features,
  graph.type = NULL,
  group.ids = NULL,
  alpha.value = 0.01,
  norm = TRUE,
  n.tile = 0,
  verbose = TRUE
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
method	the statistical method of conducting hot spot analysis. Default is "Getis-Ord"
features	a set of features to be visualized, either from vrFeatures of raw or normalized data or columns of the Metadata.
graph.type	the type of graph to determine spatial neighborhood

group.ids	a subset of categories defined in metadata column from features, throws an error if the feature does not include
alpha.value	the alpha value for the hot spot analysis test. Default is 0.01
norm	if TRUE, the normalized data is used
n.tile	should points be rasterized along x-and y-axes, typically used for molecule assays to generate features for Getis-Ord statistics
verbose	verbose

---

getNicheAssay *getNicheAssay*

Description

Create Niche Assays

Usage

```
getNicheAssay(
  object,
  assay = NULL,
  label = NULL,
  graph.type = "delaunay",
  new_feature_name = "Niche"
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
label	grouping label for Niche definition
graph.type	the type of graph to determine spatial neighborhood
new_feature_name	the name of the new feature set created for the niche assay. Default: "Niche"

getOmeTiffChannels *getOmeTiffChannels*

Description

Function that returns the channel names of an image in a ome.tiff file

Usage

```
getOmeTiffChannels(ome.tiff)
```

Arguments

ome.tiff	location to ome.tiff file

getPCA *getPCA*

Description

calculate PCA of the VoltRon objects

Usage

```
getPCA(
  object,
  assay = NULL,
  features = NULL,
  feat_type = NULL,
  data.type = "norm",
  dims = 30,
  pca.key = "pca",
  n.workers = 1,
  overwrite = FALSE,
  seed = 1
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
features	the selected features for PCA reduction
feat_type	the feature set type

data.type	the type of data used to calculate PCA from: "norm" (default), "raw" or an existing embeddings vrEmbeddingNames.
dims	the number of dimensions extracted from PCA
pca.key	the key name for the embedding, default: pca
n.workers	the number of cores/workers use for parallelization.
overwrite	Whether the existing embedding with name 'type' should be overwritten in vrEmbeddings
seed	seed

`getProfileNeighbors` *Get profile specific neighbors*

Description

Get neighbors of spatial points

Usage

```
getProfileNeighbors(
  object,
  assay = NULL,
  method = "kNN",
  k = 10,
  data.type = "pca",
  dims = seq_len(30),
  graph.key = method
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
method	the method used for graph construction, SNN or kNN
k	number of neighbors for kNN
data.type	the type of embedding used for neighborhood calculation, e.g. raw counts (raw), normalized counts (norm), PCA embeddings (pca), UMAP embeddings (umap) etc.
dims	the set of dimensions of the embedding data
graph.key	the name of the graph

getRcppAutomatedRegistration *getRcppAutomatedRegistration*

Description

Automated registration workflos with Rcpp

Usage

```
getRcppAutomatedRegistration(
  ref_image,
  query_image,
  GOOD_MATCH_PERCENT = 0.15,
  MAX_FEATURES = 500,
  invert_query = FALSE,
  invert_ref = FALSE,
  flipflop_query = "None",
  flipflop_ref = "None",
  rotate_query = "0",
  rotate_ref = "0",
  matcher = "FLANN",
  method = "Homography",
  nonrigid = "TPS (OpenCV)"
)
```

Arguments

ref_image	reference image
query_image	query image
GOOD_MATCH_PERCENT	the percentage of good matching keypoints, used by "Brute force" method
MAX_FEATURES	maximum number of detected features, i.e. keypoints, used by "Brute force" method
invert_query	invert query image?
invert_ref	invert reference image
flipflop_query	flip or flop the query image
flipflop_ref	flip or flop the reference image
rotate_query	rotation of query image
rotate_ref	rotation of reference image
matcher	the matching method for landmarks/keypoints FLANN or BRUTE-FORCE
method	the automated registration method, Homography or Homography+TPS
nonrigid	the non-rigid registration method, "TPS (OpenCV)" or "BSpline (SimpleITK)"

getRcppManualRegistration *getRcppManualRegistration*

Description

Manual registration workflow with Rcpp

Usage

```
getRcppManualRegistration(
  query_image,
  ref_image,
  query_landmark,
  reference_landmark,
  method = "Homography",
  nonrigid = "TPS (OpenCV)"
)
```

Arguments

query_image	query image
ref_image	reference image
query_landmark	query landmark points
reference_landmark	refernece landmark points
method	the automated registration method, either TPS or Homography+TPS
nonrigid	the non-rigid registration method, "TPS (OpenCV)" or "BSpline (SimpleITK)"

getSpatialNeighbors *Get spatial neighbors*

Description

get neighbors in an assay given spatial coordinates

Usage

```
getSpatialNeighbors(
  object,
  assay = NULL,
  group.by = NULL,
  group.ids = NULL,
  method = "delaunay",
```

```
  k = 10,
  radius = numeric(0),
  graph.key = method,
  calculate.distances = FALSE,
  verbose = TRUE
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
group.by	a column of metadata from Metadata used as grouping label for the spatial entities.
group.ids	a subset of categories defined in metadata column from group.by.
method	the method spatial connectivity: "delaunay", "spatialkNN", "radius".
k	number of neighbors for kNN.
radius	When method = "radius" selected, determines the radius of a neighborhood ball around each spatial point.
graph.key	the name of the graph.
calculate.distances	If TRUE, distances between neighbors are also stored in the graph.
verbose	verbose

---

getUMAP *getUMAP*

Description

calculate UMAP of the VoltRon objects

Usage

```
getUMAP(
  object,
  assay = NULL,
  data.type = "pca",
  dims = seq_len(30),
  umap.key = "umap",
  overwrite = FALSE,
  seed = 1
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
data.type	the type of data used to calculate UMAP from: "pca" (default), "raw" or "norm"
dims	the number of dimensions extracted from PCA
umap.key	the name of the umap embedding, default: umap
overwrite	Whether the existing embedding with name 'type' should be overwritten in vrEmbeddings
seed	seed

---

getVariableFeatures *getVariableFeatures*

Description

get shared variable features across multiple assays

Usage

```
getVariableFeatures(object, assay = NULL, n = 3000, ...)
```

Arguments

object	a VoltRon Object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
n	the number of features
...	additional arguements passed to vrFeatureData

---

import10Xh5 *import10Xh5*

Description

import the sparse matrix from the H5 file

Usage

```
import10Xh5(filename)
```

Arguments

filename	the path to h5 file

`importCosMx` *importCosMx*

Description

Import CosMx data

Usage

```
importCosMx(
  path,
  assay_name = "CosMx",
  image = NULL,
  image_name = "main",
  import_molecules = FALSE,
  verbose = TRUE,
  method = "CSV",
  feature_name = NULL,
  ...
)
```

Arguments

`path`	the path to the tiledb folder
`assay_name`	the assay name, default: CosMx
`image`	the reference morphology image of the CosMx assay
`image_name`	the image name of the CosMx assay, Default: main
`import_molecules`	if TRUE, molecule assay will be created along with cell assay.
`verbose`	verbose
`method`	the approach for importing the CosMx assay either by the folder of CSVs or with TileDB array.
`feature_name`	the name/key of the feature set.
`...`	additional parameters passed to formVoltRon

`importDBITSeq` *importDBITSeq*

Description

Importing DBIT-Seq data

Usage

```
importDBITSeq(
  path.rna,
  path.prot = NULL,
  size = 10,
  assay_name = "DBIT-Seq",
  sample_name = NULL,
  image_name = "main",
  channel_name = "H&E",
  ...
)
```

Arguments

path.rna	path to rna count matrix
path.prot	path to protein count matrix
size	the size of the in situ pixel (defualt is 10 (micron))
assay_name	the assay name
sample_name	the name of the sample
image_name	the image name of the Visium assay, Default: main
channel_name	the channel name of the image of the Visium assay, Default: H&E
...	additional parameters passed to formVoltRon

---

importGenePS *importGenePS*

Description

Importing GenePS data

Usage

```
importGenePS(
  dir.path,
  assay_name = "GenePS",
  sample_name = NULL,
  use_image = TRUE,
  resolution_level = 7,
  image_name = "main",
  channel_name = "DAPI",
  import_molecules = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

dir.path	path to Xenium output folder
assay_name	the assay name of the SR object
sample_name	the name of the sample
use_image	if TRUE, the DAPI image will be used.
resolution_level	the level of resolution within TIFF image. Default: 7 (971x638)
image_name	the image name of the Xenium assay, Default: main
channel_name	the channel name of the image of the Xenium assay, Default: DAPI
import_molecules	if TRUE, molecule assay will be created along with cell assay.
verbose	verbose
...	additional parameters passed to formVoltRon

---

importGeoMx *importGeoMx*

Description

Import GeoMx data

Usage

```
importGeoMx(
  dcc.path,
  pkc.file,
  summarySegment,
  summarySegmentSheetName,
  assay_name = "GeoMx",
  image = NULL,
  segment_polygons = FALSE,
  ome.tiff = NULL,
  resolution_level = 3,
  image_name = "main",
  verbose = TRUE,
  ...
)
```

Arguments

dcc.path	path to the folder where the dcc files are found
pkc.file	path to the pkc file
summarySegment	the metadata csv (sep = ";") or excel file, if the file is an excel file, summarySegmentSheetName should be provided as well.

summarySegmentSheetName
: the sheet name of the excel file, summarySegment

assay_name
: the assay name, default: GeoMx

image
: the reference morphology image of the GeoMx assay

segment_polygons
: if TRUE, the ROI polygons are parsed from the OME.TIFF file

ome.tiff
: the OME.TIFF file of the GeoMx experiment if exists

resolution_level
: the level of resolution within GeoMx OME-TIFF image, Default: 3

image_name
: the image name of the Visium assay, Default: main

verbose
: verbose

...
: additional parameters passed to formVoltRon

---

importImageData *importImageData*

Description

import an image as VoltRon object

Usage

```
importImageData(
  image,
  tile.size = 10,
  segments = NULL,
  image_name = "main",
  channels = NULL,
  series = 1,
  resolution = NULL,
  is.RGB = TRUE,
  ...
)
```

Arguments

image
: a single or a list of image paths or magick-image objects

tile.size
: the size of tiles

segments
: Either a list of segments or a GeoJSON file. This will result in a second assay in the VoltRon object to be created

image_name
: the image name of the Image assay, Default: main

channels
: the channel names of the images if multiple images are provided

series
: the series IDs of the pyramidal image, typical an integer starting from 1

resolution
: the resolution IDs of the pyramidal image, typical an integer starting from 1

is.RGB
: If TRUE, all three channel images will be converted to RGB images.

...
: additional parameters passed to formVoltRon

Examples

```
# single image
imgfile <- system.file("extdata", "DAPI.tif", package = "VoltRon")
vrdata <- importImageData(imgfile, image_name = "main")

# multiple images
imgfile <- c(system.file("extdata", "DAPI.tif", package = "VoltRon"),
             system.file("extdata", "DAPI.tif", package = "VoltRon"))
vrdata <- importImageData(imgfile, image_name = "main",
                                   channels = c("DAPI", "DAPI2"))
```

`importOpenST` *importOpenST*

Description

Importing OpenST data

Usage

```
importOpenST(
  h5ad.path,
  assay_name = "OpenST",
  sample_name = NULL,
  image_name = "main",
  channel_name = "H&E",
  verbose = TRUE,
  ...
)
```

Arguments

`h5ad.path`	path to h5ad file of STOmics output
`assay_name`	the assay name
`sample_name`	the name of the sample
`image_name`	the image name of the Visium assay, Default: main
`channel_name`	the channel name of the image of the Visium assay, Default: H&E
`verbose`	verbose
`...`	additional parameters passed to formVoltRon

`importPhenoCycler` *importPhenoCycler*

Description

Importing PhenoCycler data

Usage

```
importPhenoCycler(
  dir.path,
  assay_name = "PhenoCycler",
  sample_name = NULL,
  image_name = "main",
  type = c("inform", "processor", "qupath"),
  filter = "DAPI|Blank|Empty",
  inform.quant = c("mean", "total", "min", "max", "std"),
  verbose = TRUE,
  ...
)
```

Arguments

`dir.path`	path to PhenoCycler output folder
`assay_name`	the assay name of the SR object
`sample_name`	the name of the sample
`image_name`	the image name of the Xenium assay, Default: main
`type`	Specify which type matrix is being provided. • "`processor`": matrix generated by CODEX Processor • "`inform`": matrix generated by inForm • "`qupath`": matrix generated by QuPath
`filter`	A pattern to filter features by; pass `NA` to skip feature filtering
`inform.quant`	When `type` is "`inform`", the quantification level to read in
`verbose`	verbose
`...`	additional parameters passed to formVoltRon

`importQuPathIF` *importImageData*

Description

import an QuPath-quantified IF assay as VoltRon object

Usage

```
importQuPathIF(
  measurements,
  image,
  segments,
  image_name = "main",
  channels = NULL,
  series = 1,
  resolution = NULL,
  ...
)
```

Arguments

`measurements`	measurements
`image`	a single or a list of image paths or magick-image objects
`segments`	Either a list of segments or a GeoJSON file. This will result in a second assay in the VoltRon object to be created
`image_name`	the image name of the Image assay, Default: main
`channels`	the channel names of the images if multiple images are provided
`series`	the series IDs of the pyramidal image, typically an integer starting from 1
`resolution`	the resolution IDs of the pyramidal image, typically an integer starting from 1
`...`	additional parameters passed to formVoltRon

`importSTOmics` *importSTOmics*

Description

Importing STOmics (Stereo-Seq) data

Usage

```
importSTOmics(
  h5ad.path,
  assay_name = "STOmics",
  sample_name = NULL,
  image_name = "main",
  channel_name = "H&E",
  ...
)
```

Arguments

h5ad.path	path to h5ad file of STOmics output
assay_name	the assay name
sample_name	the name of the sample
image_name	the image name of the Visium assay, Default: main
channel_name	the channel name of the image of the Visium assay, Default: H&E
...	additional parameters passed to formVoltRon

---

importVisium *importVisium*

Description

Importing Visium data

Usage

```
importVisium(
  dir.path,
  selected_assay = "Gene Expression",
  assay_name = "Visium",
  sample_name = NULL,
  image_name = "main",
  channel_name = "H&E",
  inTissue = TRUE,
  resolution_level = "lowres",
  ...
)
```

Arguments

dir.path	path to Visium output folder
selected_assay	selected assay from Visium
assay_name	the assay name
sample_name	the name of the sample
image_name	the image name of the Visium assay, Default: main
channel_name	the channel name of the image of the Visium assay, Default: H&E
inTissue	if TRUE, only barcodes that are in the tissue will be kept (default: TRUE)
resolution_level	the level of resolution of Visium image: "lowres" (default) or "hires"
...	additional parameters passed to formVoltRon

---

importVisiumHD *importVisiumHD*

Description

Importing VisiumHD data

Usage

```
importVisiumHD(
  dir.path,
  bin.size = "8",
  selected_assay = "Gene Expression",
  assay_name = "VisiumHD",
  sample_name = NULL,
  image_name = "main",
  channel_name = "H&E",
  inTissue = TRUE,
  resolution_level = "lowres",
  ...
)
```

Arguments

dir.path	path to Visium output folder
bin.size	bin size of the VisiumHD output (Exp: "2", "8" and "16")
selected_assay	selected assay from Visium
assay_name	the assay name
sample_name	the name of the sample
image_name	the image name of the Visium assay, Default: main

channel_name	the channel name of the image of the Visium assay, Default: H&E
inTissue	if TRUE, only barcodes that are in the tissue will be kept (default: TRUE)
resolution_level	the level of resolution of Visium image: "lowres" (default) or "hires"
...	additional parameters passed to formVoltRon

---

importXenium *importXenium*

Description

Importing Xenium data

Usage

```
importXenium(
  dir.path,
  selected_assay = "Gene Expression",
  assay_name = "Xenium",
  sample_name = NULL,
  use_image = TRUE,
  morphology_image = "morphology_lowres.tif",
  resolution_level = 7,
  overwrite_resolution = TRUE,
  image_name = "main",
  channel_name = "DAPI",
  import_molecules = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

dir.path	path to Xenium output folder
selected_assay	selected assay from Xenium
assay_name	the assay name of the SR object
sample_name	the name of the sample
use_image	if TRUE, the DAPI image will be used.
morphology_image	the name of the lowred morphology image. Default: morphology_lowres.tif
resolution_level	the level of resolution within Xenium OME-TIFF image, see generateXeniumImage. Default: 7 (553x402)

overwrite_resolution
: if TRUE, the image "file.name" will be generated again although it exists at "dir.path"

image_name
: the image name of the Xenium assay, Default: main

channel_name
: the channel name of the image of the Xenium assay, Default: DAPI

import_molecules
: if TRUE, molecule assay will be created along with cell assay.

verbose
: verbose

...
: additional parameters passed to formVoltRon

---

knn_annoy *knn_annoy*

Description

knn engine employed by RcppAnnoy package, adapted from `BPCells` package.

Usage

```
knn_annoy(data, query = data, k = 10, n_trees = 50, search_k = -1)
```

Arguments

`data`	data
`query`	query data (Default: data)
`k`	number of neighbors for kNN
`n_trees`	Number of trees during index build time. More trees gives higher accuracy
`search_k`	Number of nodes to inspect during the query, or -1 for default value. Higher number gives higher accuracy

Details

knn_annoy: Use RcppAnnoy as knn engine

loadVoltRon *loadVoltRon*

Description

load VoltRon object from memory or disk

Usage

```
loadVoltRon(dir = "my_se")
```

Arguments

dir	the directory that VoltRon object is found.

melc_data *Example MELC Data*

Description

This VoltRon object is used for testing and as an example

Usage

```
melc_data
```

Format

An object of class `VoltRon` of length 1.

Source

Created to serve as an example.

Examples

```
data(melc_data)
```

`merge,VoltRon,ANY-method`

Merging VoltRon objects

Description

Given a VoltRon object, and a list of VoltRon objects, merge all.

Usage

```
## S4 method for signature 'VoltRon,ANY'
merge(x, y, samples = NULL, main.assay = NULL, verbose = TRUE)
```

Arguments

`x`	a VoltRon Object
`y`	a single or a list of VoltRon objects
`samples`	a single sample name or multiple sample names of the same size as the given VoltRon objects
`main.assay`	the name of the main assay
`verbose`	verbose

`merge,vrBlock,ANY-method`

Merging vrBlock objects

Description

Given a vrBlock object, and a list of vrSample objects, merge all.

Usage

```
## S4 method for signature 'vrBlock,ANY'
merge(x, y, samples = NULL)
```

Arguments

`x`	a vrSample object
`y`	a list of vrSample objects
`samples`	the sample names

merge,vrMetadata,ANY-method

Merging vrMetadata objects

Description

Given a vrMetadata object, and a list of vrMetadata objects, merge all.

Usage

```
## S4 method for signature 'vrMetadata,ANY'
merge(x, y)
```

Arguments

x	a vrMetadata object
y	a single or a list of vrMetadata objects

merge,vrSample,ANY-method

Merging vrSample objects

Description

Given a vrSample object, and a list of vrSample objects, merge all.

Usage

```
## S4 method for signature 'vrSample,ANY'
merge(x, y, samples = NULL)
```

Arguments

x	a vrSample object
y	a list of vrSample objects
samples	the sample names

merged_object *Example Transfer Data*

Description

This VoltRon object is used for testing and as an example for transfering information between layers and multilayer plotting.

Usage

```
merged_object
```

Format

An object of class `VoltRon` of length 1.

Source

Created to serve as an example.

Examples

```
data(merged_object)
```

Metadata *Metadata*

Description

Get the metadata of a VoltRon object.

Usage

```
Metadata(object, ...)

Metadata(object, ...) <- value

## S4 method for signature 'VoltRon'
Metadata(object, assay = NULL, type = NULL)

## S4 replacement method for signature 'VoltRon'
Metadata(object, assay = NULL, type = NULL) <- value
```

Arguments

object	a VoltRon object.
...	arguments passed to other methods.
value	new metadata
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
type	the assay type: ROI, spot or cell, or all for the entire metadata object

---

`modulateImage` *modulateImage*

Description

Modulating Magick images

Usage

```
modulateImage(object, ...)

## S4 method for signature 'VoltRon'
modulateImage(
  object,
  assay = NULL,
  name = NULL,
  reg = FALSE,
  channel = NULL,
  brightness = 100,
  saturation = 100,
  hue = 100,
  force = FALSE
)

## S4 method for signature 'vrAssay'
modulateImage(
  object,
  name = NULL,
  reg = FALSE,
  channel = NULL,
  brightness = 100,
  saturation = 100,
  hue = 100,
  force = FALSE
)

## S4 method for signature 'vrAssayV2'
```

```
modulateImage(
  object,
  name = NULL,
  reg = FALSE,
  channel = NULL,
  brightness = 100,
  saturation = 100,
  hue = 100,
  force = FALSE
)

## S4 method for signature 'vrImage'
modulateImage(
  object,
  channel = NULL,
  brightness = 100,
  saturation = 100,
  hue = 100,
  force = FALSE
)

## S4 method for signature 'vrSpatial'
modulateImage(
  object,
  channel = NULL,
  brightness = 100,
  saturation = 100,
  hue = 100,
  force = FALSE
)
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
name	the name of the image
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
channel	the name of the channel associated with the image
brightness	modulation of brightness as percentage of the current value (100 for no change)
saturation	modulation of saturation as percentage of the current value (100 for no change)
hue	modulation of hue is an absolute rotation of -180 degrees to +180 degrees from the current position corresponding to an argument range of 0 to 200 (100 for no change)
force	if TRUE, all channels will be modulated given no specific channel name

normalizeData *Normalize Data*

Description

Given a VoltRon or vrAssay object, normalize the raw count data.

Usage

```
normalizeData(object, ...)

## S4 method for signature 'VoltRon'
normalizeData(
  object,
  assay = NULL,
  method = "LogNorm",
  desiredQuantile = 0.9,
  scale = 0.2,
  sizefactor = 10000,
  feat_type = NULL
)

## S4 method for signature 'vrAssay'
normalizeData(
  object,
  method = "LogNorm",
  desiredQuantile = 0.9,
  scale = 0.2,
  sizefactor = 10000,
  feat_type = NULL
)

## S4 method for signature 'vrAssayV2'
normalizeData(
  object,
  method = "LogNorm",
  desiredQuantile = 0.9,
  scale = 0.2,
  sizefactor = 10000,
  feat_type = NULL
)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.

assay	assay name (exp: Assay1) or assay class if NULL, the default assay will be used, see vrMainAssay.
method	the normalization method: "LogNorm", "Q3Norm", "LogQ3Norm", "CLR" or "hyper.arcsine".
desiredQuantile	the quantile of the data if "QuanNorm" or "LogQuanNorm" is selected as method.
scale	the scale parameter for the hyperbolic arcsine transformation
sizefactor	size factor if method is selected as LogNorm
feat_type	the feature set type

---

open_zarr *open_zarr*

Description

open zarr store

Usage

```
open_zarr(dir, name)
```

Arguments

dir	the location of zarr store
name	name of the zarr store

---

registerSpatialData *registerSpatialData*

Description

A mini shiny app for registering images and spatial coordinates of multiple consecutive VoltRon objects.

Usage

```
registerSpatialData(
  object_list = NULL,
  reference_spatdata = NULL,
  query_spatdata = NULL,
  keypoints = NULL,
  mapping_parameters = list(),
  interactive = TRUE,
 shiny.options = list(launch.browser = getOption("shiny.launch.browser", interactive()))
)
```

Arguments

object_list	a list of VoltRon (or Seurat) objects
reference_spatdata	a reference spatial data set, used only if object_list is NULL
query_spatdata	a query spatial data set, used only if object_list is NULL
keypoints	(DEPRECATED) a list of tables, each points to matching keypoints from registered images.
mapping_parameters	for manual image registration, a list of tables, each points to matching keypoints from registered images, and for automated image registration, a set of mapping parameters
interactive	if TRUE, the shiny application for image registration will be triggered, otherwise 'mapping_parameters' or 'keypoints' should be defined.
shiny.options	a list of shiny options (launch.browser, host, port etc.) passed options argument of shinyApp. For more information, see runApp

---

resizeImage *resizeImage*

Description

Resizing Magick images

Usage

```
resizeImage(object, ...)

## S4 method for signature 'VoltRon'
resizeImage(object, assay = NULL, name = NULL, reg = FALSE, size = NULL)

## S4 method for signature 'vrAssay'
resizeImage(object, name = NULL, reg = FALSE, size = NULL)

## S4 method for signature 'vrAssayV2'
resizeImage(object, name = NULL, reg = FALSE, size = NULL)

## S4 method for signature 'vrImage'
resizeImage(object, size = NULL)

## S4 method for signature 'vrSpatial'
resizeImage(object, size = NULL)
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
name	the name of the image
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
size	the width of the resized image

SampleMetadata *SampleMetadata*

Description

Get the sample metadata of a VoltRon object

Usage

```
SampleMetadata(object)
```

Arguments

object	a VoltRon object

---

saveVoltRon *saveVoltRon*

Description

save VoltRon object in memory or on disk

Usage

```
saveVoltRon(
  object,
  assay = NULL,
  format = c("InMemoryVoltRon", "HDF5VoltRon", "ZarrVoltRon"),
  output = NULL,
  replace = FALSE,
  chunkdim = NULL,
  level = NULL,
  as.sparse = FALSE,
  verbose = TRUE,
  feature.vs.obs.engine = "BPCells"
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata.
format	the format the object should be written: InMemoryVoltRon (rds only), HDF5VoltRon (h5), or ZarrVoltRon (zarr).
output	When saving, the directory will be created if it doesn't already exist. If the directory already exists and no prefix is specified and replace is set to TRUE, then it's replaced with an empty directory.
replace	When no prefix is specified, should a pre-existing directory be replaced with a new empty one? The content of the pre-existing directory will be lost!
chunkdim	The dimensions of the chunks to use for writing the assay data to disk.
level	The compression level to use for writing the assay data to disk.
as.sparse	Whether the dataset (for HDF5 DelayedArray) should be flagged as sparse or not.
verbose	verbose
feature.vs.obs.engine	The on-disk method for the manipulating feature x obs matrices: BPCells or DelayedArray

---

slotApply *slotApply*

Description

apply to slots

Usage

```
slotApply(x, FUN, ...)
```

Arguments

x	object
FUN	function
...	arguments passed to FUN

slotToList *slotToList*

Description

slot to list

Usage

```
slotToList(x)
```

Arguments

x	object

subset,VoltRon-method *Subsetting VoltRon objects*

Description

Given a VoltRon object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'VoltRon'
subset(
  x,
  subset,
  samples = NULL,
  assays = NULL,
  spatialpoints = NULL,
  features = NULL,
  image = NULL,
  interactive = FALSE,
  use.points.only = FALSE,
  shiny.options = list(launch.browser = getOption("shiny.launch.browser", interactive()))
)
```

Arguments

x	a VoltRon object
subset	Logical statement for subsetting
samples	the set of samples to subset the object
assays	the set of assays to subset the object

spatialpoints	the set of spatial points to subset the object
features	the set of features to subset the object
image	the subseting string passed to image_crop
interactive	TRUE if interactive subsetting on the image is demanded
use.points.only	if interactive is TRUE, use spatial points instead of the reference image
shiny.options	a list of shiny options (launch.browser, host, port etc.) passed options argument of shinyApp. For more information, see runApp

Examples

```
# example data
data("visium_data")

# subset based on assay
subset(visium_data, assays = "Assay1")
subset(visium_data, assays = "Visium")

# subset based on samples
subset(visium_data, samples = "Anterior1")

# subset based on assay
subset(visium_data, spatialpoints = c("GTTATATTATCTCCCT-1_Assay1", "GTTTGGGTTTCGCCCG-1_Assay1"))

# subset based on features
subset(visium_data, features = c("Map3k19", "Rab3gap1"))

# interactive subsetting
## Not run:
visium_subset_data <- subset(visium_data, interactive = TRUE)
visium_subset <- visium_subset_data$subsets[[1]]

## End(Not run)
```

---

subset,vrAssay-method *Subsetting vrAssay objects*

Description

Given a vrAssay object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrAssay'
subset(x, subset, spatialpoints = NULL, features = NULL, image = NULL)
```

Arguments

x	a vrAssay object
subset	Logical statement for subsetting
spatialpoints	the set of spatial points to subset the object
features	the set of features to subset the object
image	the subseting string passed to image_crop

---

subset,vrAssayV2-method

Subsetting vrAssayV2 objects

Description

Given a vrAssayV2 object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrAssayV2'
subset(x, subset, spatialpoints = NULL, features = NULL, image = NULL)
```

Arguments

x	a vrAssayV2 object
subset	Logical statement for subsetting
spatialpoints	the set of spatial points to subset the object
features	the set of features to subset the object
image	the subseting string passed to image_crop

---

subset,vrBlock-method *Subsetting vrBlock objects*

Description

Given a vrBlock object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrBlock'
subset(x, subset, assays = NULL, spatialpoints = NULL, image = NULL)
```

Arguments

x	a vrSample object
subset	the subset statement
assays	the set of assays to subset the object
spatialpoints	the set of spatial points to subset the object
image	the subseting string passed to image_crop

---

subset,vrImage-method *Subsetting vrImage objects*

Description

Given a vrImage object, subset the object given one of the attributes.

Usage

```
## S4 method for signature 'vrImage'
subset(x, subset, spatialpoints = NULL, image = NULL)
```

Arguments

x	A vrImage object
subset	Logical statement for subsetting
spatialpoints	the set of spatial points to subset the object
image	the subseting string passed to image_crop

---

subset,vrLayer-method *Subsetting vrLayer objects*

Description

Given a vrLayer object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrLayer'
subset(x, subset, assays = NULL, spatialpoints = NULL, image = NULL)
```

Arguments

x	a vrLayer object
subset	the subset statement
assays	the set of assays to subset the object
spatialpoints	the set of spatial points to subset the object
image	the subseting string passed to image_crop

`subset,vrMetadata-method`

Subsetting vrMetadata objects

Description

Given a vrMetadata object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrMetadata'
subset(x, subset, samples = NULL, assays = NULL, spatialpoints = NULL)
```

Arguments

`x`	a vrMetadata object
`subset`	the subset statement
`samples`	the set of samples to subset the object
`assays`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see `SampleMetadata(object)`
`spatialpoints`	the set of spatial points to subset the object

`subset,vrSample-method`

Subsetting vrSample objects

Description

Given a vrSample object, subset the object given one of the attributes

Usage

```
## S4 method for signature 'vrSample'
subset(x, subset, assays = NULL, spatialpoints = NULL, image = NULL)
```

Arguments

`x`	a vrSample object
`subset`	the subset statement
`assays`	the set of assays to subset the object
`spatialpoints`	the set of spatial points to subset the object
`image`	the subseting string passed to image_crop

subset,vrSpatial-method

Subsetting vrSpatial objects

Description

Given a vrSpatial object, subset the object given one of the attributes.

Usage

```
## S4 method for signature 'vrSpatial'
subset(x, subset, spatialpoints = NULL, image = NULL)
```

Arguments

x	A vrSpatial object
subset	Logical statement for subsetting
spatialpoints	the set of spatial points to subset the object
image	the subseting string passed to image_crop

subsetCoordinates *subsetCoordinates*

Description

subsetting coordinates given cropping parameters of a magick image objects

Usage

```
subsetCoordinates(coords, imageinfo, crop_info)
```

Arguments

coords	the coordinates of the spatial points
imageinfo	the magick image info associated with the image
crop_info	the subseting string passed to image_crop

subsetSegments *subsetSegments*

Description

subsetting segments given cropping parameters of a magick image objects

Usage

```
subsetSegments(segments, imageinfo, crop_info)
```

Arguments

segments	the list of segments each associated with a spatial point
imageinfo	the magick image info associated with the image
crop_info	the subseting string passed to image_crop

transferData *transferData*

Description

transfer data across assays

Usage

```
transferData(
  object,
  from = NULL,
  to = NULL,
  features = NULL,
  expand = NULL,
  new_feature_name = NULL
)
```

Arguments

object	a VoltRon object
from	the name or class of assay whose data transfered to the second assay
to	the name or class of target assay where data is transfered to
features	the set of features from vrFeatures or metadata columns from Metadata that are transferred. Only one metadata feature can be transfered at a time.

expand — if TRUE, metadata features will be transformed into dummy features where each category in the feature will be a new feature. If FALSE, metadata features will not be transformed and transfered as metadata columns, else the decision will be made automatically.

new_feature_name — the name of the new feature set created from the source assay defined in `from` argument. Only used when a new assay in created.

`updateAssay` *Update Assay*

Description

update assays from vrAssay to vrAssayV2

Usage

```
updateAssay(object, ...)

## S4 method for signature 'vrAssay'
updateAssay(object)

## S4 method for signature 'vrAssayV2'
updateAssay(object)

## S4 method for signature 'VoltRon'
updateAssay(object, assay = NULL)
```

Arguments

`object`	a vrAssayV2 object to be converted to vrAssayV2
`...`	arguments passed to other methods.
`assay`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

`visium_data` *Example Visium Data*

Description

This VoltRon object is used for testing and as an example

Usage

```
visium_data
```

Format

An object of class `VoltRon` of length 1.

Source

Created to serve as an example.

Examples

```
data(visium_data)
```

`VoltRon-class` *The VoltRon Class*

Description

The VoltRon Class

Slots

`samples` A list of samples (vrSample)

`metadata` A vrMetadata object that includes metadata of ROIs, spots, and cells

`sample.metadata` Contains meta-information about each sample, layer and assay

`graph` A igraph object

`main.assay` The type of the main assay (i.e. Visium, Xenium, GeoMx etc.)

`project` Name of the project

`VoltRon-methods` *Methods for VoltRon*

Description

Methods for `VoltRon` objects for generics defined in other packages

Usage

```
## S3 method for class 'VoltRon'
x$i, ...

## S3 replacement method for class 'VoltRon'
x$i <- value

## S3 method for class 'VoltRon'
.DollarNames(x, pattern = "")

## S4 method for signature 'VoltRon,character,missing'
x[[i, j, ...]]

## S4 replacement method for signature 'VoltRon,character,missing'
x[[i, j, ...]] <- value

## S4 method for signature 'VoltRon,character,character'
x[[i, j, ...]]

## S4 replacement method for signature 'VoltRon,character,character'
x[[i, j, ...]] <- value
```

Arguments

x	A VoltRon object
i, value	Depends on the usage
	$, $<- Name (i) of a single metadata column from the main assay, see vrMainAssay
	[[, [[<- If only i is given, either a vrSample object or a vrAssay for i (and value) being name of the sample or assay. If both i and j are given, vrLayer with layer name j (and value) of vrSample with same name i.
...	Arguments passed to other methods
pattern	A regular expression. Only matching names are returned.
j	Depends on the usage, see i.

Value

[[<-: x with the metadata or associated objects added as i; if value is NULL, removes metadata or associated object i from object x

[[<-: x with the metadata or associated objects added as i; if value is NULL, removes metadata or associated object i from object x

Functions

- $: Metadata access for VoltRon objects
- `$`(VoltRon) <- value: Metadata overwrite for VoltRon objects

- `.DollarNames(VoltRon)`: Autocompletion for `$` access for `VoltRon` objects
- `x[[i`: Accessing vrAssay or vrSample objects from `VoltRon` objects
- `` `[[`(x = VoltRon, i = character, j = missing) <- value ``: Overwriting vrAssay or vrSample objects from `VoltRon` objects
- `x[[i`: Accessing vrLayer objects from `VoltRon` objects
- `` `[[`(x = VoltRon, i = character, j = character) <- value ``: Overwriting vrLayer objects from `VoltRon` objects

`vrAssay-class` *The vrAssay (VoltRon Assay) Class*

Description

The vrAssay (VoltRon Assay) Class

Slots

`rawdata` raw data

`normdata` normalized data

`featuredata` feature metadata

`embeddings` list of embeddings

`image` a list of vrImage objects

`params` additional parameters used by different assay types

`type` the type of the assay (tile, molecule, cell, spot, ROI)

`name` the assay name

`main_image` the key of the main image

`vrAssayNames` *Get Assay names*

Description

Given a VoltRon, vrMetadata or vrAssay object, get/set names of assays.

Usage

```
vrAssayNames(object, ...)

## S4 method for signature 'VoltRon'
vrAssayNames(object, assay = NULL)

## S4 method for signature 'vrMetadata'
vrAssayNames(object)

## S4 method for signature 'vrAssay'
vrAssayNames(object)

## S4 method for signature 'vrAssayV2'
vrAssayNames(object)

## S4 replacement method for signature 'vrAssay'
vrAssayNames(object) <- value

## S4 replacement method for signature 'vrAssayV2'
vrAssayNames(object) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
value	assay name

---

vrAssayParams *Get assay parameters*

Description

Given a vrAssay object, if there are any, get a list of parameters of the assay(s)

Usage

```
vrAssayParams(object, param = NULL)
```

Arguments

object	a vrAssay object
param	the parameter value to return

vrAssayTypes *Get assay types*

Description

Given a VoltRon or vrAssay object, get types of assays. Here, an assay type is of either tile, molecule, cell, spot or ROI.

Usage

```
vrAssayTypes(object, ...)

## S4 method for signature 'VoltRon'
vrAssayTypes(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrAssayTypes(object)

## S4 method for signature 'vrAssayV2'
vrAssayTypes(object)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

vrAssayV2-class *The vrAssayV2 (VoltRon Assay) Class*

Description

The vrAssayV2 (VoltRon Assay) Class

Slots

`data` list of count/normalized datasets

`featuredata` list of feature metadata

`embeddings` list of embeddings

`image` a list of vrImage objects

`params` additional parameters used by different assay types

`type` the type of the assay (tile, molecule, cell, spot, ROI)

`name` the assay name

`main_image` the key of the main image

`main_featureset` the key of the main feature set

`vrBarPlot` *vrBarPlot*

Description

vrBarPlot

Usage

```
vrBarPlot(
  object,
  features = NULL,
  assay = NULL,
  x.label = NULL,
  group.by = "Sample",
  split.by = NULL,
  norm = TRUE,
  log = FALSE,
  ncol = 2,
  nrow = NULL
)
```

Arguments

`object`	a VoltRon object
`features`	a set of features to be visualized, either from vrFeatures of raw or normalized data or columns of the Metadata.
`assay`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
`x.label`	labels of the x axis
`group.by`	a column of metadata from Metadata used as grouping label for the spatial entities
`split.by`	the column to split the barplots by
`norm`	if TRUE, the normalized data is used
`log`	if TRUE, data features (excluding metadata features) will be log transformed
`ncol`	column wise number of plots, for ggarrange
`nrow`	row wise number of plots, for ggarrange

vrBlock-class *The vrBlock (VoltRon Block) Class*

Description

The vrBlock (VoltRon Block) Class

Slots

`layer` A list of layers (vrLayer)

`zlocation` a vector of z coordinates of layers

`adjacency` an adjacency matrix of connected layers within a block

vrCoordinates *vrCoordinates*

Description

Given a VoltRon, vrAssay or vrSpatial object, get and set the coordinates of assays and coordinate systems

Usage

```
vrCoordinates(object, ...)

## S4 method for signature 'VoltRon'
vrCoordinates(
  object,
  assay = NULL,
  image_name = NULL,
  spatial_name = NULL,
  reg = FALSE
)

## S4 method for signature 'vrAssay'
vrCoordinates(object, image_name = NULL, spatial_name = NULL, reg = FALSE)

## S4 method for signature 'vrAssayV2'
vrCoordinates(object, image_name = NULL, spatial_name = NULL, reg = FALSE)

## S4 method for signature 'vrImage'
vrCoordinates(object)

## S4 method for signature 'vrSpatial'
vrCoordinates(object)
```

```
## S4 replacement method for signature 'VoltRon'
vrCoordinates(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrAssay'
vrCoordinates(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrAssayV2'
vrCoordinates(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrImage'
vrCoordinates(object) <- value

## S4 replacement method for signature 'vrSpatial'
vrCoordinates(object) <- value

vrCoordinates(object, ...) <- value
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
image_name	(deprecated, use spatial_name) the name/key of the image associated with the coordinates
spatial_name	the name/key of the spatial system associated with the coordinates
reg	TRUE if registered coordinates of the main image (vrMainImage) is requested
value	coordinates

---

vrData *vrData*

Description

Get data of assays

Usage

```
vrData(object, ...)

## S4 method for signature 'VoltRon'
vrData(
  object,
  assay = NULL,
```

```
  features = NULL,
  feat_type = NULL,
  norm = FALSE,
  ...
)

## S4 method for signature 'vrAssay'
vrData(object, features = NULL, feat_type = NULL, norm = FALSE, ...)

## S4 method for signature 'vrAssayV2'
vrData(object, features = NULL, feat_type = NULL, norm = FALSE, ...)
```

Arguments

object	a VoltRon or vrAssay object.
...	additional parameters passed to other methods and vrImages
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
features	the set of features
feat_type	the feature set type
norm	TRUE if normalized data should be returned

`vrEmbeddingFeaturePlot` *vrEmbeddingFeaturePlot*

Description

Plotting features of spatially resolved cells and spots on embeddings from multiple assays in a VoltRon object.

Usage

```
vrEmbeddingFeaturePlot(
  object,
  embedding = "pca",
  features = NULL,
  combine.features = FALSE,
  n.tile = NULL,
  norm = TRUE,
  log = FALSE,
  assay = NULL,
  ncol = 2,
  nrow = NULL,
  font.size = 2,
```

```
  pt.size = 1,
  shape = 16,
  keep.scale = "feature",
  common.legend = TRUE,
  collapse.plots = TRUE
)
```

Arguments

object	a VoltRon object
embedding	the embedding type, i.e. pca, umap etc.
features	a set of features to be visualized, either from vrFeatures of raw or normalized data or columns of the Metadata.
combine.features	whether to combine all features in one plot
n.tile	The number of tiles on x-and y-axis for rasterizing points (see geom_tile). The rasterization is performed automatically for large number of points Only applicable to spots, cells and molecules. If `n.tile = 0` will turn of automated rasterization.
norm	if TRUE, the normalized data is used
log	if TRUE, data features (excluding metadata features) will be log transformed
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
ncol	column wise number of plots, for ggarrange
nrow	row wise number of plots, for ggarrange
font.size	font size
pt.size	point size
shape	shape
keep.scale	whether unify all scales for all features or not
common.legend	whether to use a common legend for all plots, see ggarrange
collapse.plots	whether to combine all ggplots

---

vrEmbeddingNames *Get Embedding names*

Description

Given a VoltRon or vrAssay object, give names of embeddings

Usage

```
vrEmbeddingNames(object, ...)

## S4 method for signature 'VoltRon'
vrEmbeddingNames(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrEmbeddingNames(object)

## S4 method for signature 'vrAssayV2'
vrEmbeddingNames(object)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

---

vrEmbeddingPlot *vrEmbeddingPlot*

Description

Plotting embeddings of cells and spots on associated images from multiple assays in a VoltRon object.

Usage

```
vrEmbeddingPlot(
  object,
  embedding = "pca",
  group.by = "Sample",
  group.ids = NULL,
  split.by = NULL,
  colors = NULL,
  n.tile = NULL,
  assay = NULL,
  ncol = 2,
  nrow = NULL,
  font.size = 5,
  pt.size = 1,
  shape = 16,
  label = FALSE,
  common.legend = TRUE,
  collapse.plots = TRUE
)
```

Arguments

object	a VoltRon object
embedding	the embedding type, i.e. pca, umap etc.
group.by	a column of metadata from Metadata used as grouping label for the spatial entities
group.ids	a subset of categories defined in metadata column from group.by
split.by	a column of metadata from Metadata used as splitting spatial entities into ggplot panels, see facet_wrap
colors	the color set for group.by. Should be of the same size of group.id (if specified) or unique elements in group.by
n.tile	The number of tiles on x-and y-axis for rasterizing points (see geom_tile). The rasterization is performed automatically for large number of points Only applicable to spots, cells and molecules. If n.tile = 0 will turn of automated rasterization.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
ncol	column wise number of plots, for facet_wrap
nrow	row wise number of plots, for facet_wrap
font.size	font size of labels, if label is TRUE
pt.size	point size
shape	shape
label	if TRUE, the labels of the ROI assays will be visualized
common.legend	whether to use a common legend for all plots, see ggarrange
collapse.plots	whether to combine all ggplots

---

vrEmbeddings *vrEmbeddings*

Description

Given a VoltRon or vrAssay object, get embeddings of spatial points

Usage

```
vrEmbeddings(object, ...)

## S4 method for signature 'VoltRon'
vrEmbeddings(object, assay = NULL, type = "pca", dims = seq_len(30))

## S4 method for signature 'vrAssay'
vrEmbeddings(object, type = "pca", dims = seq_len(30))
```

```
## S4 method for signature 'vrAssayV2'
vrEmbeddings(object, type = "pca", dims = seq_len(30))

## S4 replacement method for signature 'vrAssay'
vrEmbeddings(object, type = "pca") <- value

## S4 replacement method for signature 'vrAssayV2'
vrEmbeddings(object, type = "pca") <- value

## S4 replacement method for signature 'VoltRon'
vrEmbeddings(object, assay = NULL, type = "pca", overwrite = FALSE) <- value

vrEmbeddings(object, ...) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
type	the key name for the embedding
dims	the set of dimensions of the embedding data
value	new embedding data
overwrite	Whether the existing embedding with name 'type' should be overwritten

---

vrFeatureData *Feature Data*

Description

Get and set feature data from the main.assay

Usage

```
vrFeatureData(object, ...)

## S4 method for signature 'VoltRon'
vrFeatureData(object, assay = NULL, feat_type = NULL)

## S4 method for signature 'vrAssay'
vrFeatureData(object)

## S4 method for signature 'vrAssayV2'
vrFeatureData(object, feat_type = NULL)
```

```
## S4 replacement method for signature 'VoltRon'
vrFeatureData(object, assay = NULL) <- value

## S4 replacement method for signature 'vrAssay'
vrFeatureData(object) <- value

## S4 replacement method for signature 'vrAssayV2'
vrFeatureData(object, feat_type = NULL) <- value

vrFeatureData(object, ...) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
feat_type	the feature set type
value	new feature metadata

`vrFeatures` *vrFeatures*

Description

Get names of the features.

Usage

```
vrFeatures(object, ...)

## S4 method for signature 'VoltRon'
vrFeatures(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrFeatures(object)

## S4 method for signature 'vrAssayV2'
vrFeatures(object)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

vrFeatureTypeNames *vrFeatureTypeNames*

Description

Get the names of the feature types

Usage

```
vrFeatureTypeNames(object, ...)

## S4 method for signature 'vrAssayV2'
vrFeatureTypeNames(object)

## S4 method for signature 'vrAssay'
vrFeatureTypeNames(object)

## S4 method for signature 'VoltRon'
vrFeatureTypeNames(object, assay = NULL)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay. If given as "all", then provides a summary of spatial systems across all assays

vrGraph *vrGraph*

Description

Get graph of a VoltRon object

Usage

```
vrGraph(object, assay = NULL, graph.type = NULL)

vrGraph(object, assay = NULL, graph.type = "kNN") <- value
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
graph.type	the type of the graph, either custom or given by getProfileNeighbors or getSpatialNeighbors functions
value	new graph

---

vrGraphNames *vrGraphNames*

Description

Get names of all graphs

Usage

```
vrGraphNames(object, assay = NULL)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

---

vrHeatmapPlot *HeatmapPlot*

Description

HeatmapPlot

Usage

```
vrHeatmapPlot(
  object,
  assay = NULL,
  features = NULL,
  group.by = "clusters",
  norm = TRUE,
  scaled = TRUE,
  show_row_names = NULL,
  cluster_rows = TRUE,
  show_heatmap_legend = FALSE,
```

```
  outlier.quantile = 0.99,
  highlight.some = FALSE,
  n_highlight = 30,
  font.size = 13.2,
  ...
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
features	a set of features to be visualized from vrFeatures of raw or normalized data.
group.by	a column of metadata from Metadata used as grouping label for the spatial entities
norm	if TRUE, the normalized data is used
scaled	if TRUE, the data will be scaled before visualization
show_row_names	if TRUE, row names of the heatmap will be shown
cluster_rows	if TRUE, the rows of the heatmap will be clustered
show_heatmap_legend	if TRUE, the heatmap legend is shown
outlier.quantile	quantile for detecting outliers whose values are set to the quantile, change to lower values to adjust large number of outliers, default: 0.99
highlight.some	if TRUE, some rows will be showed at random, reproducible by seed argument
n_highlight	the number of row labels shown, if show_row_names is TRUE
font.size	font size
...	additional parameters passed to getVariableFeatures

---

vrImage-class *The vrImage (VoltRon Image) Class*

Description

The vrImage (VoltRon Image) Class

Slots

coords	spatial coordinates of the assay
segments	spatial coordinates of the segments, if available
image	image of the spatial assay, bitmap class
main_channel	the key of the main channel of vrImage object

`vrImageChannelNames` *vrImageChannelNames*

Description

Get names of all image channels

Usage

```
vrImageChannelNames(object, ...)

## S4 method for signature 'VoltRon'
vrImageChannelNames(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrImageChannelNames(object, name = NULL)

## S4 method for signature 'vrAssayV2'
vrImageChannelNames(object, name = NULL)

## S4 method for signature 'vrImage'
vrImageChannelNames(object, return.report = TRUE)

## S4 method for signature 'vrSpatial'
vrImageChannelNames(object, return.report = TRUE)
```

Arguments

`object`	a VoltRon, vrAssay or vrSpatial object.
`...`	arguments passed to other methods.
`assay`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
`name`	the key of the image
`return.report`	if TRUE and no image is present, return a character stating that there is no image

`vrImageNames` *vrImageNames*

Description

Get names of all images

Usage

```
vrImageNames(object, ...)

## S4 method for signature 'VoltRon'
vrImageNames(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrImageNames(object)

## S4 method for signature 'vrAssayV2'
vrImageNames(object)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay. If equals to "all", then provides a summary of spatial systems across all assays

---

vrImages *vrImages*

Description

Get images of VoltRon objects

Usage

```
vrImages(object, ...)

## S4 method for signature 'VoltRon'
vrImages(
  object,
  assay = NULL,
  name = NULL,
  reg = FALSE,
  channel = NULL,
  as.raster = FALSE,
  scale.perc = 100
)

## S4 method for signature 'vrAssay'
vrImages(
  object,
  name = NULL,
```

```
  reg = FALSE,
  channel = NULL,
  as.raster = FALSE,
  scale.perc = 100
)

## S4 method for signature 'vrAssayV2'
vrImages(
  object,
  name = NULL,
  reg = FALSE,
  channel = NULL,
  as.raster = FALSE,
  scale.perc = 100
)

## S4 method for signature 'vrImage'
vrImages(object, channel = NULL, as.raster = FALSE, scale.perc = 100)

## S4 method for signature 'vrSpatial'
vrImages(object, channel = NULL, as.raster = FALSE, scale.perc = 100)

## S4 replacement method for signature 'vrAssay'
vrImages(object, name = NULL, channel = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrAssayV2'
vrImages(object, name = NULL, channel = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrImage'
vrImages(object, channel = NULL) <- value

## S4 replacement method for signature 'vrSpatial'
vrImages(object, channel = NULL) <- value

vrImages(object, ...) <- value
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
name	the name of the main spatial system
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
channel	the name of the channel associated with the image
as.raster	return as raster
scale.perc	scale percentage if lower resolution image needed

value	a raster image or an image of magick-image object

---

vrLayer-class *The vrLayer (VoltRon Layer) Class*

Description

The vrLayer (VoltRon Layer) Class

Slots

assay A list of assays (vrAssay)

connectivity the connectivity graph

---

vrLayer-methods *Methods for vrLayer objects*

Description

Methods for vrLayer objects for generics defined in other packages

Usage

```
## S4 method for signature 'vrLayer,character,ANY'
x[[i]]

## S4 replacement method for signature 'vrLayer,character,ANY'
x[[i]] <- value
```

Arguments

x	A vrLayer object
i	the name of assay associated with the layer, see SampleMetadata
value	a vrAssayV2 object, see vrAssayV2

Functions

- x[[i: Accessing vrAssay objects from vrLayer objects
- `[[`(x = vrLayer, i = character, j = ANY) <- value: Overwriting vrAssay objects from vrLayer objects

vrMainAssay *Main Assay*

Description

Get and set the main assay of a VoltRon object

Get and set the main assay of a VoltRon object

Usage

```
vrMainAssay(object, ...)

vrMainAssay(object, ...) <- value

## S4 method for signature 'VoltRon'
vrMainAssay(object)

## S4 replacement method for signature 'VoltRon'
vrMainAssay(object) <- value
```

Arguments

object	a VoltRon object
...	arguments passed to other methods.
value	new assay name

vrMainChannel *vrMainChannel*

Description

Get and set the main channel name of the spatial system.

Usage

```
vrMainChannel(object, ...)

## S4 method for signature 'vrAssay'
vrMainChannel(object, name = NULL)

## S4 method for signature 'vrAssayV2'
vrMainChannel(object, name = NULL)

## S4 method for signature 'vrImage'
vrMainChannel(object)
```

```
## S4 method for signature 'vrSpatial'
vrMainChannel(object)

## S4 replacement method for signature 'vrAssay'
vrMainChannel(object, name = NULL) <- value

## S4 replacement method for signature 'vrAssayV2'
vrMainChannel(object, name = NULL) <- value

## S4 replacement method for signature 'vrImage'
vrMainChannel(object) <- value

## S4 replacement method for signature 'vrSpatial'
vrMainChannel(object) <- value

vrMainChannel(object, ...) <- value
```

Arguments

object	a vrAssay or vrSpatial object
...	arguments passed to other methods.
name	the name of the image
value	the name of main channel of the spatial system

`vrMainFeatureType` *vrMainFeatureType*

Description

Get the main feature type

Set the main image

Usage

```
vrMainFeatureType(object, ...)

## S4 method for signature 'VoltRon'
vrMainFeatureType(object, assay = NULL)

## S4 method for signature 'vrAssayV2'
vrMainFeatureType(object)

## S4 method for signature 'vrAssay'
vrMainFeatureType(object)
```

```
## S4 replacement method for signature 'VoltRon'
vrMainFeatureType(object, assay = NULL) <- value

## S4 replacement method for signature 'vrAssayV2'
vrMainFeatureType(object, ignore = FALSE) <- value

## S4 replacement method for signature 'vrAssay'
vrMainFeatureType(object, ignore = FALSE) <- value

vrMainFeatureType(object, ...) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay. If given as "all", then provides a summary of spatial systems across all assays.
value	the name of main feature set.
ignore	ignore if some assays dont have the feature set name

---

vrMainImage *vrMainImage*

Description

Get the main image

Set the main image

Usage

```
vrMainImage(object, ...)

## S4 method for signature 'VoltRon'
vrMainImage(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrMainImage(object)

## S4 method for signature 'vrAssayV2'
vrMainImage(object)

## S4 replacement method for signature 'VoltRon'
vrMainImage(object, assay = NULL) <- value

## S4 replacement method for signature 'vrAssay'
```

```
vrMainImage(object, ignore = FALSE) <- value

## S4 replacement method for signature 'vrAssayV2'
vrMainImage(object, ignore = FALSE) <- value

vrMainImage(object, ...) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay. If given as "all", then provides a summary of spatial systems across all assays.
value	the name of main spatial coordinate system.
ignore	if TRUE, the non-existing spatial coordinate system will be ignored.

---

vrMainSpatial *vrMainSpatial*

Description

Get the main spatial system name

Set the main image

Usage

```
vrMainSpatial(object, ...)

## S4 method for signature 'VoltRon'
vrMainSpatial(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrMainSpatial(object)

## S4 method for signature 'vrAssayV2'
vrMainSpatial(object)

## S4 replacement method for signature 'VoltRon'
vrMainSpatial(object, assay = NULL) <- value

## S4 replacement method for signature 'vrAssay'
vrMainSpatial(object, ignore = FALSE) <- value

## S4 replacement method for signature 'vrAssayV2'
vrMainSpatial(object, ignore = FALSE) <- value

vrMainSpatial(object, ...) <- value
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. If NULL, the default assay will be used, see vrMainAssay.
value	the name of main spatial coordinate system
ignore	if TRUE, the non-existing spatial coordinate system will be ignored.

---

vrMetadata-class *The vrMetadata (VoltRon Metadata) Class*

Description

The vrMetadata (VoltRon Metadata) Class

Slots

`tile` the metadata of tiles

`molecule` the metadata of molecules

`cell` the metadata of cells

`spot` the metadata of spot

`ROI` the metadata of ROI

---

vrNeighbourhoodEnrichment *vrNeighbourhoodEnrichment*

Description

Conduct Neighborhood enrichment test for pairs of clusters for all assays

Usage

```
vrNeighbourhoodEnrichment(
  object,
  assay = NULL,
  group.by = NULL,
  graph.type = "delaunay",
  num.sim = 1000,
  seed = 1,
  verbose = TRUE
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
group.by	a column from metadata to separate spatial points
graph.type	the type of graph to determine spatial neighborhood
num.sim	the number of simulations
seed	seed
verbose	verbose

---

vrNeighbourhoodEnrichmentPlot

vrNeighbourhoodEnrichmentPlot

Description

Plotting results of vrNeighbourhoodEnrichment

Usage

```
vrNeighbourhoodEnrichmentPlot(
  results,
  assay = NULL,
  type = c("assoc", "segreg")
)
```

Arguments

results	The results from vrNeighbourhoodEnrichment
assay	assay name (exp: Assay1), see SampleMetadata.
type	the type of spatial test. Either "assoc" for association test or "segreg" for segregation test

vrProportionPlot *vrPercentagePlot*

Description

vrPercentagePlot

Usage

```
vrProportionPlot(
  object,
  assay = NULL,
  x.label = NULL,
  split.by = NULL,
  split.method = "facet_wrap",
  ncol = 2,
  nrow = NULL
)
```

Arguments

object	a VoltRon object
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
x.label	labels of the x axis
split.by	the column to split the barplots by
split.method	either facet_grid or facet_wrap, not affected if split.by is NULL
ncol	column wise number of plots, for ggarrange
nrow	row wise number of plots, for ggarrange

vrSample-class *The vrSample (VoltRon Sample) Class*

Description

The vrSample (VoltRon Sample) Class

Slots

layer A list of layers (vrLayer)

zlocation a vector of z coordinates of layers

adjacency an adjacency matrix of connected layers within a block

vrSample-methods *Methods for vrSample objects*

Description

Methods for vrSample objects for generics defined in other packages

Usage

```
## S4 method for signature 'vrSample,character,ANY'
x[[i]]

## S4 replacement method for signature 'vrSample,character,ANY'
x[[i]] <- value

## S4 method for signature 'vrBlock,character,ANY'
x[[i]]

## S4 replacement method for signature 'vrBlock,character,ANY'
x[[i]] <- value
```

Arguments

x	A vrSample object
i	the name of layer associated with the sample, see SampleMetadata
value	a vrLayer object, see vrLayer

Functions

- x[[i: Accessing vrLayer objects from vrSample objects
- `[[`(x = vrSample, i = character, j = ANY) <- value: Accessing vrLayer objects from vrSample objects
- x[[i: (deprecated) Accessing vrLayer objects from vrBlock objects
- `[[`(x = vrBlock, i = character, j = ANY) <- value: (deprecated) Overwriting vrLayer objects from vrBlock objects

vrSampleNames *Get Sample names*

Description

Given a vrMetadata object, give names of samples

Usage

```
vrSampleNames(object, ...)

## S4 method for signature 'vrMetadata'
vrSampleNames(object)
```

Arguments

object	a vrMetadata object.
...	arguments passed to other methods.

vrScatterPlot *vrScatterPlot*

Description

get a scatter plot between two features

Usage

```
vrScatterPlot(
  object,
  feature.1,
  feature.2,
  norm = TRUE,
  assay = NULL,
  pt.size = 2,
  font.size = 2,
  group.by = "label",
  label = FALSE,
  trend = FALSE
)
```

Arguments

object	a VoltRon object
feature.1	first feature
feature.2	second feature
norm	if TRUE, the normalize data will be used
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
pt.size	point size
font.size	font size
group.by	a column of metadata from Metadata used as grouping label for the spatial entities
label	whether labels are visualized or not
trend	inserting a trend line two the scatter plot

---

vrSegments *vrSegments*

Description

Given a VoltRon, vrAssay or vrSpatial object, get and set the segment coordinates of assays and coordinate systems

Usage

```
vrSegments(object, ...)

## S4 method for signature 'VoltRon'
vrSegments(
  object,
  assay = NULL,
  image_name = NULL,
  spatial_name = NULL,
  reg = FALSE,
  as.data.frame = FALSE
)

## S4 method for signature 'vrAssay'
vrSegments(object, image_name = NULL, spatial_name = NULL, reg = FALSE)

## S4 method for signature 'vrAssayV2'
vrSegments(object, image_name = NULL, spatial_name = NULL, reg = FALSE)

## S4 method for signature 'vrImage'
vrSegments(object)
```

```
## S4 method for signature 'vrSpatial'
vrSegments(object)

## S4 replacement method for signature 'VoltRon'
vrSegments(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrAssay'
vrSegments(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrAssayV2'
vrSegments(object, image_name = NULL, spatial_name = NULL, reg = FALSE) <- value

## S4 replacement method for signature 'vrImage'
vrSegments(object) <- value

## S4 replacement method for signature 'vrSpatial'
vrSegments(object) <- value

vrSegments(object, ...) <- value
```

Arguments

object	a VoltRon, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
image_name	(deprecated, use spatial_name) the name/key of the image associated with the coordinates
spatial_name	the name/key of the spatial system associated with the coordinates
reg	TRUE if registered coordinates of the main image (vrMainImage) is requested
as.data.frame	if TRUE, the coordinates of segment nodes will be returned as a data frame
value	segments

---

vrSpatial-class *The vrSpatial (VoltRon Spatial) Class*

Description

The vrSpatial (VoltRon Spatial) Class

Slots

`coords` spatial coordinates of the assay

`segments` spatial coordinates of the segments, if available

`image` image of the spatial assay, bitmap class

`main_channel` the key of the main channel of vrImage object

`vrSpatialFeaturePlot` *vrSpatialFeaturePlot*

Description

Plotting single/multiple features of spatially resolved cells, spots, and ROI on associated images from multiple assays in a VoltRon object.

Usage

```
vrSpatialFeaturePlot(
  object,
  features,
  combine.features = FALSE,
  group.by = "label",
  plot.segments = FALSE,
  n.tile = NULL,
  norm = TRUE,
  log = FALSE,
  assay = NULL,
  graph.name = NULL,
  ncol = 2,
  nrow = NULL,
  font.size = 2,
  pt.size = 2,
  cell.shape = 16,
  title.size = 10,
  alpha = 0.6,
  keep.scale = "feature",
  label = FALSE,
  spatial = NULL,
  channel = NULL,
  background.color = NULL,
  background = NULL,
  reg = FALSE,
  crop = FALSE,
  scale.image = TRUE,
  common.legend = FALSE,
  legend.loc = "right",
  collapse.plots = TRUE
)
```

Arguments

object	a VoltRon object
features	a set of features to be visualized, either from vrFeatures of raw or normalized data or columns of the Metadata.
combine.features	whether to combine all features in one plot
group.by	a column of metadata from Metadata used as grouping label for the spatial entities
plot.segments	plot segments from vrSegments instead of points
n.tile	The number of tiles on x-and y-axis for rasterizing points (see geom_tile). The rasterization is performed automatically for large number of points Only applicable to spots, cells and molecules. If `n.tile = 0` will turn of automated rasterization.
norm	if TRUE, the normalized data is used
log	if TRUE, data features (excluding metadata features) will be log transformed
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
graph.name	if not NULL, the spatial graph is with name `graph.name` is visualized as well, see vrGraphNames
ncol	column wise number of plots, for ggarrange
nrow	row wise number of plots, for ggarrange
font.size	font size
pt.size	point size
cell.shape	the shape of the points representing cells, see geom_point
title.size	title size of legend and plot
alpha	alpha level of colors of visualized points and segments
keep.scale	whether unify all scales for all features or not
label	if TRUE, labels of ROIs will be visualized too
spatial	the name of the main spatial system
channel	the name of the channel associated with the image
background.color	the color of plot background if a channel is not specified, or the spatial coord system doesnt have an image.
background	(DEPRECATED) the background of the plot. Either an image name, see vrImageNames or a vector of length two with image name and a channel name, see vrImageChannelNames. Type "black" or "white" for black or white backgrounds. if NULL, the main image (vrMainSpatial) and main channel (vrMainChannel) will be in the background. Otherwise the background will be grey.
reg	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
crop	whether to crop an image of a spot assay to the extend of spots

scale.image	if TRUE, background image will be scaled down to a low resolution (width: 1000px)
common.legend	whether to use a common legend for all plots, see ggarrange
legend.loc	the location of the legend, default is "right"
collapse.plots	whether to combine all ggplots

---

vrSpatialNames *vrSpatialNames*

Description

Get names of all spatial systems

Usage

```
vrSpatialNames(object, ...)

## S4 method for signature 'VoltRon'
vrSpatialNames(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrSpatialNames(object)

## S4 method for signature 'vrAssayV2'
vrSpatialNames(object)
```

Arguments

object	a VoltRon or vrAssay object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay. If equals to "all", then provides a summary of spatial systems across all assays

---

vrSpatialPlot *vrSpatialPlot*

Description

Plotting identification of spatially resolved cells, spots, and ROI on associated images from multiple assays in a VoltRon object.

Usage

```
vrSpatialPlot(
  object,
  group.by = "Sample",
  plot.segments = FALSE,
  group.ids = NULL,
  colors = NULL,
  n.tile = NULL,
  assay = NULL,
  graph.name = NULL,
  graph.edge.color = "orange",
  reduction = NULL,
  ncol = 2,
  nrow = NULL,
  font.size = 2,
  pt.size = 2,
  cell.shape = 21,
  alpha = 1,
  label = FALSE,
  spatial = NULL,
  channel = NULL,
  background.color = NULL,
  background = NULL,
  reg = FALSE,
  crop = FALSE,
  combine.groups = FALSE,
  legend.pt.size = 2,
  legend.text.size = 14,
  scale.image = TRUE,
  legend.loc = "right",
  common.legend = TRUE,
  collapse.plots = TRUE,
  interactive = FALSE,
  shiny.options = list()
)
```

Arguments

`object`	a VoltRon object
`group.by`	a column of metadata from Metadata used as grouping label for the spatial entities
`plot.segments`	plot segments from vrSegments instead of points
`group.ids`	a subset of categories defined in metadata column from `group.by`
`colors`	the color set for group.by. Should be of the same size of group.id (if specified) or unique elements in group.by
`n.tile`	The number of tiles on x-and y-axis for rasterizing points (see geom_tile). The rasterization is performed automatically for large number of points Only ap-

	plicable to spots, cells and molecules. If `n.tile = 0` will turn of automated rasterization.
`assay`	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
`graph.name`	if not NULL, the spatial graph is with name `graph.name` is visualized as well, see vrGraphNames
`graph.edge.color`	the colors of the graph edges, if `graph.name` is not NULL.
`reduction`	used by `vrSpatialPlotVitessce` to visualize an embedding alongside with the spatial plot.
`ncol`	column wise number of plots, for ggarrange
`nrow`	row wise number of plots, for ggarrange
`font.size`	font size
`pt.size`	point size
`cell.shape`	the shape of the points representing cells, see geom_point
`alpha`	alpha level of colors of visualized points and segments
`label`	if TRUE, the labels of the ROI assays will be visualized
`spatial`	the name of the main spatial system
`channel`	the name of the channel associated with the image
`background.color`	the color of plot background if a channel is not specified, or the spatial coord system doesnt have an image.
`background`	(DEPRECATED) the background of the plot. Either an image name, see vrImageNames or a vector of length two with image name and a channel name, see vrImageChannelNames. Type "black" or "white" for black or white backgrounds. if NULL, the main image (vrMainSpatial) and main channel (vrMainChannel) will be in the background. Otherwise the background will be grey.
`reg`	TRUE if registered coordinates of the main image (vrMainSpatial) is requested
`crop`	whether to crop an image of a spot assay to the extend of spots
`combine.groups`	if TRUE, tile colors will reflect relative abundance of either of two groups, strictly for visualizing two groups when assay is a molecule typed and tiled (see `n.tile`).
`legend.pt.size`	the size of points at the legend
`legend.text.size`	the size of the text at the legend
`scale.image`	if TRUE, background image will be scaled down to a low resolution (width: 1000px)
`legend.loc`	the location of the legend, default is "right"
`common.legend`	whether to use a common legend for all plots, see ggarrange
`collapse.plots`	whether to combine all ggplots
`interactive`	if TRUE, run interactive plot
`shiny.options`	a list of shiny options (browser, host, port etc.) passed `options` arguement of shinyApp. For more information, see runApp

vrSpatialPoints *vrSpatialPoints*

Description

Get and set spatial entities.

Usage

```
vrSpatialPoints(object, ...)

## S4 method for signature 'VoltRon'
vrSpatialPoints(object, assay = NULL)

## S4 method for signature 'vrMetadata'
vrSpatialPoints(object, assay = NULL)

## S4 method for signature 'vrAssay'
vrSpatialPoints(object)

## S4 method for signature 'vrAssayV2'
vrSpatialPoints(object)

## S4 method for signature 'vrImage'
vrSpatialPoints(object)

## S4 method for signature 'vrSpatial'
vrSpatialPoints(object)

## S4 method for signature 'vrSample'
vrSpatialPoints(object)

## S4 method for signature 'vrBlock'
vrSpatialPoints(object)

## S4 method for signature 'vrLayer'
vrSpatialPoints(object)

## S4 replacement method for signature 'vrAssay'
vrSpatialPoints(object) <- value

## S4 replacement method for signature 'vrAssayV2'
vrSpatialPoints(object) <- value

## S4 replacement method for signature 'vrImage'
vrSpatialPoints(object) <- value
```

```
## S4 replacement method for signature 'vrSpatial'
vrSpatialPoints(object) <- value

vrSpatialPoints(object, ...) <- value
```

Arguments

object	a VoltRon, vrSample, vrLayer, vrAssay or vrSpatial object.
...	arguments passed to other methods.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.
value	names for spatial points

Examples

```
vrSpatialPoints(visium_data)
vrSpatialPoints(visium_data, assay = "Visium")
vrSpatialPoints(visium_data, assay = "Assay1")
```

---

vrViolinPlot *vrViolinPlot*

Description

vrViolinPlot

Usage

```
vrViolinPlot(
  object,
  features = NULL,
  assay = NULL,
  group.by = "Sample",
  norm = TRUE,
  pt.size = 0.5,
  plot.points = TRUE,
  ncol = 2,
  nrow = NULL
)
```

Arguments

object	a VoltRon object
features	a set of features to be visualized, either from vrFeatures of raw or normalized data or columns of the Metadata.
assay	assay name (exp: Assay1) or assay class (exp: Visium, Xenium), see SampleMetadata. if NULL, the default assay will be used, see vrMainAssay.

group.by	a column of metadata from Metadata used as grouping label for the spatial entities
norm	if TRUE, the normalized data is used
pt.size	point size
plot.points	if TRUE, measures are visualized as points as well.
ncol	column wise number of plots, for ggarrange
nrow	row wise number of plots, for ggarrange

---

warpImage *warpImage*

Description

Warping a query image given a homography image

Usage

```
warpImage(ref_image, query_image, mapping)
```

Arguments

ref_image	reference image
query_image	query image
mapping	a list of the homography matrices and TPS keypoints

---

warpSimpleITKImage *getRcppWarpImage*

Description

Warping a query image given a homography image

Usage

```
warpSimpleITKImage(ref_image, query_image, mapping)
```

Arguments

ref_image	reference image
query_image	query image
mapping	a list of the homography matrices and TPS keypoints

xenium_data *Example Xenium Data*

Description

This VoltRon object is used for testing and as an example

Usage

```
xenium_data
```

Format

An object of class `VoltRon` of length 1.

Source

Created to serve as an example.

Examples

```
data(xenium_data)
```

zarrcreateGroup *zarrcreateGroup*

Description

get information of an ImageArray object

Usage

```
zarrcreateGroup(store, name)
```

Arguments

`store`	the location of (zarr) store
`name`	name of the group

Index