

# Package ‘Giotto’

December 9, 2019

**Title** Spatial single-cell transcriptomics pipeline.

**Version** 0.1.4

**Description** Pipeline to process, analyze and visualize (spatial) single-cell expression data.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.1

**Depends** data.table (>= 1.12.2),  
ggplot2 (>= 3.1.1),  
base (>= 3.5.1),  
utils (>= 3.5.1),  
R (>= 3.5.1)

**Imports** Rtsne (>= 0.15),  
uwot (>= 0.0.0.9010),  
multinet (>= 3.0.2),  
FactoMineR (>= 1.34),  
factoextra (>= 1.0.5),  
cowplot (>= 0.9.4),  
grDevices,  
RColorBrewer (>= 1.1-2),  
jackstraw (>= 1.3),  
dbscan (>= 1.1-3),  
ggalluvial (>= 0.9.1),  
scales (>= 1.0.0),  
ComplexHeatmap (>= 1.20.0),  
qvalue (>= 2.14.1),  
lfa (>= 1.12.0),  
igraph (>= 1.2.4.1),  
plotly,  
reticulate,  
magrittr,  
limma,  
ggdendro,  
smfishHmrf,  
matrixStats (>= 0.55.0),  
IRanges,  
devtools,  
reshape2

**Suggests** knitr,  
 rmarkdown,  
 MAST,  
 scran (>= 1.10.1),  
 png,  
 tiff,  
 biomaRt

**biocViews**

**VignetteBuilder** knitr

**Remotes** lambdamoses/smfishhmr-f-r

## R topics documented:

addCellMetadata . . . . .	6
addCellStatistics . . . . .	7
addGeneMetadata . . . . .	8
addGeneStatistics . . . . .	9
addHMRF . . . . .	10
addNetworkLayout . . . . .	10
addStatistics . . . . .	11
adjustGiottoMatrix . . . . .	12
aes_string2 . . . . .	13
allCellCellcommunicationsScores . . . . .	13
all_plots_save_function . . . . .	14
annotateGiotto . . . . .	15
annotateSpatialNetwork . . . . .	16
annotate_spatlocs_with_spatgrid_2D . . . . .	17
annotate_spatlocs_with_spatgrid_3D . . . . .	17
average_gene_gene_expression_in_groups . . . . .	18
binGetSpatialGenes . . . . .	18
calculateHVG . . . . .	20
calculateMetaTable . . . . .	21
calculateMetaTableCells . . . . .	22
calculate_spatial_genes_python . . . . .	23
cellProximityBarplot . . . . .	24
cellProximityEnrichment . . . . .	25
cellProximityHeatmap . . . . .	26
cellProximityNetwork . . . . .	27
cellProximitySpatPlot . . . . .	28
cellProximitySpatPlot2D . . . . .	30
cellProximitySpatPlot3D . . . . .	32
cellProximityVisPlot . . . . .	34
cellProximityVisPlot_2D_ggplot . . . . .	36
cellProximityVisPlot_2D_plotly . . . . .	38
cellProximityVisPlot_3D_plotly . . . . .	39
changeGiottoInstructions . . . . .	41
clusterCells . . . . .	42
combineMetadata . . . . .	44
convertEnsemblToGeneSymbol . . . . .	45
createGiottoInstructions . . . . .	45

createGiottoObject . . . . .	46
createHeatmap_DT . . . . .	48
createNearestNetwork . . . . .	49
createSpatialEnrich . . . . .	51
createSpatialGrid . . . . .	52
createSpatialGrid_2D . . . . .	53
createSpatialGrid_3D . . . . .	54
createSpatialNetwork . . . . .	55
create_average_detection_DT . . . . .	56
create_average_DT . . . . .	56
create_cell_type_random_cell_IDs . . . . .	57
create_cluster_matrix . . . . .	58
create_dimObject . . . . .	58
decide_cluster_order . . . . .	59
detectSpatialPatterns . . . . .	60
dimCellPlot . . . . .	61
dimCellPlot2D . . . . .	63
dimGenePlot . . . . .	66
dimGenePlot2D . . . . .	68
dimGenePlot3D . . . . .	70
dimPlot . . . . .	72
dimPlot2D . . . . .	75
dimPlot2D_single . . . . .	78
dimPlot3D . . . . .	80
direction_test_CPG . . . . .	82
doHclust . . . . .	83
doHMRF . . . . .	84
doKmeans . . . . .	85
doLeidenCluster . . . . .	87
doLeidenSubCluster . . . . .	88
doLouvainCluster . . . . .	90
doLouvainCluster_community . . . . .	91
doLouvainCluster_multinet . . . . .	92
doLouvainSubCluster . . . . .	93
doLouvainSubCluster_community . . . . .	95
doLouvainSubCluster_multinet . . . . .	97
doRandomWalkCluster . . . . .	98
doSNNCluster . . . . .	100
dt_to_matrix . . . . .	101
exportGiottoViewer . . . . .	101
exprOnlyCellCellcommunicationScores . . . . .	102
extended_gini_fun . . . . .	103
extractNearestNetwork . . . . .	104
fDataDT . . . . .	104
filterCombinations . . . . .	105
filterCPGscores . . . . .	106
filterDistributions . . . . .	107
filterGiotto . . . . .	108
findGiniMarkers . . . . .	109
findGiniMarkers_one_vs_all . . . . .	110
findMarkers . . . . .	111
findMarkers_one_vs_all . . . . .	113

findMastMarkers	114
findMastMarkers_one_vs_all	115
findScranMarkers	116
findScranMarkers_one_vs_all	117
find_grid_2D	118
find_grid_3D	118
find_grid_x	118
find_grid_y	119
find_grid_z	119
fish_function	119
fish_function2	119
FSV_show	120
GenePattern_show	121
general_save_function	122
get10Xmatrix	123
getCellProximityGeneScores	123
getClusterSimilarity	125
getDendrogramSplits	126
getDistinctColors	127
getGeneToGeneScores	127
get_cell_to_cell_sorted_name_conversion	128
get_interaction_gene_enrichment	129
get_specific_interaction_gene_enrichment	129
ggplot_save_function	130
giotto-class	131
hyperGeometricEnrich	132
kmeans_binarize	133
loadHMRf	133
makeSignMatrixPAGE	134
makeSignMatrixRank	134
make_simulated_network	135
mergeClusters	135
mygini_fun	136
nnDT_to_kNN	137
node_clusters	137
normalizeGiotto	138
OR_function2	139
PAGEEnrich	139
pDataDT	140
plotCPGscores	141
plotGTGscores	142
plotHeatmap	143
plotly_axis_scale_2D	145
plotly_axis_scale_3D	146
plotly_grid	146
plotly_network	147
plotMetaDataCellsHeatmap	148
plotMetaDataHeatmap	150
plotPCA	152
plotPCA_2D	154
plotPCA_3D	156
plotTSNE	157

plotTSNE_2D . . . . .	159
plotTSNE_3D . . . . .	161
plotUMAP . . . . .	163
plotUMAP_2D . . . . .	165
plotUMAP_3D . . . . .	167
plot_network_layer_ggplot . . . . .	168
plot_point_layer_ggplot . . . . .	169
plot_spat_point_layer_ggplot . . . . .	171
print.giotto . . . . .	173
rankEnrich . . . . .	173
rank_binarize . . . . .	174
readGiottoInstructions . . . . .	174
removeCellAnnotation . . . . .	175
removeGeneAnnotation . . . . .	175
replaceGiottoInstructions . . . . .	176
runPCA . . . . .	176
runSNE . . . . .	177
runUMAP . . . . .	179
selectPatternGenes . . . . .	180
select_expression_values . . . . .	181
show,giotto-method . . . . .	181
showClusterDendrogram . . . . .	182
showClusterHeatmap . . . . .	183
showCPGscores . . . . .	184
showGeneExpressionProximityScore . . . . .	185
showGiottoInstructions . . . . .	186
showGTGscores . . . . .	186
showIntExpressionProximityScore . . . . .	187
showPattern . . . . .	188
showPattern2D . . . . .	189
showPattern3D . . . . .	190
showPatternGenes . . . . .	192
showProcessingSteps . . . . .	193
showTopGeneToGene . . . . .	193
signPCA . . . . .	194
spatCellPlot . . . . .	195
spatCellPlot2D . . . . .	198
spatDimCellPlot . . . . .	201
spatDimCellPlot2D . . . . .	205
spatDimGenePlot . . . . .	209
spatDimGenePlot2D . . . . .	211
spatDimGenePlot3D . . . . .	213
spatDimPlot . . . . .	216
spatDimPlot2D . . . . .	220
spatDimPlot3D . . . . .	224
spatGenePlot . . . . .	227
spatGenePlot2D . . . . .	229
spatGenePlot3D . . . . .	231
Spatial_AEH . . . . .	232
Spatial_DE . . . . .	233
spatPlot . . . . .	234
spatPlot2D . . . . .	237

spatPlot2D_single . . . . .	240
spatPlot3D . . . . .	243
specificCellCellcommunicationScores . . . . .	245
split_dendrogram_in_two . . . . .	246
stitchFieldCoordinates . . . . .	247
subClusterCells . . . . .	248
subsetGiotto . . . . .	250
subsetGiottoLocs . . . . .	250
viewHMRFresults . . . . .	251
viewHMRFresults2D . . . . .	252
viewHMRFresults3D . . . . .	253
violinPlot . . . . .	254
visDimGenePlot . . . . .	255
visDimGenePlot_2D_ggplot . . . . .	257
visDimGenePlot_3D_plotly . . . . .	258
visDimPlot . . . . .	260
visDimPlot_2D_ggplot . . . . .	262
visDimPlot_2D_plotly . . . . .	264
visDimPlot_3D_plotly . . . . .	266
visForceLayoutPlot . . . . .	267
visGenePlot . . . . .	269
visGenePlot_2D_ggplot . . . . .	271
visGenePlot_3D_plotly . . . . .	272
visPlot . . . . .	274
visPlot_2D_ggplot . . . . .	276
visPlot_2D_plotly . . . . .	279
visPlot_3D_plotly . . . . .	280
visSpatDimGenePlot . . . . .	282
visSpatDimGenePlot_2D . . . . .	285
visSpatDimGenePlot_3D . . . . .	287
visSpatDimPlot . . . . .	289
visSpatDimPlot_2D . . . . .	291
visSpatDimPlot_3D . . . . .	294
writeHMRFresults . . . . .	296
write_giotto_viewer_annotation . . . . .	296
write_giotto_viewer_dim_reduction . . . . .	297
write_giotto_viewer_numeric_annotation . . . . .	298

**Index****299**

addCellMetadata

*addCellMetadata***Description**

adds cell metadata to the giotto object

**Usage**

```
addCellMetadata(
  gobject,
  new_metadata,
  by_column = FALSE,
  column_cell_ID = NULL
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>new_metadata</code>	new cell metadata to use (data.table, data.frame, ...)
<code>by_column</code>	merge metadata based on cell_ID column in pDataDT (default = FALSE)
<code>column_cell_ID</code>	column name of new metadata to use if <code>by_column = TRUE</code>

**Details**

You can add additional cell metadata in two manners: 1. Provide a data.table or data.frame with cell annotations in the same order as the cell\_ID column in pDataDT(gobject) 2. Provide a data.table or data.frame with cell annotations and specify which column contains the cell IDs, these cell IDs need to match with the cell\_ID column in pDataDT(gobject)

**Value**

giotto object

**Examples**

```
addCellMetadata(gobject)
```

---

<code>addCellStatistics</code>	<i>addCellStatistics</i>
--------------------------------	--------------------------

---

**Description**

adds cells statistics to the giotto object

**Usage**

```
addCellStatistics(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  detection_threshold = 0,
  return_gobject = TRUE
)
```

**Arguments**

**gobject**                giotto object  
**expression\_values**        expression values to use  
**detection\_threshold**        detection threshold to consider a gene detected  
**return\_gobject**    boolean: return giotto object (default = TRUE)

**Details**

This function will add the following statistics to cell metadata:

- **nr\_genes**: Denotes in how many genes are detected per cell
- **perc\_genes**: Denotes what percentage of genes is detected per cell
- **total\_expr**: Shows the total sum of gene expression per cell

**Value**

giotto object if **return\_gobject** = TRUE

**Examples**

```
addCellStatistics(gobject)
```

---

addGeneMetadata	<i>addGeneMetadata</i>
-----------------	------------------------

---

**Description**

adds gene metadata to the giotto object

**Usage**

```
addGeneMetadata(gobject, new_metadata, by_column = F, column_gene_ID = NULL)
```

**Arguments**

**gobject**                giotto object  
**new\_metadata**        new metadata to use  
**by\_column**            merge metadata based on gene\_ID column in fDataDT  
**column\_cell\_ID**    column name of new metadata to use if **by\_column** = TRUE

**Details**

You can add additional gene metadata in two manners: 1. Provide a data.table or data.frame with gene annotations in the same order as the gene\_ID column in fDataDT(gobject) 2. Provide a data.table or data.frame with gene annotations and specify which column contains the gene IDs, these gene IDs need to match with the gene\_ID column in fDataDT(gobject)



**Value**

giotto object

**Examples**

```
addGeneMetadata(gobject)
```

---

addGeneStatistics	<i>addGeneStatistics</i>
-------------------	--------------------------

---

**Description**

adds gene statistics to the giotto object

**Usage**

```
addGeneStatistics(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  detection_threshold = 0,
  return_gobject = TRUE
)
```

**Arguments**

gobject            giotto object  
 expression\_values            expression values to use  
 detection\_threshold            detection threshold to consider a gene detected  
 return\_gobject    boolean: return giotto object (default = TRUE)

**Details**

This function will add the following statistics to gene metadata:

- nr\_cells: Denotes in how many cells the gene is detected
- per\_cells: Denotes in what percentage of cells the gene is detected
- total\_expr: Shows the total sum of gene expression in all cells
- mean\_expr: Average gene expression in all cells
- mean\_expr\_det: Average gene expression in cells with detectable levels of the gene

**Value**

giotto object if return\_gobject = TRUE

**Examples**

```
addGeneStatistics(gobject)
```

---

addHMRF

*addHMRF*


---

**Description**

Add selected results from doHMRF to the giotto object

**Usage**

```
addHMRF(gobject, HMRFoutput, k = NULL, betas_to_add = NULL, hmrf_name = NULL)
```

**Arguments**

gobject	giotto object
HMRFoutput	HMRF output from doHMRF()
k	number of domains
betas_to_add	results from different betas that you want to add
name	specify a custom name

**Details**

Description ...

**Value**

giotto object

**Examples**

```
addHMRF(gobject)
```

---

addNetworkLayout

*addNetworkLayout*


---

**Description**

Add a network layout for a selected nearest neighbor network

**Usage**

```
addNetworkLayout(
  gobject,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  layout_type = c("drl"),
  options_list = NULL,
  layout_name = "layout",
  return_gobject = TRUE
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>nn_network_to_use</code>	kNN or sNN
<code>network_name</code>	name of NN network to be used
<code>layout_type</code>	layout algorithm to use
<code>options_list</code>	list of options for selected layout
<code>layout_name</code>	name for layout
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)

**Details**

This function creates layout coordinates based on the provided kNN or sNN. Currently only the force-directed graph layout "drl", see [layout\\_with\\_drl](#), is implemented. This provides an alternative to tSNE or UMAP based visualizations.

**Value**

giotto object with updated layout for selected NN network

**Examples**

```
addNetworkLayout(gobject)
```

---

addStatistics

*addStatistics*


---

**Description**

adds genes and cells statistics to the giotto object

**Usage**

```
addStatistics(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  detection_threshold = 0,
  return_gobject = TRUE
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>detection_threshold</code>	detection threshold to consider a gene detected
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)

**Details**

See [addGeneStatistics](#) and [addCellStatistics](#)

**Value**

giotto object if return\_gobject = TRUE, else a list with results

**Examples**

```
addStatistics(gobject)
```

---

adjustGiottoMatrix	<i>adjustGiottoMatrix</i>
--------------------	---------------------------

---

**Description**

normalize and/or scale expression values of Giotto object

**Usage**

```
adjustGiottoMatrix(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  batch_columns = NULL,
  covariate_columns = NULL,
  return_gobject = TRUE,
  update_slot = c("custom")
)
```

**Arguments**

gobject	giotto object
expression_values	expression values to use
batch_columns	metadata columns that represent different batch (max = 2)
covariate_columns	metadata columns that represent covariates to regress out
return_gobject	boolean: return giotto object (default = TRUE)
update_slot	expression slot that will be updated (default = custom)

**Details**

This function implements the [limma::removeBatchEffect](#) function to remove known batch effects and to adjust expression values according to provided covariates.

**Value**

giotto object

**Examples**

```
adjustGiottoMatrix(gobject)
```

---

aes_string2	<i>aes_string2</i>
-------------	--------------------

---

**Description**

makes sure aes\_string can also be used with names that start with numeric values

**Usage**

```
aes_string2(...)
```

---

allCellCellcommunicationsScores	<i>allCellCellcommunicationsScores</i>
---------------------------------	--

---

**Description**

All Cell-Cell communication scores based on spatial expression of interacting cells

**Usage**

```
allCellCellcommunicationsScores(
  gobject,
  spatial_network_name = "spatial_network",
  cluster_column = "cell_types",
  random_iter = 100,
  gene_set_1,
  gene_set_2,
  log2FC_addendum = 0.1,
  min_observations = 2,
  verbose = c("a little", "a lot", "none")
)
```

**Arguments**

gobject	giotto object to use
spatial_network_name	spatial network to use for identifying interacting cells
cluster_column	cluster column with cell type information
random_iter	number of iterations
gene_set_1	first specific gene set from gene pairs
gene_set_2	second specific gene set from gene pairs
log2FC_addendum	addendum to add when calculating log2FC
min_observations	minimum number of interactions needed to be considered
verbose	verbose

**Details**

Details will follow.

**Value**

Cell-Cell communication scores for gene pairs based on spatial interaction

**Examples**

```
allCellCellcommunicationsScores(gobject)
```

---

```
all_plots_save_function
```

```
all_plots_save_function
```

---

**Description**

Function to automatically save plots to directory of interest

**Usage**

```
all_plots_save_function(  
  gobject,  
  plot_object,  
  save_dir = NULL,  
  save_folder = NULL,  
  save_name = NULL,  
  default_save_name = "giotto_plot",  
  save_format = NULL,  
  show_saved_plot = F,  
  ncol = 1,  
  nrow = 1,  
  scale = 1,  
  base_width = NULL,  
  base_height = NULL,  
  base_aspect_ratio = NULL,  
  units = NULL,  
  dpi = NULL,  
  limitsize = TRUE,  
  ...  
)
```

**Arguments**

gobject	giotto object
plot_object	object to plot
save_dir	directory to save to
save_folder	folder in save_dir to save to
save_name	name of plot

save_format	format (e.g. png, tiff, pdf, ...)
show_saved_plot	load & display the saved plot
ncol	number of columns
nrow	number of rows
scale	scale
base_width	width
base_height	height
base_aspect_ratio	aspect ratio
units	units
dpi	Plot resolution
limitsize	When TRUE (the default), ggsave will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.
...	additional parameters to ggplot_save_function or general_save_function

### See Also

[Giotto::general\\_save\\_function](#)

### Examples

```
all_plots_save_function(gobject)
```

---

annotateGiotto	<i>annotateGiotto</i>
----------------	-----------------------

---

### Description

Converts cluster results into provided annotation.

### Usage

```
annotateGiotto(
  gobject,
  annotation_vector = NULL,
  cluster_column = NULL,
  name = "cell_types"
)
```

### Arguments

gobject	giotto object
annotation_vector	named annotation vector (names = cluster ids)
cluster_column	cluster column to convert to annotation names
name	new name for annotation column

**Details**

You need to specify which (cluster) column you want to annotate and you need to provide an annotation vector like this:

- 1. identify the cell type of each cluster
- 2. create a vector of these cell types, e.g. `cell_types = c('T-cell', 'B-cell', 'Stromal')`
- 3. provide original cluster names to previous vector, e.g. `names(cell_types) = c(2, 1, 3)`

**Value**

giotto object

**Examples**

```
annotateGiotto(gobject)
```

---

```
annotateSpatialNetwork
```

```
annotateSpatialNetwork
```

---

**Description**

Annotate spatial network with cell metadata information.

**Usage**

```
annotateSpatialNetwork(
  gobject,
  spatial_network_name = "spatial_network",
  cluster_column
)
```

**Arguments**

```
gobject          giotto object
spatial_network_name
                  name of spatial network to use
cluster_column   name of column to use for clusters
```

**Value**

annotated network in data.table format

**Examples**

```
annotateSpatialNetwork(gobject)
```



---

```

annotate_spatlocs_with_spatgrid_2D
    annotate_spatlocs_with_spatgrid_2D

```

---

### Description

annotate spatial locations with 2D spatial grid information

### Usage

```
annotate_spatlocs_with_spatgrid_2D(spatloc, spatgrid)
```

### Arguments

spatloc	spatial_locs slot from giotto object
spatgrid	selected spatial_grid slot from giotto object

### Value

annotated spatial location data.table

### Examples

```
annotate_spatlocs_with_spatgrid_2D()
```

---

```

annotate_spatlocs_with_spatgrid_3D
    annotate_spatlocs_with_spatgrid_3D

```

---

### Description

annotate spatial locations with 3D spatial grid information

### Usage

```
annotate_spatlocs_with_spatgrid_3D(spatloc, spatgrid)
```

### Arguments

spatloc	spatial_locs slot from giotto object
spatgrid	selected spatial_grid slot from giotto object

### Value

annotated spatial location data.table

### Examples

```
annotate_spatlocs_with_spatgrid_3D()
```

---

```
average_gene_gene_expression_in_groups
      average_gene_gene_expression_in_groups
```

---

### Description

calculate average expression per cluster

### Usage

```
average_gene_gene_expression_in_groups(
  gobject,
  cluster_column = "cell_types",
  gene_set_1,
  gene_set_2
)
```

### Arguments

<code>gobject</code>	giotto object to use
<code>cluster_column</code>	cluster column with cell type information
<code>gene_set_1</code>	first specific gene set from gene pairs
<code>gene_set_2</code>	second specific gene set from gene pairs

### Details

Details will follow.

### Value

data.table with average expression scores for each cluster

### Examples

```
average_gene_gene_expression_in_groups(gobject)
```

---

```
binGetSpatialGenes      binGetSpatialGenes
```

---

### Description

Rapid computation of genes that are spatially clustered

**Usage**

```
binGetSpatialGenes(
  gobject,
  bin_method = c("kmeans", "rank"),
  expression_values = c("normalized", "scaled", "custom"),
  subset_genes = NULL,
  spatial_network_name = "spatial_network",
  nstart = 3,
  iter_max = 10,
  percentage_rank = 10,
  do_fisher_test = F,
  community_expectation = 5,
  verbose = F
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>bin_method</code>	method to binarize gene expression
<code>expression_values</code>	expression values to use
<code>subset_genes</code>	only select a subset of genes to test
<code>spatial_network_name</code>	name of spatial network to use (default = 'spatial_network')
<code>nstart</code>	kmeans: nstart parameter
<code>iter_max</code>	kmeans: iter.max parameter
<code>do_fisher_test</code>	perform fisher test
<code>community_expectation</code>	cell degree expectation in spatial communities
<code>verbose</code>	be verbose
<code>rank_percentage</code>	percentage of top cells for binarization

**Details**

We provide two ways to identify spatial genes based on gene expression binarization. Both methods are identical except for how binarization is performed.

- 1. binarize: Each gene is binarized (0 or 1) in each cell with **kmeans** (k = 2) or based on **rank** percentile
- 2. network: All cells are connected through a k-nearest neighbor network
- 3. contingency table: A contingency table is calculated based on all pairwise cell-cell interactions (0-0, 0-1, 1-0 or 1-1)
- 4. For each gene an odds-ratio (OR) and fisher.test (optional) is calculated

Additionally 2 other statistics are provided:

- Number of cells with high expression (binary = 1)
- total and ratio of highly connected cells: Cells with a connectivity higher than community\_expectation

By selecting a subset of likely spatial genes (e.g. highly variable genes) the function will be much faster.

Value

data.table with results (see details)

Examples

```
binGetSpatialGenes(gobject)
```

calculateHVG	<i>calculateHVG</i>
--------------	---------------------

Description

compute highly variable genes

Usage

```
calculateHVG(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  method = c("cov_groups", "cov_loess"),  
  reverse_log_scale = FALSE,  
  logbase = 2,  
  expression_threshold = 0,  
  nr_expression_groups = 20,  
  zscore_threshold = 1.5,  
  HVGname = "hvg",  
  difference_in_cov = 0.1,  
  show_plot = NA,  
  return_plot = NA,  
  save_plot = NA,  
  save_param = list(),  
  default_save_name = "HVGplot",  
  return_gobject = TRUE  
)
```

Arguments

- gobject            giotto object
- expression\_values            expression values to use
- method            method to calculate highly variable genes
- reverse\_log\_scale            reverse log-scale of expression values (default = FALSE)
- logbase            if reverse\_log\_scale is TRUE, which log base was used?
- expression\_threshold            expression threshold to consider a gene detected
- nr\_expression\_groups            number of expression groups for cov\_groups
- zscore\_threshold            zscore to select hvg for cov\_groups

HVGname            name for highly variable genes in cell metadata  
 difference\_in\_cov            minimum difference in coefficient of variance required  
 show\_plot            show plot  
 return\_plot            return ggplot object  
 save\_plot            directly save the plot [boolean]  
 save\_param            list of saving parameters from all\_plots\_save\_function()  
 default\_save\_name            default save name for saving, don't change, change save\_name in save\_param  
 return\_gobject    boolean: return giotto object (default = TRUE)

## Details

Currently we provide 2 ways to calculate highly variable genes: **1. high coeff of variance (COV) within groups:**

First genes are binned (*nr\_expression\_groups*) into average expression groups and the COV for each gene is converted into a z-score within each bin. Genes with a z-score higher than the threshold (*zscore\_threshold*) are considered highly variable.

## 2. high COV based on loess regression prediction:

A predicted COV is calculated for each gene using loess regression ( $COV \sim \log(\text{mean expression})$ ). Genes that show a higher than predicted COV (*difference\_in\_cov*) are considered highly variable.

## Value

giotto object highly variable genes appended to gene metadata (fDataDT)

## Examples

```
calculateHVG(gobject)
```

---

calculateMetaTable	<i>calculateMetaTable</i>
--------------------	---------------------------

---

## Description

calculates the average gene expression for one or more (combined) annotation columns.

## Usage

```
calculateMetaTable(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  metadata_cols = NULL,  
  selected_genes = NULL  
)
```

**Arguments**

gobject            giotto object  
 expression\_values      expression values to use  
 metadata\_cols    annotation columns found in pDataDT(gobject)  
 selected\_genes   subset of genes to use

**Value**

data.table with average expression values for each gene per (combined) annotation

**Examples**

```
calculateMetaTable(gobject)
```

---

```
calculateMetaTableCells
```

*calculateMetaTableCells*

---

**Description**

calculates the average metadata values for one or more (combined) annotation columns.

**Usage**

```
calculateMetaTableCells(
  gobject,
  value_cols = NULL,
  metadata_cols = NULL,
  spat_enr_names = NULL
)
```

**Arguments**

gobject            giotto object  
 value\_cols        metadata or enrichment value columns to use  
 metadata\_cols    annotation columns found in pDataDT(gobject)  
 spat\_enr\_names   which spatial enrichment results to include

**Value**

data.table with average metadata values per (combined) annotation

**Examples**

```
calculateMetaTableCells(gobject)
```

---

```
calculate_spatial_genes_python  
    calculate_spatial_genes_python
```

---

## Description

Calculate spatial genes using distance matrix.

## Usage

```
calculate_spatial_genes_python(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  metric = "euclidean",  
  subset_genes = NULL,  
  rbp_p = 0.95,  
  examine_top = 0.3,  
  python_path = NULL  
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>metric</code>	distance metric to use
<code>subset_genes</code>	only run on this subset of genes
<code>rbp_p</code>	fractional binarization threshold
<code>examine_top</code>	top fraction to evaluate with silhouette
<code>python_path</code>	specify specific path to python if required

## Details

Description of how we compute spatial pattern genes.

## Value

data.table with spatial scores

## Examples

```
calculate_spatial_genes_python(gobject)
```

---

cellProximityBarplot    *cellProximityBarplot*

---

## Description

Create barplot from cell-cell proximity scores

## Usage

```
cellProximityBarplot(
  gobject,
  CPscore,
  min_orig_ints = 5,
  min_sim_ints = 5,
  p_val = 0.05,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "cellProximityBarplot"
)
```

## Arguments

<code>gobject</code>	giotto object
<code>CPscore</code>	CPscore, output from <code>cellProximityEnrichment()</code>
<code>min_orig_ints</code>	filter on minimum original cell-cell interactions
<code>min_sim_ints</code>	filter on minimum simulated cell-cell interactions
<code>p_val</code>	p-value
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>

## Details

This function creates a barplot that shows the spatial proximity enrichment or depletion of cell type pairs.

## Value

ggplot barplot

## Examples

```
cellProximityBarplot(CPscore)
```



---

```
cellProximityEnrichment  
  cellProximityEnrichment
```

---

## Description

Compute cell-cell interaction enrichment (observed vs expected)

## Usage

```
cellProximityEnrichment(  
  gobject,  
  spatial_network_name = "spatial_network",  
  cluster_column,  
  number_of_simulations = 100  
)
```

## Arguments

<code>gobject</code>	giotto object
<code>spatial_network_name</code>	name of spatial network to use
<code>cluster_column</code>	name of column to use for clusters
<code>number_of_simulations</code>	number of simulations to create expected observations

## Details

Spatial proximity enrichment or depletion between pairs of cell types is calculated by calculating the observed over the expected frequency of cell-cell proximity interactions. The expected frequency is the average frequency calculated from a number of spatial network simulations. Each individual simulation is obtained by reshuffling the cell type labels of each node (cell) in the spatial network.

## Value

List of cell Proximity scores (CPscores) in data.table format. The first data.table (`raw_sim_table`) shows the raw observations of both the original and simulated networks. The second data.table (`enrichm_res`) shows the enrichment results.

## Examples

```
cellProximityEnrichment(gobject)
```

---

cellProximityHeatmap    *cellProximityHeatmap*


---

## Description

Create heatmap from cell-cell proximity scores

## Usage

```
cellProximityHeatmap(
  gobject,
  CPscore,
  scale = T,
  order_cell_types = T,
  color_breaks = NULL,
  color_names = NULL,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "cellProximityHeatmap"
)
```

## Arguments

gobject	giotto object
CPscore	CPscore, output from cellProximityEnrichment()
scale	scale cell-cell proximity interaction scores
order_cell_types	order cell types based on enrichment correlation
color_breaks	numerical vector of length 3 to represent min, mean and maximum
color_names	character color vector of length 3
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

This function creates a heatmap that shows the spatial proximity enrichment or depletion of cell type pairs.

## Value

ggplot heatmap

**Examples**

```
cellProximityHeatmap(CPscore)
```

---

```
cellProximityNetwork    cellProximityNetwork
```

---

**Description**

Create network from cell-cell proximity scores

**Usage**

```
cellProximityNetwork(
  gobject,
  CPscore,
  remove_self_edges = FALSE,
  self_loop_strength = 0.1,
  color_depletion = "lightgreen",
  color_enrichment = "red",
  rescale_edge_weights = TRUE,
  edge_weight_range_depletion = c(0.1, 1),
  edge_weight_range_enrichment = c(1, 5),
  layout = "Fruchterman",
  only_show_enrichment_edges = F,
  edge_width_range = c(0.1, 2),
  node_size = 4,
  node_text_size = 6,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "cellProximityNetwork"
)
```

**Arguments**

gobject	giotto object
CPscore	CPscore, output from cellProximityEnrichment()
remove_self_edges	remove enrichment/depletion edges with itself
self_loop_strength	size of self-loops
color_depletion	color for depleted cell-cell interactions
color_enrichment	color for enriched cell-cell interactions
rescale_edge_weights	rescale edge weights (boolean)
edge_weight_range_depletion	numerical vector of length 2 to rescale depleted edge weights

edge_weight_range_enrichment	numerical vector of length 2 to rescale enriched edge weights
layout	layout algorithm to use to draw nodes and edges
only_show_enrichment_edges	show only the enriched pairwise scores
edge_width_range	range of edge width
node_size	size of nodes
node_text_size	size of node labels
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

### Details

This function creates a network that shows the spatial proximity enrichment or depletion of cell type pairs.

### Value

igraph plot

### Examples

```
cellProximityNetwork(CPscore)
```

---

cellProximitySpatPlot *cellProximitySpatPlot*

---

### Description

Visualize 2D cell-cell interactions according to spatial coordinates in ggplot mode

### Usage

```
cellProximitySpatPlot(gobject, ...)
```

### Arguments

gobject	giotto object
interaction_name	cell-cell interaction name
cluster_column	cluster column with cell clusters
sdimx	x-axis dimension name (default = 'sdimx')
sdimy	y-axis dimension name (default = 'sdimy')

cell_color	color for cells (see details)
cell_color_code	named vector with colors
color_as_factor	convert color column to factor
show_other_cells	decide if show cells not in network
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
coord_fix_ratio	fix ratio between x and y-axis
show_legend	show legend
point_size_select	size of selected points
point_select_border_col	border color of selected points
point_select_border_stroke	stroke size of selected points
point_size_other	size of other points
point_other_border_col	border color of other points
point_other_border_stroke	stroke size of other points
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## See Also

[cellProximitySpatPlot2D](#) and [cellProximitySpatPlot3D](#) for 3D

**Examples**

```
cellProximitySpatPlot(gobject)
```

---

```
cellProximitySpatPlot2D
```

```
cellProximitySpatPlot2D
```

---

**Description**

Visualize 2D cell-cell interactions according to spatial coordinates in ggplot mode

**Usage**

```
cellProximitySpatPlot2D(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = "sdimx",
  sdimy = "sdimy",
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  show_other_cells = F,
  show_network = F,
  show_other_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  coord_fix_ratio = 1,
  show_legend = T,
  point_size_select = 2,
  point_select_border_col = "black",
  point_select_border_stroke = 0.05,
  point_size_other = 1,
  point_alpha_other = 0.3,
  point_other_border_col = "lightgrey",
  point_other_border_stroke = 0.01,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "cellProximitySpatPlot2D"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name

cluster_column	cluster column with cell clusters
sdimx	x-axis dimension name (default = 'sdimx')
sdimy	y-axis dimension name (default = 'sdimy')
cell_color	color for cells (see details)
cell_color_code	named vector with colors
color_as_factor	convert color column to factor
show_other_cells	decide if show cells not in network
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
coord_fix_ratio	fix ratio between x and y-axis
show_legend	show legend
point_size_select	size of selected points
point_select_border_col	border color of selected points
point_select_border_stroke	stroke size of selected points
point_size_other	size of other points
point_other_border_col	border color of other points
point_other_border_stroke	stroke size of other points
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

**Examples**

```
cellProximitySpatPlot2D(gobject)
```

---

```
cellProximitySpatPlot3D
```

```
cellProximitySpatPlot2D
```

---

**Description**

Visualize 3D cell-cell interactions according to spatial coordinates in plotly mode

**Usage**

```
cellProximitySpatPlot3D(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = "sdimx",
  sdimy = "sdimy",
  sdimz = "sdimz",
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  show_other_cells = T,
  show_network = T,
  show_other_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  show_legend = T,
  point_size_select = 4,
  point_size_other = 2,
  point_alpha_other = 0.5,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "cellProximitySpatPlot3D",
  ...
)
```



**Arguments**

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name
<code>cluster_column</code>	cluster column with cell clusters
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor
<code>show_other_cells</code>	decide if show cells not in network
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>show_legend</code>	show legend
<code>point_size_select</code>	size of selected points
<code>point_size_other</code>	size of other points
<code>show_plot</code>	show plots
<code>return_plot</code>	return plotly object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>

**Details**

Description of parameters.

**Value**

plotly

**Examples**

```
cellProximitySpatPlot3D(gobject)
```

---

cellProximityVisPlot	<i>cellProximityVisPlot</i>
----------------------	-----------------------------

---

## Description

Visualize cell-cell interactions according to spatial coordinates

## Usage

```
cellProximityVisPlot(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = NULL,
  sdimy = NULL,
  sdimz = NULL,
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  show_other_cells = F,
  show_network = F,
  show_other_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  coord_fix_ratio = 1,
  show_legend = T,
  point_size_select = 2,
  point_select_border_col = "black",
  point_select_border_stroke = 0.05,
  point_size_other = 1,
  point_alpha_other = 0.3,
  point_other_border_col = "lightgrey",
  point_other_border_stroke = 0.01,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  plot_method = c("ggplot", "plotly"),
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name
<code>cluster_column</code>	cluster column with cell clusters

<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>coord_fix_ratio</code>	fix ratio between x and y-axis
<code>show_legend</code>	show legend
<code>point_size_select</code>	size of selected points
<code>point_select_border_col</code>	border color of selected points
<code>point_select_border_stroke</code>	stroke size of selected points
<code>point_size_other</code>	size of other points
<code>point_other_border_col</code>	border color of other points
<code>point_other_border_stroke</code>	stroke size of other points

**Details**

Description of parameters.

**Value**

ggplot or plotly

**Examples**

```
cellProximityVisPlot(gobject)
```

---

```
cellProximityVisPlot_2D_ggplot
      cellProximityVisPlot_2D_ggplot
```

---

## Description

Visualize 2D cell-cell interactions according to spatial coordinates in ggplot mode

## Usage

```
cellProximityVisPlot_2D_ggplot(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = NULL,
  sdimy = NULL,
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  show_other_cells = F,
  show_network = F,
  show_other_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  coord_fix_ratio = 1,
  show_legend = T,
  point_size_select = 2,
  point_select_border_col = "black",
  point_select_border_stroke = 0.05,
  point_size_other = 1,
  point_alpha_other = 0.3,
  point_other_border_col = "lightgrey",
  point_other_border_stroke = 0.01,
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name
<code>cluster_column</code>	cluster column with cell clusters
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors

color_as_factor	convert color column to factor
show_other_cells	decide if show cells not in network
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
coord_fix_ratio	fix ratio between x and y-axis
show_legend	show legend
point_size_select	size of selected points
point_select_border_col	border color of selected points
point_select_border_stroke	stroke size of selected points
point_size_other	size of other points
point_other_border_col	border color of other points
point_other_border_stroke	stroke size of other points

## Details

Description of parameters.

## Value

ggplot

## Examples

```
cellProximityVisPlot_2D_ggplot(gobject)
```

---

```
cellProximityVisPlot_2D_plotly
      cellProximityVisPlot_2D_plotly
```

---

## Description

Visualize 2D cell-cell interactions according to spatial coordinates in plotly mode

## Usage

```
cellProximityVisPlot_2D_plotly(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = NULL,
  sdimy = NULL,
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  show_other_cells = F,
  show_network = F,
  show_other_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  show_legend = T,
  point_size_select = 2,
  point_size_other = 1,
  point_alpha_other = 0.3,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name
<code>cluster_column</code>	cluster column with cell clusters
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors

color_as_factor	convert color column to factor
show_other_cells	decide if show cells not in network
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
show_legend	show legend
point_size_select	size of selected points
coord_fix_ratio	fix ratio between x and y-axis

**Details**

Description of parameters.

**Value**

plotly

**Examples**

```
cellProximityVisPlot_2D_plotly(gobject)
```

---

```
cellProximityVisPlot_3D_plotly
  cellProximityVisPlot_3D_plotly
```

---

**Description**

Visualize 3D cell-cell interactions according to spatial coordinates in plotly mode

**Usage**

```
cellProximityVisPlot_3D_plotly(
  gobject,
  interaction_name = NULL,
  cluster_column = NULL,
  sdimx = NULL,
  sdimy = NULL,
  sdimz = NULL,
  cell_color = NULL,
  cell_color_code = NULL,
```

```

    color_as_factor = T,
    show_other_cells = F,
    show_network = F,
    show_other_network = F,
    network_color = NULL,
    spatial_network_name = "spatial_network",
    show_grid = F,
    grid_color = NULL,
    spatial_grid_name = "spatial_grid",
    show_legend = T,
    point_size_select = 2,
    point_size_other = 1,
    point_alpha_other = 0.5,
    axis_scale = c("cube", "real", "custom"),
    custom_ratio = NULL,
    x_ticks = NULL,
    y_ticks = NULL,
    z_ticks = NULL,
    ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>interaction_name</code>	cell-cell interaction name
<code>cluster_column</code>	cluster column with cell clusters
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor
<code>show_other_cells</code>	decide if show cells not in network
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>show_legend</code>	show legend
<code>point_size_select</code>	size of selected points
<code>coord_fix_ratio</code>	fix ratio between x and y-axis



**Details**

Description of parameters.

**Value**

plotly

**Examples**

```
cellProximityVisPlot_3D_plotly(gobject)
```

---

changeGiottoInstructions  
*changeGiottoInstructions*

---

**Description**

Function to change one or more instructions from giotto object

**Usage**

```
changeGiottoInstructions(  
  gobject,  
  params = NULL,  
  new_values = NULL,  
  return_gobject = TRUE  
)
```

**Arguments**

gobject	giotto object
params	parameter(s) to change
new_values	new value(s) for parameter(s)
return_gobject	(boolean) return giotto object

**Value**

named vector with giotto instructions

**Examples**

```
changeGiottoInstructions()
```

clusterCells

*clusterCells***Description**

cluster cells using a variety of different methods

**Usage**

```
clusterCells(
  gobject,
  cluster_method = c("leiden", "louvain_community", "louvain_multinet", "randomwalk",
    "sNNclust", "kmeans", "hierarchical"),
  name = "cluster_name",
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  pyth_leid_resolution = 1,
  pyth_leid_weight_col = "weight",
  pyth_leid_part_type = c("RBConfigurationVertexPartition", "ModularityVertexPartition"),
  pyth_leid_init_memb = NULL,
  pyth_leid_iterations = 1000,
  pyth_louv_resolution = 1,
  pyth_louv_weight_col = NULL,
  python_louv_random = F,
  python_path = NULL,
  louvain_gamma = 1,
  louvain_omega = 1,
  walk_steps = 4,
  walk_clusters = 10,
  walk_weights = NA,
  sNNclust_k = 20,
  sNNclust_eps = 4,
  sNNclust_minPts = 16,
  borderPoints = TRUE,
  expression_values = c("normalized", "scaled", "custom"),
  genes_to_use = NULL,
  dim_reduction_to_use = c("cells", "pca", "umap", "tsne"),
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  distance_method = c("original", "pearson", "spearman", "euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski"),
  km_centers = 10,
  km_iter_max = 100,
  km_nstart = 1000,
  km_algorithm = "Hartigan-Wong",
  hc_agglomeration_method = c("ward.D2", "ward.D", "single", "complete", "average",
    "mcquitty", "median", "centroid"),
  hc_k = 10,
  hc_h = NULL,
  return_gobject = TRUE,
  set_seed = T,
```

```

    seed_number = 1234
)

```

### Arguments

gobject	giotto object
cluster_method	community cluster method to use
name	name for new clustering result
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use
pyth_leid_resolution	resolution for leiden
pyth_leid_weight_col	column to use for weights
pyth_leid_part_type	partition type to use
pyth_leid_init_memb	initial membership
pyth_leid_iterations	number of iterations
pyth_louv_resolution	resolution for louvain
pyth_louv_weight_col	python louvain param: weight column
python_louv_random	python louvain param: random
python_path	specify specific path to python if required
louvain_gamma	louvain param: gamma or resolution
louvain_omega	louvain param: omega
walk_steps	randomwalk: number of steps
walk_clusters	randomwalk: number of clusters
walk_weights	randomwalk: weight column
sNNclust_k	SNNclust: k neighbors to use
sNNclust_eps	SNNclust: epsilon
sNNclust_minPts	SNNclust: min points
borderPoints	SNNclust: border points
expression_values	expression values to use
genes_to_use	= NULL,
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	name of reduction 'pca',
dimensions_to_use	dimensions to use

distance_method	distance method
km_centers	kmeans centers
km_iter_max	kmeans iterations
km_nstart	kmeans random starting points
km_algorithm	kmeans algorithm
hc_agglomeration_method	hierarchical clustering method
hc_k	hierachical number of clusters
hc_h	hierarchical tree cutoff
return_gobject	boolean: return giotto object (default = TRUE)
set_seed	set seed
seed_number	number for seed

### Details

Wrapper for the different clustering methods.

### Value

giotto object with new clusters appended to cell metadata

### See Also

[doLeidenCluster](#), [doLouvainCluster\\_community](#), [doLouvainCluster\\_multinet](#), [doLouvainCluster](#), [doRandomWalkCluster](#), [doSNNCluster](#), [doKmeans](#), [doHclust](#)

### Examples

```
clusterCells(gobject)
```

---

combineMetadata

*combineMetadata*

---

### Description

This function combines the cell metadata with spatial locations and enrichment results from createSpatialEnrich

### Usage

```
combineMetadata(gobject, spat_enr_names = NULL)
```

### Arguments

gobject	Giotto object
spat_enr_names	names of spatial enrichment results to include

**Value**

Extended cell metadata in data.table format.

**Examples**

```
combineMetadata(gobject)
```

---

```
convertEnsemblToGeneSymbol  
  convertEnsemblToGeneSymbol
```

---

**Description**

This function convert ensembl gene IDs from a matrix to official gene symbols

**Usage**

```
convertEnsemblToGeneSymbol(matrix, species = c("mouse", "human"))
```

**Arguments**

matrix	an expression matrix with ensembl gene IDs as rownames
species	species to use for gene symbol conversion

**Details**

This function requires that the biomaRt library is installed

**Value**

expression matrix with gene symbols as rownames

**Examples**

```
convertEnsemblToGeneSymbol(matrix)
```

---

```
createGiottoInstructions  
  createGiottoInstructions
```

---

**Description**

Function to set global instructions for giotto functions

**Usage**

```
createGiottoInstructions(
  python_path = NULL,
  show_plot = NULL,
  return_plot = NULL,
  save_plot = NULL,
  save_dir = NULL,
  plot_format = NULL,
  dpi = NULL,
  units = NULL,
  height = NULL,
  width = NULL
)
```

**Arguments**

python_path	path to python binary to use
show_plot	print plot to console, default = TRUE
return_plot	return plot as object, default = TRUE
save_plot	automatically save plot, default = FALSE
save_dir	path to directory where to save plots
dpi	resolution for raster images
height	height of plots
width	width of plots

**Value**

named vector with giotto instructions

**Examples**

```
createGiottoInstructions()
```

---

createGiottoObject	<i>create Giotto object</i>
--------------------	-----------------------------

---

**Description**

Function to create a giotto object

**Usage**

```
createGiottoObject(
  raw_exprs,
  spatial_locs = NULL,
  norm_expr = NULL,
  norm_scaled_expr = NULL,
  custom_expr = NULL,
  cell_metadata = NULL,
```

```

    gene_metadata = NULL,
    spatial_network = NULL,
    spatial_network_name = NULL,
    spatial_grid = NULL,
    spatial_grid_name = NULL,
    spatial_enrichment = NULL,
    spatial_enrichment_name = NULL,
    dimension_reduction = NULL,
    nn_network = NULL,
    offset_file = NULL,
    instructions = NULL
)

```

### Arguments

raw_exprs	matrix with raw expression counts [required]
spatial_locs	data.table or data.frame with coordinates for cell centroids
norm_expr	normalized expression values
norm_scaled_expr	scaled expression values
custom_expr	custom expression values
cell_metadata	cell annotation metadata
gene_metadata	gene annotation metadata
spatial_network	list of spatial network(s)
spatial_network_name	list of spatial network name(s)
spatial_grid	list of spatial grid(s)
spatial_grid_name	list of spatial grid name(s)
spatial_enrichment	list of spatial enrichment score(s) for each spatial region
spatial_enrichment_name	list of spatial enrichment name(s)
dimension_reduction	list of dimension reduction(s)
nn_network	list of nearest neighbor network(s)
offset_file	file used to stitch fields together (optional)
instructions	list of instructions or output result from createGiottoInstructions

### Details

**[Requirements]** To create a giotto object you need to provide at least a matrix with genes as row names and cells as column names. To include spatial information about cells (or regions) you need to provide a data.table or data.frame with coordinates for all spatial dimensions. This can be 2D (x and y) or 3D (x, y, x). The row order for the cell coordinates should be the same as the column order for the provided expression data.

**[Instructions]** Additionally an instruction file, generated manually or with [createGiottoInstructions](#) can be provided to instructions, if not a default instruction file will be created for the Giotto object.

**[Multiple fields]** In case a dataset consists of multiple fields, like seqFISH+ for example, an offset file can be provided to stitch the different fields together. [stitchFieldCoordinates](#) can be used to generate such an offset file.

**[Processed data]** Processed count data, such as normalized data, can be provided using one of the different expression slots (norm\_expr, norm\_scaled\_expr, custom\_expr).

**[Metadata]** Cell and gene metadata can be provided using the cell and gene metadata slots. This data can also be added afterwards using the [addGeneMetadata](#) or [addCellMetadata](#) functions.

**[Other information]** Additional information can be provided through the appropriate slots:

- spatial networks
- spatial girds
- spatial enrichments
- dimensions reductions
- nearest neighbours networks

### Value

giotto object

### Examples

```
createGiottoObject(raw_exprs, spatial_locs)
```

---

createHeatmap\_DT

*createHeatmap\_DT*

---

### Description

creates order for clusters

### Usage

```
createHeatmap_DT(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  cluster_column = NULL,
  cluster_order = c("size", "correlation", "custom"),
  cluster_custom_order = NULL,
  cluster_cor_method = "pearson",
  cluster_hclust_method = "ward.D",
  gene_order = c("custom", "correlation"),
  gene_custom_order = NULL,
  gene_cor_method = "pearson",
  gene_hclust_method = "complete"
)
```



**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>genes</code>	genes to use
<code>cluster_column</code>	name of column to use for clusters
<code>cluster_order</code>	method to determine cluster order
<code>cluster_custom_order</code>	custom order for clusters
<code>cluster_cor_method</code>	method for cluster correlation
<code>cluster_hclust_method</code>	method for hierarchical clustering of clusters
<code>gene_order</code>	method to determine gene order
<code>gene_custom_order</code>	custom order for genes
<code>gene_cor_method</code>	method for gene correlation
<code>gene_hclust_method</code>	method for hierarchical clustering of genes

**Details**

Creates input data.tables for plotHeatmap function.

**Value**

list

**Examples**

```
createHeatmap_DT(gobject)
```

---

<code>createNearestNetwork</code>	<i>createNearestNetwork</i>
-----------------------------------	-----------------------------

---

**Description**

create a nearest neighbour (NN) network

**Usage**

```
createNearestNetwork(
  gobject,
  type = c("sNN", "kNN"),
  dim_reduction_to_use = "pca",
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  genes_to_use = NULL,
```

```

expression_values = c("normalized", "scaled", "custom"),
name = "sNN.pca",
return_gobject = TRUE,
k = 30,
minimum_shared = 5,
top_shared = 3,
verbose = T,
...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>type</code>	sNN or kNN
<code>dim_reduction_to_use</code>	dimension reduction method to use
<code>dim_reduction_name</code>	name of dimension reduction set to use
<code>dimensions_to_use</code>	number of dimensions to use as input
<code>genes_to_use</code>	if <code>dim_reduction_to_use = NULL</code> , which genes to use
<code>expression_values</code>	expression values to use
<code>name</code>	arbitrary name for NN network
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>k</code>	number of k neighbors to use
<code>minimum_shared</code>	minimum shared neighbors
<code>top_shared</code>	keep at ...
<code>verbose</code>	be verbose
<code>...</code>	additional parameters for kNN and sNN functions from dbscan

### Details

This function creates a k-nearest neighbour (kNN) or shared nearest neighbour (sNN) network based on the provided dimension reduction space. To run it directly on the gene expression matrix set `dim_reduction_to_use = NULL`.

See also [kNN](#) and [sNN](#) for more information about how the networks are created.

Output for kNN:

- from: cell\_ID for source cell
- to: cell\_ID for target cell
- distance: distance between cells
- weight: weight =  $1/(1 + \text{distance})$

Output for sNN:

- from: cell\_ID for source cell
- to: cell\_ID for target cell
- distance: distance between cells

- weight:  $1/(1 + \text{distance})$
- shared: number of shared neighbours
- rank: ranking of pairwise cell neighbours

For sNN networks two additional parameters can be set:

- minimum\_shared: minimum number of shared neighbours needed
- top\_shared: keep this number of the top shared neighbours, irrespective of minimum\_shared setting

## Value

giotto object with updated NN network

## Examples

```
createNearestNetwork(gobject)
```

---

createSpatialEnrich	<i>createSpatialEnrich</i>
---------------------	----------------------------

---

## Description

Function to calculate gene signature enrichment scores per spatial position using a hypergeometric test.

## Usage

```
createSpatialEnrich(
  gobject,
  enrich_method = c("PAGE", "rank", "hypergeometric"),
  sign_matrix,
  expression_values = c("normalized", "scaled", "custom"),
  reverse_log_scale = TRUE,
  logbase = 2,
  output_enrichment = c("original", "zscore"),
  name = "PAGE",
  return_gobject = TRUE
)
```

## Arguments

gobject	Giotto object
enrich_method	method for gene signature enrichment calculation
sign_matrix	Matrix of signature genes for each cell type / process
expression_values	expression values to use
reverse_log_scale	reverse expression values from log scale
logbase	log base to use if reverse_log_scale = TRUE

output\_enrichment      how to return enrichment output  
 name                    to give to spatial enrichment results, default = PAGE  
 return\_gobject    return giotto object

### Details

For details see the individual functions:

- PAGE: [PAGEEnrich](#)
- PAGE: [rankEnrich](#)
- PAGE: [hyperGeometricEnrich](#)

### Value

Giotto object or enrichment results if return\_gobject = FALSE

### Examples

```
createSpatialEnrich(gobject)
```

---

createSpatialGrid	<i>createSpatialGrid</i>
-------------------	--------------------------

---

### Description

Create a spatial grid.

### Usage

```
createSpatialGrid(
  gobject,
  sdimx_stepsize = NULL,
  sdimy_stepsize = NULL,
  sdimz_stepsize = NULL,
  minimum_padding = 1,
  name = "spatial_grid",
  return_gobject = TRUE
)
```

### Arguments

gobject                  giotto object  
 sdimx\_stepsize    stepsize along the x-axis  
 sdimy\_stepsize    stepsize along the y-axis  
 sdimz\_stepsize    stepsize along the z-axis  
 minimum\_padding      minimum padding on the edges  
 name                  name for spatial grid (default = 'spatial\_grid')  
 return\_gobject    boolean: return giotto object (default = TRUE)

**Details**

Creates a spatial grid with defined x, y (and z) dimensions. The dimension units are based on the provided spatial location units.

**Value**

giotto object with updated spatial grid slot

**Examples**

```
createSpatialGrid(gobject)
```

---

```
createSpatialGrid_2D    createSpatialGrid_2D
```

---

**Description**

create a spatial grid for 2D spatial data.

**Usage**

```
createSpatialGrid_2D(
  gobject,
  sdimx_stepsize = NULL,
  sdimy_stepsize = NULL,
  minimum_padding = 1,
  name = "spatial_grid",
  return_gobject = TRUE
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>sdimx_stepsize</code>	stepsize along the x-axis
<code>sdimy_stepsize</code>	stepsize along the y-axis
<code>minimum_padding</code>	minimum padding on the edges
<code>name</code>	name for spatial grid (default = 'spatial_grid')
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)

**Details**

Creates a spatial grid with defined x, y (and z) dimensions. The dimension units are based on the provided spatial location units.

**Value**

giotto object with updated spatial grid slot

**Examples**

```
createSpatialGrid_2D(gobject)
```

---

`createSpatialGrid_3D`    *createSpatialGrid\_3D*

---

### Description

Create a spatial grid for 3D spatial data.

### Usage

```
createSpatialGrid_3D(  
  gobject,  
  sdimx_stepsize = NULL,  
  sdimy_stepsize = NULL,  
  sdimz_stepsize = NULL,  
  minimum_padding = 1,  
  name = "spatial_grid",  
  return_gobject = TRUE  
)
```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx_stepsize</code>	stepsize along the x-axis
<code>sdimy_stepsize</code>	stepsize along the y-axis
<code>sdimz_stepsize</code>	stepsize along the z-axis
<code>minimum_padding</code>	minimum padding on the edges
<code>name</code>	name for spatial grid (default = 'spatial_grid')
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)

### Details

Creates a spatial grid with defined x, y (and z) dimensions. The dimension units are based on the provided spatial location units.

### Value

giotto object with updated spatial grid slot

### Examples

```
createSpatialGrid_3D(gobject)
```

---

```
createSpatialNetwork    createSpatialNetwork
```

---

## Description

Create a spatial network based on cell centroid physical distances.

## Usage

```
createSpatialNetwork(
  gobject,
  k = 4,
  dimensions = "all",
  maximum_distance = NULL,
  minimum_k = 0,
  name = "spatial_network",
  verbose = F,
  return_gobject = TRUE
)
```

## Arguments

<code>gobject</code>	giotto object
<code>k</code>	number of nearest neighbors based on physical distance
<code>dimensions</code>	which spatial dimensions to use (default = all)
<code>maximum_distance</code>	distance cutoff for nearest neighbors to consider
<code>minimum_k</code>	minimum nearest neighbours if <code>maximum_distance</code> != NULL
<code>name</code>	name for spatial network (default = 'spatial_network')
<code>verbose</code>	verbose
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)

## Details

Creates a spatial network connecting single-cells based on their physical distance to each other. Number of neighbors can be determined by `k`, maximum distance from each cell with or without setting a minimum `k` for each cell.

**dimensions:** default = 'all' which takes all possible dimensions. Alternatively you can provide a character vector that specifies the spatial dimensions to use, e.g. `c("sdimx", "sdimy")` or a numerical vector, e.g. `2:3`

**maximum\_distance:** to create a network based on maximum distance only, you also need to set `k` to a very high value, e.g. `k = 100`

## Value

giotto object with updated spatial network slot

## Examples

```
createSpatialNetwork(gobject)
```

---

```
create_average_detection_DT
      create_average_detection_DT
```

---

### Description

calculates average gene detection for a cell metadata factor (e.g. cluster)

### Usage

```
create_average_detection_DT(
  gobject,
  meta_data_name,
  expression_values = c("normalized", "scaled", "custom"),
  detection_threshold = 0
)
```

### Arguments

```
gobject          giotto object
meta_data_name   name of metadata column to use
expression_values
                  which expression values to use
detection_threshold
                  detection threshold to consider a gene detected
```

### Value

data.table with average gene expression values for each factor

---

```
create_average_DT      create_average_DT
```

---

### Description

calculates average gene expression for a cell metadata factor (e.g. cluster)

### Usage

```
create_average_DT(
  gobject,
  meta_data_name,
  expression_values = c("normalized", "scaled", "custom")
)
```

### Arguments

```
gobject          giotto object
meta_data_name   name of metadata column to use
expression_values
                  which expression values to use
```



**Value**

data.table with average gene expression values for each factor

---

```
create_cell_type_random_cell_IDs  
  create_cell_type_random_cell_IDs
```

---

**Description**

creates randomized cell ids within a selection of cell types

**Usage**

```
create_cell_type_random_cell_IDs(  
  gobject,  
  cluster_column = "cell_types",  
  needed_cell_types  
)
```

**Arguments**

`gobject`                giotto object to use

`cluster_column`   cluster column with cell type information

`needed_cell_types`  
                      vector of cell type names for which a random id will be found

**Details**

Details will follow.

**Value**

list of randomly sampled cell ids with same cell type composition

**Examples**

```
create_cell_type_random_cell_IDs(gobject)
```

---

create\_cluster\_matrix    *create\_cluster\_matrix*

---

### Description

creates aggregated matrix for a given clustering

### Usage

```
create_cluster_matrix(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  gene_subset = NULL
)
```

### Examples

```
create_cluster_matrix(gobject)
```

---

create\_dimObject            *create\_dimObject*

---

### Description

Creates an object that stores a dimension reduction output

### Usage

```
create_dimObject(
  name = "test",
  reduction_method = NULL,
  coordinates = NULL,
  misc = NULL
)
```

### Arguments

name	arbitrary name for object
reduction_method	method used to reduce dimensions
coordinates	accepts the coordinates after dimension reduction
misc	any additional information will be added to this slot

### Value

number of distinct colors

---

decide_cluster_order	<i>decide_cluster_order</i>
----------------------	-----------------------------

---

## Description

creates order for clusters

## Usage

```
decide_cluster_order(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  genes,  
  cluster_column = NULL,  
  cluster_order = c("size", "correlation", "custom"),  
  cluster_custom_order = NULL,  
  cor_method = "pearson",  
  hclust_method = "ward.D"  
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>genes</code>	genes to use
<code>cluster_column</code>	name of column to use for clusters
<code>cluster_order</code>	method to determine cluster order
<code>cluster_custom_order</code>	custom order for clusters
<code>cor_method</code>	method for correlation
<code>hclust_method</code>	method for hierarchical clustering

## Details

Calculates order for clusters.

## Value

custom

## Examples

```
decide_cluster_order(gobject)
```

---

detectSpatialPatterns    *detectSpatialPatterns*

---

## Description

Identify spatial patterns through PCA on average expression in a spatial grid.

## Usage

```
detectSpatialPatterns(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  spatial_grid_name = "spatial_grid",
  min_cells_per_grid = 4,
  scale_unit = F,
  ncp = 100,
  show_plot = T,
  PC_zscore = 1.5
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>spatial_grid_name</code>	name of spatial grid to use (default = 'spatial_grid')
<code>min_cells_per_grid</code>	minimum number of cells in a grid to be considered
<code>scale_unit</code>	scale features
<code>ncp</code>	number of principal components to calculate
<code>show_plot</code>	show plots
<code>PC_zscore</code>	minimum z-score of variance explained by a PC

## Details

Steps to identify spatial patterns:

- 1. average gene expression for cells within a grid, see `createSpatialGrid`
- 2. perform PCA on the average grid expression profiles
- 3. convert variance of principal components (PCs) to z-scores and select PCs based on a z-score threshold

## Value

spatial pattern object 'spatPatObj'

## Examples

```
detectSpatialPatterns(gobject)
```

dimCellPlot

*dimCellPlot***Description**

Visualize cells according to dimension reduction coordinates

**Usage**

```
dimCellPlot(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  spat_enr_names = NULL,
  cell_annotation_values,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  edge_alpha = NULL,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  legend_text = 8,
  axis_text = 8,
  axis_title = 8,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
```

```

    default_save_name = "dimCellPlot"
)

```

### Arguments

```

gobject          giotto object
dim_reduction_to_use
                  dimension reduction to use
dim_reduction_name
                  dimension reduction name
dim1_to_use       dimension to use on x-axis
dim2_to_use       dimension to use on y-axis
spat_enr_names    names of spatial enrichment results to include
cell_annotation_values
                  numeric cell annotation columns
show_NN_network   show underlying NN network
nn_network_to_use
                  type of NN network to use (kNN vs sNN)
network_name      name of NN network to use, if show_NN_network = TRUE
cell_color_gradient
                  vector with 3 colors for numeric data
gradient_midpoint
                  midpoint for color gradient
gradient_limits   vector with lower and upper limits
select_cell_groups
                  select subset of cells/clusters based on cell_color parameter
select_cells      select subset of cells based on cell IDs
show_other_cells  display not selected cells
other_cell_color  color of not selected cells
other_point_size  size of not selected cells
show_cluster_center
                  plot center of selected clusters
show_center_label
                  plot label of selected clusters
center_point_size
                  size of center points
label_size        size of labels
label_fontface    font of labels
edge_alpha        column to use for alpha of the edges
point_size        size of point (cell)
point_border_col  color of border around points

```

point_border_stroke	stroke size of border around points
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
title	title for plot, defaults to cell_color parameter

### Details

Description of parameters. For 3D plots see [dimCellPlot2D](#)

### Value

ggplot

### Examples

```
dimCellPlot(gobject)
```

---

dimCellPlot2D	<i>dimCellPlot2D</i>
---------------	----------------------

---

### Description

Visualize cells according to dimension reduction coordinates

### Usage

```
dimCellPlot2D(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  spat_enr_names = NULL,
  cell_annotation_values,
  show_NN_network = F,
```

```

nn_network_to_use = "sNN",
network_name = "sNN.pca",
cell_color_gradient = c("blue", "white", "red"),
gradient_midpoint = NULL,
gradient_limits = NULL,
select_cell_groups = NULL,
select_cells = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 0.5,
show_cluster_center = F,
show_center_label = T,
center_point_size = 4,
center_point_border_col = "black",
center_point_border_stroke = 0.1,
label_size = 4,
label_fontface = "bold",
edge_alpha = NULL,
point_size = 1,
point_border_col = "black",
point_border_stroke = 0.1,
show_legend = T,
legend_text = 8,
axis_text = 8,
axis_title = 8,
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "dimCellPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_annotation_values</code>	numeric cell annotation columns
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)



network_name	name of NN network to use, if show_NN_network = TRUE
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
title	title for plot, defaults to cell_color parameter

**Details**

Description of parameters. For 3D plots see [dimPlot3D](#)

**Value**

ggplot

**Examples**

```
dimCellPlot2D(gobject)
```

---

dimGenePlot

*dimGenePlot*


---

**Description**

Visualize cells and gene expression according to dimension reduction coordinates

**Usage**

```
dimGenePlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  edge_alpha = NULL,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  genes_high_color = "red",
  genes_mid_color = "white",
  genes_low_color = "blue",
  point_border_col = "black",
  point_border_stroke = 0.1,
  midpoint = 0,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_legend = T,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "dimGenePlot"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha</code>	column to use for alpha of the edges
<code>scale_alpha_with_expression</code>	scale expression with ggplot alpha parameter
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>midpoint</code>	size of point (cell)
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>show_legend</code>	show legend
<code>show_plot</code>	show plots
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>...</code>	parameters for <code>cowplot::save_plot()</code>

**Details**

Description of parameters.

**Value**

ggplot

**See Also**[dimGenePlot3D](#)**Examples**

```
dimGenePlot(gobject)
```

---

dimGenePlot2D

---

*dimGenePlot2D*


---

**Description**

Visualize cells and gene expression according to dimension reduction coordinates

**Usage**

```
dimGenePlot2D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  edge_alpha = NULL,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  genes_high_color = "red",
  genes_mid_color = "white",
  genes_low_color = "blue",
  point_border_col = "black",
  point_border_stroke = 0.1,
  midpoint = 0,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_legend = T,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "dimGenePlot2D"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha</code>	column to use for alpha of the edges
<code>scale_alpha_with_expression</code>	scale expression with ggplot alpha parameter
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>midpoint</code>	size of point (cell)
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>show_legend</code>	show legend
<code>show_plot</code>	show plots
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>...</code>	parameters for <code>cowplot::save_plot()</code>

**Details**

Description of parameters.

**Value**

ggplot

**See Also**[dimGenePlot3D](#)**Examples**

```
dimGenePlot2D(gobject)
```

---

dimGenePlot3D	<i>dimGenePlot3D</i>
---------------	----------------------

---

**Description**

Visualize cells and gene expression according to dimension reduction coordinates

**Usage**

```
dimGenePlot3D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = 3,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  cluster_column = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 1,
  edge_alpha = NULL,
  point_size = 2,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  show_legend = T,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "dimGenePlot3D"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>show_legend</code>	show legend
<code>show_plot</code>	show plots
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>...</code>	parameters for <code>cowplot::save_plot()</code>

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
dimGenePlot3D(gobject)
```

dimPlot

*dimPlot***Description**

Visualize cells according to dimension reduction coordinates

**Usage**

```
dimPlot(
  gobject,
  group_by = NULL,
  group_by_subset = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  spat_enr_names = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  edge_alpha = NULL,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  legend_text = 8,
  axis_text = 8,
  axis_title = 8,
  title = NULL,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
```



```

    cow_align = "h",
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "dimPlot"
)

```

## Arguments

<code>gobject</code>	giotto object
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters

center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
title	title for plot, defaults to cell_color parameter
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column

## Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [dimPlot3D](#)

## Value

ggplot

## Examples

```
dimPlot(gobject)
```

---

dimPlot2D

*dimPlot2D*


---

## Description

Visualize cells according to dimension reduction coordinates

## Usage

```
dimPlot2D(
  gobject,
  group_by = NULL,
  group_by_subset = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  spat_enr_names = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  edge_alpha = NULL,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  title = NULL,
  show_legend = T,
  legend_text = 8,
  axis_text = 8,
  axis_title = 8,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
```

```

cow_align = "h",
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "dimPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters

center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
title	title for plot, defaults to cell_color parameter
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column

## Details

Description of parameters. For 3D plots see [dimPlot3D](#)

## Value

ggplot

## Examples

```
dimPlot2D(gobject)
```

---

dimPlot2D_single	<i>dimPlot2D_single</i>
------------------	-------------------------

---

## Description

Visualize cells according to dimension reduction coordinates

## Usage

```
dimPlot2D_single(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  spat_enr_names = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  edge_alpha = NULL,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  title = NULL,
  show_legend = T,
  legend_text = 8,
  axis_text = 8,
  axis_title = 8,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "dimPlot2D_single"
```

)

**Arguments**

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)

point_border_col	color of border around points
point_border_stroke	stroke size of border around points
title	title for plot, defaults to cell_color parameter
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

Details

Description of parameters. For 3D plots see [dimPlot3D](#)

Value

ggplot

Examples

dimPlot2D\_single(gobject)

---

dimPlot3D	<i>dimPlot3D</i>
-----------	------------------

---

Description

Visualize cells according to dimension reduction coordinates

Usage

```
dimPlot3D(  
  gobject,  
  dim_reduction_to_use = "umap",  
  dim_reduction_name = "umap",  
  dim1_to_use = 1,  
  dim2_to_use = 2,  
  dim3_to_use = 3,  
  select_cell_groups = NULL,  
  select_cells = NULL,  
  show_other_cells = T,  
  other_cell_color = "lightgrey",  
  other_point_size = 2,
```



```

    show_NN_network = F,
    nn_network_to_use = "sNN",
    network_name = "sNN.pca",
    color_as_factor = T,
    cell_color = NULL,
    cell_color_code = NULL,
    show_cluster_center = F,
    show_center_label = T,
    center_point_size = 4,
    label_size = 4,
    edge_alpha = NULL,
    point_size = 3,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "dim3D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>color_as_factor</code>	convert color column to factor
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>show_cluster_center</code>	plot center of selected clusters

show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
show_legend	show legend

Details

Description of parameters.

Value

plotly

Examples

```
dimPlot3D(gobject)
```

---

direction_test_CPG	<i>direction_test_CPG</i>
--------------------	---------------------------

---

Description

shows direction of change

Usage

```
direction_test(x, min_fdr = 0.05)
```

Examples

```
direction_test_CPG()
```

---

doHclust	<i>doHclust</i>
----------	-----------------

---

## Description

cluster cells using hierarchical clustering algorithm

## Usage

```
doHclust(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes_to_use = NULL,
  dim_reduction_to_use = c("cells", "pca", "umap", "tsne"),
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  distance_method = c("pearson", "spearman", "original", "euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski"),
  agglomeration_method = c("ward.D2", "ward.D", "single", "complete", "average",
    "mcquitty", "median", "centroid"),
  k = 10,
  h = NULL,
  name = "hclust",
  return_gobject = TRUE,
  set_seed = T,
  seed_number = 1234
)
```

## Arguments

gobject	giotto object
expression_values	expression values to use
genes_to_use	subset of genes to use
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimensions reduction name
dimensions_to_use	dimensions to use
distance_method	distance method
agglomeration_method	agglomeration method for hclust
k	number of final clusters
h	cut hierarchical tree at height = h
name	name for hierarchical clustering
return_gobject	boolean: return giotto object (default = TRUE)
set_seed	set seed
seed_number	number for seed

**Details**

Description on how to use Kmeans clustering method.

**Value**

giotto object with new clusters appended to cell metadata

**See Also**

[hclust](#)

**Examples**

```
doHclust(gobject)
```

---

doHMRF	<i>doHMRF</i>
--------	---------------

---

**Description**

Run HMRF

**Usage**

```
doHMRF(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  spatial_network_name = "spatial_network",
  spatial_genes = NULL,
  spatial_dimensions = c("sdimx", "sdimy", "sdimz"),
  dim_reduction_to_use = NULL,
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  name = "test",
  k = 10,
  betas = c(0, 2, 50),
  tolerance = 1e-10,
  zscore = c("none", "rowcol", "colrow"),
  numinit = 100,
  python_path = NULL,
  output_folder = NULL,
  overwrite_output = TRUE
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>spatial_network_name</code>	name of spatial network to use for HMRF

spatial\_genes    spatial genes to use for HMRF  
 spatial\_dimensions    select spatial dimensions to use, default is all possible dimensions  
 dim\_reduction\_to\_use    use another dimension reduction set as input  
 dim\_reduction\_name    name of dimension reduction set to use  
 dimensions\_to\_use    number of dimensions to use as input  
 name    name of HMRF run  
 k    number of HMRF domains  
 betas    betas to test for  
 tolerance    tolerance  
 zscore    zscore  
 numinit    number of initializations  
 python\_path    python path to use  
 output\_folder    output folder to save results  
 overwrite\_output    overwrite output folder

## Details

Description of HMRF parameters ...

## Value

Creates a directory with results that can be viewed with viewHMRFresults

## Examples

```
doHMRF(gobject)
```

---

doKmeans	<i>doKmeans</i>
----------	-----------------

---

## Description

cluster cells using kmeans algorithm

## Usage

```
doKmeans(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes_to_use = NULL,
  dim_reduction_to_use = c("cells", "pca", "umap", "tsne"),
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  distance_method = c("original", "pearson", "spearman", "euclidean", "maximum",
```

```

    "manhattan", "canberra", "binary", "minkowski"),
    centers = 10,
    iter_max = 100,
    nstart = 1000,
    algorithm = "Hartigan-Wong",
    name = "kmeans",
    return_gobject = TRUE,
    set_seed = T,
    seed_number = 1234
)

```

### Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>genes_to_use</code>	subset of genes to use
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimensions reduction name
<code>dimensions_to_use</code>	dimensions to use
<code>distance_method</code>	distance method
<code>centers</code>	number of final clusters
<code>iter_max</code>	kmeans maximum iterations
<code>nstart</code>	kmeans nstart
<code>algorithm</code>	kmeans algorithm
<code>name</code>	name for kmeans clustering
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>set_seed</code>	set seed
<code>seed_number</code>	number for seed

### Details

Description on how to use Kmeans clustering method.

### Value

giotto object with new clusters appended to cell metadata

### See Also

[kmeans](#)

### Examples

```
doKmeans(gobject)
```

doLeidenCluster

*doLeidenCluster***Description**

cluster cells using a NN-network and the Leiden community detection algorithm

**Usage**

```
doLeidenCluster(
  gobject,
  name = "leiden_clus",
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  python_path = NULL,
  resolution = 1,
  weight_col = "weight",
  partition_type = c("RBConfigurationVertexPartition", "ModularityVertexPartition"),
  init_membership = NULL,
  n_iterations = 1000,
  return_gobject = TRUE,
  set_seed = T,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for cluster
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>python_path</code>	specify specific path to python if required
<code>resolution</code>	resolution
<code>weight_col</code>	weight column to use for edges
<code>partition_type</code>	The type of partition to use for optimisation.
<code>init_membership</code>	initial membership of cells for the partition
<code>n_iterations</code>	number of iterations to run the Leiden algorithm. If the number of iterations is negative, the Leiden algorithm is run until an iteration in which there was no improvement.
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>set_seed</code>	set seed
<code>seed_number</code>	number for seed

## Details

This function is a wrapper for the Leiden algorithm implemented in python, which can detect communities in graphs of millions of nodes (cells), as long as they can fit in memory. See the <https://github.com/vtraag/leidenalg> github page or the <https://leidenalg.readthedocs.io/en/stable/index.html> readthedocs page for more information.

Partition types available and information:

- **RBConfigurationVertexPartition**: Implements Reichardt and Bornholdt's Potts model with a configuration null model. This quality function is well-defined only for positive edge weights. This quality function uses a linear resolution parameter.
- **ModularityVertexPartition**: Implements modularity. This quality function is well-defined only for positive edge weights. It does *not* use the resolution parameter

Set `weight_col = NULL` to give equal weight (=1) to each edge.

## Value

giotto object with new clusters appended to cell metadata

## Examples

```
doLeidenCluster(gobject)
```

---

doLeidenSubCluster	<i>doLeidenSubCluster</i>
--------------------	---------------------------

---

## Description

Further subcluster cells using a NN-network and the Leiden algorithm

## Usage

```
doLeidenSubCluster(
  gobject,
  name = "sub_pleiden_clus",
  cluster_column = NULL,
  selected_clusters = NULL,
  hvg_param = list(reverse_log_scale = T, difference_in_variance = 1, expression_values
    = "normalized"),
  hvg_min_perc_cells = 5,
  hvg_mean_expr_det = 1,
  use_all_genes_as_hvg = FALSE,
  min_nr_of_hvg = 5,
  pca_param = list(expression_values = "normalized", scale_unit = T),
  nn_param = list(dimensions_to_use = 1:20),
  k_neighbors = 10,
  resolution = 0.5,
  n_iterations = 500,
  python_path = NULL,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
```



```

    return_gobject = TRUE,
    verbose = T
)

```

### Arguments

<code>gobject</code>	giotto object
<code>name</code>	name for new clustering result
<code>cluster_column</code>	cluster column to subcluster
<code>selected_clusters</code>	only do subclustering on these clusters
<code>hvg_param</code>	parameters for calculateHVG
<code>hvg_min_perc_cells</code>	threshold for detection in min percentage of cells
<code>hvg_mean_expr_det</code>	threshold for mean expression level in cells with detection
<code>use_all_genes_as_hvg</code>	forces all genes to be HVG and to be used as input for PCA
<code>min_nr_of_hvg</code>	minimum number of HVG, or all genes will be used as input for PCA
<code>pca_param</code>	parameters for runPCA
<code>nn_param</code>	parameters for parameters for createNearestNetwork
<code>k_neighbors</code>	number of k for createNearestNetwork
<code>resolution</code>	resolution of Leiden clustering
<code>n_iterations</code>	number of iterations to run the Leiden algorithm.
<code>python_path</code>	specify specific path to python if required
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>verbose</code>	verbose

### Details

This function performs subclustering using the Leiden algorithm on selected clusters. The systematic steps are:

- 1. subset Giotto object
- 2. identify highly variable genes
- 3. run PCA
- 4. create nearest neighbouring network
- 5. do Leiden clustering

### Value

giotto object with new subclusters appended to cell metadata

**See Also**[doLeidenCluster](#)**Examples**

```
doLeidenSubCluster(gobject)
```

doLouvainCluster

*doLouvainCluster***Description**

cluster cells using a NN-network and the Louvain algorithm.

**Usage**

```
doLouvainCluster(
  gobject,
  version = c("community", "multinet"),
  name = "louvain_clus",
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  python_path = NULL,
  resolution = 1,
  weight_col = NULL,
  gamma = 1,
  omega = 1,
  louv_random = F,
  return_gobject = TRUE,
  set_seed = F,
  seed_number = 1234,
  ...
)
```

**Arguments**

gobject	giotto object
version	implemented version of Louvain clustering to use
name	name for cluster
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use
python_path	[community] specify specific path to python if required
resolution	[community] resolution
gamma	[multinet] Resolution parameter for modularity in the generalized louvain method.
omega	[multinet] Inter-layer weight parameter in the generalized louvain method.
return_gobject	boolean: return giotto object (default = TRUE)
set_seed	set seed
seed_number	number for seed

**Details**

Louvain clustering using the community or multinet implementation of the louvain clustering algorithm.

**Value**

giotto object with new clusters appended to cell metadata

**See Also**

[doLouvainCluster\\_community](#) and [doLouvainCluster\\_multinet](#)

**Examples**

```
doLouvainCluster(gobject)
```

---

```
doLouvainCluster_community
doLouvainCluster_community
```

---

**Description**

cluster cells using a NN-network and the Louvain algorithm from the community module in Python

**Usage**

```
doLouvainCluster_community(
    gobject,
    name = "louvain_clus",
    nn_network_to_use = "sNN",
    network_name = "sNN.pca",
    python_path = NULL,
    resolution = 1,
    weight_col = NULL,
    louv_random = F,
    return_gobject = TRUE,
    set_seed = F,
    seed_number = 1234,
    ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for cluster
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>python_path</code>	specify specific path to python if required
<code>resolution</code>	resolution

weight_col	weight column to use for edges
louv_random	Will randomize the node evaluation order and the community evaluation order to get different partitions at each call
return_gobject	boolean: return giotto object (default = TRUE)
set_seed	set seed
seed_number	number for seed

### Details

This function is a wrapper for the Louvain algorithm implemented in Python, which can detect communities in graphs of nodes (cells). See the <https://python-louvain.readthedocs.io/en/latest/index.html> readthedocs page for more information.

Set *weight\_col* = *NULL* to give equal weight (=1) to each edge.

### Value

giotto object with new clusters appended to cell metadata

### Examples

```
doLouvainCluster_community(gobject)
```

---

```
doLouvainCluster_multinet
doLouvainCluster_multinet
```

---

### Description

cluster cells using a NN-network and the Louvain algorithm from the multinet package in R.

### Usage

```
doLouvainCluster_multinet(
  gobject,
  name = "louvain_clus",
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  gamma = 1,
  omega = 1,
  return_gobject = TRUE,
  set_seed = F,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for cluster
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>gamma</code>	Resolution parameter for modularity in the generalized louvain method.
<code>omega</code>	Inter-layer weight parameter in the generalized louvain method.
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>set_seed</code>	set seed
<code>seed_number</code>	number for seed

**Details**

See [glouvain\\_ml](#) from the `multinet` package in R for more information.

**Value**

giotto object with new clusters appended to cell metadata

**Examples**

```
doLouvainCluster_multinet(gobject)
```

---

<code>doLouvainSubCluster</code>	<i>doLouvainSubCluster</i>
----------------------------------	----------------------------

---

**Description**

subcluster cells using a NN-network and the Louvain algorithm

**Usage**

```
doLouvainSubCluster(
  gobject,
  name = "sub_louvain_clus",
  version = c("community", "multinet"),
  cluster_column = NULL,
  selected_clusters = NULL,
  hvg_param = list(reverse_log_scale = T, difference_in_variance = 1, expression_values
    = "normalized"),
  hvg_min_perc_cells = 5,
  hvg_mean_expr_det = 1,
  use_all_genes_as_hvg = FALSE,
  min_nr_of_hvg = 5,
  pca_param = list(expression_values = "normalized", scale_unit = T),
  nn_param = list(dimensions_to_use = 1:20),
  k_neighbors = 10,
  resolution = 0.5,
```

```

    gamma = 1,
    omega = 1,
    python_path = NULL,
    nn_network_to_use = "sNN",
    network_name = "sNN.pca",
    return_gobject = TRUE,
    verbose = T
)

```

## Arguments

<code>gobject</code>	giotto object
<code>name</code>	name for new clustering result
<code>version</code>	version of Louvain algorithm to use
<code>cluster_column</code>	cluster column to subcluster
<code>selected_clusters</code>	only do subclustering on these clusters
<code>hvg_param</code>	parameters for calculateHVG
<code>hvg_min_perc_cells</code>	threshold for detection in min percentage of cells
<code>hvg_mean_expr_det</code>	threshold for mean expression level in cells with detection
<code>use_all_genes_as_hvg</code>	forces all genes to be HVG and to be used as input for PCA
<code>min_nr_of_hvg</code>	minimum number of HVG, or all genes will be used as input for PCA
<code>pca_param</code>	parameters for runPCA
<code>nn_param</code>	parameters for parameters for createNearestNetwork
<code>k_neighbors</code>	number of k for createNearestNetwork
<code>resolution</code>	resolution for community algorithm
<code>gamma</code>	gamma
<code>omega</code>	omega
<code>python_path</code>	specify specific path to python if required
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>verbose</code>	verbose

## Details

This function performs subclustering using the Louvain algorithm on selected clusters. The systematic steps are:

- 1. subset Giotto object
- 2. identify highly variable genes
- 3. run PCA
- 4. create nearest neighbouring network
- 5. do Louvain clustering

**Value**

giotto object with new subclusters appended to cell metadata

**See Also**

[doLouvainCluster\\_multinet](#) and [doLouvainCluster\\_community](#)

**Examples**

```
doLouvainSubCluster(gobject)
```

---

```
doLouvainSubCluster_community
```

```
doLouvainSubCluster_community
```

---

**Description**

subcluster cells using a NN-network and the Louvain community detection algorithm

**Usage**

```
doLouvainSubCluster_community(
  gobject,
  name = "sub_louvain_comm_clus",
  cluster_column = NULL,
  selected_clusters = NULL,
  hvg_param = list(reverse_log_scale = T, difference_in_variance = 1, expression_values
    = "normalized"),
  hvg_min_perc_cells = 5,
  hvg_mean_expr_det = 1,
  use_all_genes_as_hvg = FALSE,
  min_nr_of_hvg = 5,
  pca_param = list(expression_values = "normalized", scale_unit = T),
  nn_param = list(dimensions_to_use = 1:20),
  k_neighbors = 10,
  resolution = 0.5,
  python_path = NULL,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  return_gobject = TRUE,
  verbose = T
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for new clustering result
<code>cluster_column</code>	cluster column to subcluster
<code>selected_clusters</code>	only do subclustering on these clusters

hvg_param	parameters for calculateHVG
hvg_min_perc_cells	threshold for detection in min percentage of cells
hvg_mean_expr_det	threshold for mean expression level in cells with detection
use_all_genes_as_hvg	forces all genes to be HVG and to be used as input for PCA
min_nr_of_hvg	minimum number of HVG, or all genes will be used as input for PCA
pca_param	parameters for runPCA
nn_param	parameters for parameters for createNearestNetwork
k_neighbors	number of k for createNearestNetwork
resolution	resolution
python_path	specify specific path to python if required
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use
return_gobject	boolean: return giotto object (default = TRUE)
verbose	verbose

## Details

This function performs subclustering using the Louvain community algorithm on selected clusters. The systematic steps are:

- 1. subset Giotto object
- 2. identify highly variable genes
- 3. run PCA
- 4. create nearest neighbouring network
- 5. do Louvain community clustering

## Value

giotto object with new subclusters appended to cell metadata

## See Also

[doLouvainCluster\\_community](#)

## Examples

```
doLouvainSubCluster_community(gobject)
```



---

```
doLouvainSubCluster_multinet
    doLouvainSubCluster_multinet
```

---

## Description

subcluster cells using a NN-network and the Louvain multinet detection algorithm

## Usage

```
doLouvainSubCluster_multinet(
  gobject,
  name = "sub_louvain_mult_clus",
  cluster_column = NULL,
  selected_clusters = NULL,
  hvg_param = list(reverse_log_scale = T, difference_in_variance = 1, expression_values
    = "normalized"),
  hvg_min_perc_cells = 5,
  hvg_mean_expr_det = 1,
  use_all_genes_as_hvg = FALSE,
  min_nr_of_hvg = 5,
  pca_param = list(expression_values = "normalized", scale_unit = T),
  nn_param = list(dimensions_to_use = 1:20),
  k_neighbors = 10,
  gamma = 1,
  omega = 1,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  return_gobject = TRUE,
  verbose = T
)
```

## Arguments

<code>gobject</code>	giotto object
<code>name</code>	name for new clustering result
<code>cluster_column</code>	cluster column to subcluster
<code>selected_clusters</code>	only do subclustering on these clusters
<code>hvg_param</code>	parameters for calculateHVG
<code>hvg_min_perc_cells</code>	threshold for detection in min percentage of cells
<code>hvg_mean_expr_det</code>	threshold for mean expression level in cells with detection
<code>use_all_genes_as_hvg</code>	forces all genes to be HVG and to be used as input for PCA
<code>min_nr_of_hvg</code>	minimum number of HVG, or all genes will be used as input for PCA
<code>pca_param</code>	parameters for runPCA

nn_param	parameters for parameters for createNearestNetwork
k_neighbors	number of k for createNearestNetwork
gamma	gamma
omega	omega
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use
return_gobject	boolean: return giotto object (default = TRUE)
verbose	verbose
python_path	specify specific path to python if required

### Details

This function performs subclustering using the Louvain multinet algorithm on selected clusters. The systematic steps are:

- 1. subset Giotto object
- 2. identify highly variable genes
- 3. run PCA
- 4. create nearest neighbouring network
- 5. do Louvain multinet clustering

### Value

giotto object with new subclusters appended to cell metadata

### See Also

[doLouvainCluster\\_multinet](#)

### Examples

```
doLouvainSubCluster_multinet(gobject)
```

---

doRandomWalkCluster     *doRandomWalkCluster*

---

### Description

Cluster cells using a random walk approach.

**Usage**

```
doRandomWalkCluster(
  gobject,
  name = "random_walk_clus",
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  walk_steps = 4,
  walk_clusters = 10,
  walk_weights = NA,
  return_gobject = TRUE,
  set_seed = F,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for cluster
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>walk_steps</code>	number of walking steps
<code>walk_clusters</code>	number of final clusters
<code>walk_weights</code>	cluster column defining the walk weights
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>set_seed</code>	set seed
<code>seed_number</code>	number for seed

**Details**

See [cluster\\_walktrap](#) function from the `igraph` package in R for more information.

**Value**

giotto object with new clusters appended to cell metadata

**Examples**

```
doRandomWalkCluster(gobject)
```

doSNNCluster

*doSNNCluster***Description**

Cluster cells using a SNN cluster approach.

**Usage**

```
doSNNCluster(
  gobject,
  name = "sNN_clus",
  nn_network_to_use = "kNN",
  network_name = "kNN.pca",
  k = 20,
  eps = 4,
  minPts = 16,
  borderPoints = TRUE,
  return_gobject = TRUE,
  set_seed = F,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for cluster
<code>nn_network_to_use</code>	type of NN network to use (only works on kNN)
<code>network_name</code>	name of kNN network to use
<code>k</code>	Neighborhood size for nearest neighbor sparsification to create the shared NN graph.
<code>eps</code>	Two objects are only reachable from each other if they share at least <code>eps</code> nearest neighbors.
<code>minPts</code>	minimum number of points that share at least <code>eps</code> nearest neighbors for a point to be considered a core points.
<code>borderPoints</code>	should borderPoints be assigned to clusters like in DBSCAN?
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>set_seed</code>	set seed
<code>seed_number</code>	number for seed

**Details**

See [sNNclust](#) from dbscan package

**Value**

giotto object with new clusters appended to cell metadata

**Examples**

```
doSNNCluster(gobject)
```

---

dt_to_matrix	<i>dt_to_matrix</i>
--------------	---------------------

---

**Description**

converts data.table to matrix

**Usage**

```
dt_to_matrix(x)
```

**Examples**

```
dt_to_matrix(x)
```

---

exportGiottoViewer	<i>exportGiottoViewer</i>
--------------------	---------------------------

---

**Description**

compute highly variable genes

**Usage**

```
exportGiottoViewer(
  gobject,
  output_directory = NULL,
  spat_enr_names = NULL,
  factor_annotations,
  numeric_annotations,
  dim_reductions,
  dim_reduction_names,
  expression_values = c("scaled", "normalized", "custom"),
  dim_red_rounding = NULL,
  dim_red_rescale = c(-20, 20),
  expression_rounding = 2,
  overwrite_dir = T,
  verbose = T
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>output_directory</code>	directory where to save the files
<code>spat_enr_names</code>	spatial enrichment results to include for annotations
<code>factor_annotations</code>	giotto cell annotations to view as factor
<code>numeric_annotations</code>	giotto cell annotations to view as numeric
<code>dim_reductions</code>	high level dimension reductions to view
<code>dim_reduction_names</code>	specific dimension reduction names
<code>expression_values</code>	expression values to use in Viewer
<code>dim_red_rounding</code>	numerical indicating how to round the coordinates
<code>dim_red_rescale</code>	numericals to rescale the coordinates
<code>expression_rounding</code>	numerical indicating how to round the expression data
<code>overwrite_dir</code>	overwrite files in the directory if it already existed
<code>verbose</code>	be verbose

**Details**

Giotto Viewer expects the results from Giotto Analyzer in a specific format, which is provided by this function. To include enrichment results from [createSpatialEnrich](#) include the provided spatial enrichment name (default PAGE or rank) and add the gene signature names (.e.g cell types) to the numeric annotations parameter.

**Value**

writes the necessary output to use in Giotto Viewer

**Examples**

```
exportGiottoViewer(gobject)
```

---

<code>exprOnlyCellCellcommunicationScores</code>
<i>exprOnlyCellCellcommunicationScores</i>

---

**Description**

Cell-Cell communication scores based on expression only

**Usage**

```
exprOnlyCellCellcommunicationScores(
  gobject,
  cluster_column = "cell_types",
  random_iter = 100,
  gene_set_1,
  gene_set_2,
  log2FC_addendum = 0.1,
  verbose = T
)
```

**Arguments**

gobject	giotto object to use
cluster_column	cluster column with cell type information
random_iter	number of iterations
gene_set_1	first specific gene set from gene pairs
gene_set_2	second specific gene set from gene pairs
log2FC_addendum	addendum to add when calculating log2FC
verbose	verbose

**Details**

Details will follow.

**Value**

Cell-Cell communication scores for gene pairs based on expression only

**Examples**

```
exprOnlyCellCellcommunicationScores(gobject)
```

---

extended_gini_fun	<i>extended_gini_fun</i>
-------------------	--------------------------

---

**Description**

calculate gini coefficient on a minimum length vector

**Usage**

```
extended_gini_fun(x, weights = rep(1, length = length(x)), minimum_length = 16)
```

**Value**

gini coefficient

---

extractNearestNetwork	<i>extractNearestNetwork</i>
-----------------------	------------------------------

---

### Description

Extracts a NN-network from a Giotto object

### Usage

```
extractNearestNetwork(
  gobject,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  output = c("igraph", "data.table")
)
```

### Arguments

gobject	giotto object
nn_network_to_use	kNN or sNN
network_name	name of NN network to be used
output	return a igraph or data.table object

### Value

igraph or data.table object

### Examples

```
extractNearestNetwork(gobject)
```

---

fDataDT	<i>fDataDT</i>
---------	----------------

---

### Description

show gene metadata

### Usage

```
fDataDT(gobject)
```

### Arguments

gobject	giotto object
---------	---------------

### Value

data.table with gene metadata



**Examples**

```
pDataDT(gobject)
```

---

filterCombinations	<i>filterCombinations</i>
--------------------	---------------------------

---

**Description**

Shows how many genes and cells are lost with combinations of thresholds.

**Usage**

```
filterCombinations(
  gobject,
  expression_values = c("raw", "normalized", "scaled", "custom"),
  expression_thresholds = c(1, 2),
  gene_det_in_min_cells = c(5, 50),
  min_det_genes_per_cell = c(200, 400),
  scale_x_axis = "identity",
  x_axis_offset = 0,
  scale_y_axis = "identity",
  y_axis_offset = 0,
  show_plot = TRUE
)
```

**Arguments**

gobject	giotto object
expression_values	expression values to use
expression_thresholds	all thresholds to consider a gene expressed
gene_det_in_min_cells	minimum number of cells that should express a gene to consider that gene further
min_det_genes_per_cell	minimum number of expressed genes per cell to consider that cell further
scale_x_axis	ggplot transformation for x-axis (e.g. log2)
x_axis_offset	x-axis offset to be used together with the scaling transformation
scale_y_axis	ggplot transformation for y-axis (e.g. log2)
y_axis_offset	y-axis offset to be used together with the scaling transformation
show_plot	show plot

**Details**

Creates a scatterplot that visualizes the number of genes and cells that are lost with a specific combination of a gene and cell threshold given an arbitrary cutoff to call a gene expressed. This function can be used to make an informed decision at the filtering step with filterGiotto.

**Value**

list of data.table and ggplot object

**Examples**

```
filterCombinations(gobject)
```

---

filterCPGscores	<i>filterCPGscores</i>
-----------------	------------------------

---

**Description**

visualize Cell Proximity Gene enrichment scores

**Usage**

```
filterCPGscores(  
  CPGscore,  
  min_cells = 5,  
  min_fdr = 0.05,  
  min_spat_diff = 0.2,  
  min_log2_fc = 0.5,  
  keep_int_duplicates = TRUE,  
  direction = c("both", "up", "down")  
)
```

**Arguments**

min_cells	min number of cells threshold
min_fdr	false_discovery threshold
min_spat_diff	spatial difference threshold
min_log2_fc	min log2 fold-change
keep_int_duplicates	keep both cell_A-cell_B and cell_B-cell_A
direction	expression changes to keep
method	visualization method

**Details**

This function filters the output from `getCellProximityGeneScores` based on false-discovery rate, minimum absolute difference, minimum log fold-change and direction of change.

**Value**

Gene to gene scores in data.table format

**Examples**

```
filterCPGscores(CPGscore)
```

---

filterDistributions	<i>filterDistributions</i>
---------------------	----------------------------

---

## Description

show gene or cell distribution after filtering on expression threshold

## Usage

```
filterDistributions(  
  gobject,  
  expression_values = c("raw", "normalized", "scaled", "custom"),  
  expression_threshold = 1,  
  detection = c("genes", "cells"),  
  plot_type = c("histogram", "violin"),  
  nr_bins = 30,  
  fill_color = "lightblue",  
  scale_axis = "identity",  
  axis_offset = 0,  
  show_plot = TRUE  
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>expression_threshold</code>	threshold to consider a gene expressed
<code>detection</code>	consider genes or cells
<code>plot_type</code>	type of plot
<code>nr_bins</code>	number of bins for histogram plot
<code>fill_color</code>	fill color for plots
<code>scale_axis</code>	ggplot transformation for axis (e.g. log2)
<code>axis_offset</code>	offset to be used together with the scaling transformation
<code>show_plot</code>	show plot

## Value

ggplot object

## Examples

```
filterDistributions(gobject)
```

---

filterGiotto	<i>filterGiotto</i>
--------------	---------------------

---

## Description

filter Giotto object based on expression threshold

## Usage

```
filterGiotto(  
  gobject,  
  expression_values = c("raw", "normalized", "scaled", "custom"),  
  expression_threshold = 1,  
  gene_det_in_min_cells = 100,  
  min_det_genes_per_cell = 100,  
  verbose = F  
)
```

## Arguments

gobject	giotto object
expression_values	expression values to use
expression_threshold	threshold to consider a gene expressed
gene_det_in_min_cells	minimum # of cells that need to express a gene
min_det_genes_per_cell	minimum # of genes that need to be detected in a cell
verbose	verbose

## Details

The function [filterCombinations](#) can be used to explore the effect of different parameter values.

## Value

giotto object

## Examples

```
filterGiotto(gobject)
```

---

findGiniMarkers	<i>findGiniMarkers</i>
-----------------	------------------------

---

## Description

Identify marker genes for selected clusters based on gini detection and expression scores.

## Usage

```
findGiniMarkers(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  group_1 = NULL,
  group_2 = NULL,
  min_expr_gini_score = 0.5,
  min_det_gini_score = 0.5,
  detection_threshold = 0,
  rank_score = 1
)
```

## Arguments

gobject	giotto object
expression_values	gene expression values to use
cluster_column	clusters to use
subset_clusters	selection of clusters to compare
group_1	group 1 cluster IDs from cluster_column for pairwise comparison
group_2	group 2 cluster IDs from cluster_column for pairwise comparison
min_expr_gini_score	filter on minimum gini coefficient for expression
min_det_gini_score	filter on minimum gini coefficient for detection
detection_threshold	detection threshold for gene expression
rank_score	rank scores to include

## Details

Detection of marker genes using the [https://en.wikipedia.org/wiki/Gini\\_coefficient](https://en.wikipedia.org/wiki/Gini_coefficient) gini coefficient is based on the following steps/principles per gene:

- 1. calculate average expression per cluster
- 2. calculate detection fraction per cluster
- 3. calculate gini-coefficient for av. expression values over all clusters

- 4. calculate gini-coefficient for detection fractions over all clusters
- 5. convert gini-scores to rank scores
- 6. for each gene create combined gini score = av. expr gini x detection gini
- 7. for each gene create and sort on combined rank score = expr rank x detection rank

As a results "top gini" genes are genes that are very selectively expressed in a specific cluster, however not always expressed in all cells of that cluster. In other words highly specific, but not necessarily sensitive at the single-cell level.

To perform differential expression between cluster groups you need to specify cluster IDs to the parameters *group\_1* and *group\_2*.

### Value

data.table with marker genes

### Examples

```
findGiniMarkers(gobject)
```

---

```
findGiniMarkers_one_vs_all
      findGiniMarkers_one_vs_all
```

---

### Description

Identify marker genes for all clusters in a one vs all manner based on gini detection and expression scores.

### Usage

```
findGiniMarkers_one_vs_all(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  min_expr_gini_score = 0.5,
  min_det_gini_score = 0.5,
  detection_threshold = 0,
  min_genes = 10,
  verbose = TRUE
)
```

### Arguments

```
gobject      giotto object
expression_values
              gene expression values to use
cluster_column clusters to use
subset_clusters
              selection of clusters to compare
```

```

min_expr_gini_score      filter on minimum gini coefficient on expression
min_det_gini_score       filter on minimum gini coefficient on detection
detection_threshold      detection threshold for gene expression
min_genes                minimum genes to keep per cluster, overrides pval and logFC
verbose                  be verbose

```

**Value**

data.table with marker genes

**See Also**

[findGiniMarkers](#)

**Examples**

```
findGiniMarkers_one_vs_all(gobject)
```

---

findMarkers	<i>findMarkers</i>
-------------	--------------------

---

**Description**

Identify marker genes for selected clusters.

**Usage**

```

findMarkers(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  method = c("scrn", "gini", "mast"),
  subset_clusters = NULL,
  group_1 = NULL,
  group_2 = NULL,
  min_expr_gini_score = 0.5,
  min_det_gini_score = 0.5,
  detection_threshold = 0,
  rank_score = 1,
  group_1_name = NULL,
  group_2_name = NULL,
  adjust_columns = NULL,
  ...
)

```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>cluster_column</code>	clusters to use
<code>method</code>	method to use to detect differentially expressed genes
<code>subset_clusters</code>	selection of clusters to compare
<code>group_1</code>	group 1 cluster IDs from <code>cluster_column</code> for pairwise comparison
<code>group_2</code>	group 2 cluster IDs from <code>cluster_column</code> for pairwise comparison
<code>min_expr_gini_score</code>	gini: filter on minimum gini coefficient for expression
<code>min_det_gini_score</code>	gini: filter minimum gini coefficient for detection
<code>detection_threshold</code>	gini: detection threshold for gene expression
<code>rank_score</code>	gini: rank scores to include
<code>group_1_name</code>	mast: custom name for <code>group_1</code> clusters
<code>group_2_name</code>	mast: custom name for <code>group_2</code> clusters
<code>adjust_columns</code>	mast: column in <code>pDataDT</code> to adjust for (e.g. detection rate)
<code>...</code>	additional parameters for the <code>findMarkers</code> function in <code>scrn</code> or <code>zlm</code> function in <code>MAST</code>

**Details**

Wrapper for all individual functions to detect marker genes for clusters.

**Value**

data.table with marker genes

**See Also**

[findScranMarkers](#), [findGiniMarkers](#) and [findMastMarkers](#)

**Examples**

```
findMarkers(gobject)
```



---

```
findMarkers_one_vs_all
      findMarkers_one_vs_all
```

---

## Description

Identify marker genes for all clusters in a one vs all manner.

## Usage

```
findMarkers_one_vs_all(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  method = c("scrn", "gini", "mast"),
  pval = 0.01,
  logFC = 0.5,
  min_genes = 10,
  min_expr_gini_score = 0.5,
  min_det_gini_score = 0.5,
  detection_threshold = 0,
  rank_score = 1,
  adjust_columns = NULL,
  verbose = TRUE,
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>cluster_column</code>	clusters to use
<code>subset_clusters</code>	selection of clusters to compare
<code>method</code>	method to use to detect differentially expressed genes
<code>pval</code>	scrn & mast: filter on minimal p-value
<code>logFC</code>	scan & mast: filter on logFC
<code>min_genes</code>	minimum genes to keep per cluster, overrides pval and logFC
<code>min_expr_gini_score</code>	gini: filter on minimum gini coefficient for expression
<code>min_det_gini_score</code>	gini: filter minimum gini coefficient for detection
<code>detection_threshold</code>	gini: detection threshold for gene expression
<code>rank_score</code>	gini: rank scores to include
<code>adjust_columns</code>	mast: column in pDataDT to adjust for (e.g. detection rate)

verbose	be verbose
...	additional parameters for the findMarkers function in scan or zlm function in MAST

### Details

Wrapper for all one vs all functions to detect marker genes for clusters.

### Value

data.table with marker genes

### See Also

[findScranMarkers\\_one\\_vs\\_all](#), [findGiniMarkers\\_one\\_vs\\_all](#) and [findMastMarkers\\_one\\_vs\\_all](#)

### Examples

```
findMarkers_one_vs_all(gobject)
```

---

findMastMarkers	<i>findMastMarkers</i>
-----------------	------------------------

---

### Description

Identify marker genes for selected clusters based on the MAST package.

### Usage

```
findMastMarkers(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  group_1 = NULL,
  group_1_name = NULL,
  group_2 = NULL,
  group_2_name = NULL,
  adjust_columns = NULL,
  ...
)
```

### Arguments

gobject	giotto object
expression_values	gene expression values to use
cluster_column	clusters to use
group_1	group 1 cluster IDs from cluster_column for pairwise comparison
group_1_name	custom name for group_1 clusters
group_2	group 2 cluster IDs from cluster_column for pairwise comparison

group\_2\_name      custom name for group\_2 clusters  
 adjust\_columns   column in pDataDT to adjust for (e.g. detection rate)  
 ...                additional parameters for the zlm function in MAST

### Details

This is a minimal convenience wrapper around the [zlm](#) from the MAST package to detect differentially expressed genes.

### Value

data.table with marker genes

### Examples

```
findMastMarkers(gobject)
```

---

```
findMastMarkers_one_vs_all
      findMastMarkers_one_vs_all
```

---

### Description

Identify marker genes for all clusters in a one vs all manner based on the MAST package.

### Usage

```
findMastMarkers_one_vs_all(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  adjust_columns = NULL,
  pval = 0.001,
  logFC = 1,
  min_genes = 10,
  verbose = TRUE,
  ...
)
```

### Arguments

gobject            giotto object  
 expression\_values      gene expression values to use  
 cluster\_column    clusters to use  
 subset\_clusters      selection of clusters to compare  
 adjust\_columns    column in pDataDT to adjust for (e.g. detection rate)  
 pval                filter on minimal p-value

logFC	filter on logFC
min_genes	minimum genes to keep per cluster, overrides pval and logFC
verbose	be verbose
...	additional parameters for the zlm function in MAST

**Value**

data.table with marker genes

**See Also**

[findMastMarkers](#)

**Examples**

```
findMastMarkers_one_vs_all(gobject)
```

---

findScranMarkers	<i>findScranMarkers</i>
------------------	-------------------------

---

**Description**

Identify marker genes for all or selected clusters based on scran's implementation of findMarkers.

**Usage**

```
findScranMarkers(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  group_1 = NULL,
  group_2 = NULL,
  ...
)
```

**Arguments**

gobject	giotto object
expression_values	gene expression values to use
cluster_column	clusters to use
subset_clusters	selection of clusters to compare
group_1	group 1 cluster IDs from cluster_column for pairwise comparison
group_2	group 2 cluster IDs from cluster_column for pairwise comparison
...	additional parameters for the findMarkers function in scran

**Details**

This is a minimal convenience wrapper around the [findMarkers](#) function from the `scrn` package.

To perform differential expression between cluster groups you need to specify cluster IDs to the parameters `group_1` and `group_2`.

**Value**

data.table with marker genes

**Examples**

```
findScranMarkers(gobject)
```

---

```
findScranMarkers_one_vs_all
  findScranMarkers_one_vs_all
```

---

**Description**

Identify marker genes for all clusters in a one vs all manner based on `scrn`'s implementation of `findMarkers`.

**Usage**

```
findScranMarkers_one_vs_all(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  subset_clusters = NULL,
  pval = 0.01,
  logFC = 0.5,
  min_genes = 10,
  verbose = TRUE,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>cluster_column</code>	clusters to use
<code>subset_clusters</code>	subset of clusters to use
<code>pval</code>	filter on minimal p-value
<code>logFC</code>	filter on logFC
<code>min_genes</code>	minimum genes to keep per cluster, overrides <code>pval</code> and <code>logFC</code>
<code>verbose</code>	be verbose
<code>...</code>	additional parameters for the <code>findMarkers</code> function in <code>scrn</code>

**Value**

data.table with marker genes

**See Also**

[findScranMarkers](#)

**Examples**

```
findScranMarkers_one_vs_all(gobject)
```

---

find_grid_2D	<i>find_grid_2D</i>
--------------	---------------------

---

**Description**

find grid location in 2D

**Usage**

```
find_grid_2D(grid_DT, x_loc, y_loc)
```

---

find_grid_3D	<i>find_grid_3D</i>
--------------	---------------------

---

**Description**

find grid location in 3D

**Usage**

```
find_grid_3D(grid_DT, x_loc, y_loc, z_loc)
```

---

find_grid_x	<i>find_grid_x</i>
-------------	--------------------

---

**Description**

find grid location on x-axis

**Usage**

```
find_grid_x(grid_DT, x_loc)
```

---

find_grid_y	<i>find_grid_y</i>
-------------	--------------------

---

**Description**

find grid location on y-axis

**Usage**

find\_grid\_y(grid\_DT, y\_loc)

---

find_grid_z	<i>find_grid_z</i>
-------------	--------------------

---

**Description**

find grid location on z-axis

**Usage**

find\_grid\_z(grid\_DT, z\_loc)

---

fish_function	<i>fish_function</i>
---------------	----------------------

---

**Description**

perform fisher exact test

**Usage**

fish\_function(x\_to, x\_from)

---

fish_function2	<i>fish_function2</i>
----------------	-----------------------

---

**Description**

perform fisher exact test

**Usage**

fish\_function2(A, B, C, D)

---

FSV_show	<i>FSV_show</i>
----------	-----------------

---

**Description**

Visualize spatial variable genes caculated by spatial\_DE

**Usage**

```
FSV_show(  
  results,  
  ms_results = NULL,  
  size = c(4, 2, 1),  
  color = c("blue", "green", "red"),  
  sig_alpha = 0.5,  
  unsig_alpha = 0.5  
)
```

**Arguments**

results	results caculated by spatial_DE
ms_results	ms_results caculated by spatial_DE
size	indicate different levels of qval
color	indicate different SV features
sig_alpha	transparency of significant genes
unsig_alpha	transparency of insignificant genes

**Details**

Description of parameters.

**Value**

nothing

**Examples**

```
FSV_show(results)
```



---

GenePattern_show	<i>GenePattern_show</i>
------------------	-------------------------

---

## Description

Visualize genes distribution patterns calculated by spatial\_AEH

## Usage

```
GenePattern_show(  
  gobject = NULL,  
  AEH_results = NULL,  
  sdimx = NULL,  
  sdimy = NULL,  
  point_size = 3,  
  point_alpha = 1,  
  low_color = "blue",  
  mid_color = "white",  
  high_color = "red",  
  midpoint = 0  
)
```

## Arguments

<code>gobject</code>	giotto object
<code>AEH_results</code>	results from spatial_AEH
<code>sdimx</code>	x axis of spatial locus
<code>sdimy</code>	y axis of spatial locus
<code>point_size</code>	size of points to indicate cells
<code>point_alpha</code>	transparency of points to indicate cells
<code>low_color</code>	color to indicate low score level
<code>mid_color</code>	color to indicate middle score level
<code>high_color</code>	color to indicate high score level
<code>midpoint</code>	point to set mid_color

## Details

Description of parameters.

## Value

nothing

## Examples

```
GenePattern_show(gobject,AEH_results)
```

---

general\_save\_function    *general\_save\_function*

---

## Description

Function to automatically save plots to directory of interest

## Usage

```
general_save_function(
  gobject,
  plot_object,
  save_dir = NULL,
  save_folder = NULL,
  save_name = NULL,
  default_save_name = "giotto_plot",
  save_format = c("png", "tiff", "pdf", "svg"),
  show_saved_plot = F,
  base_width = NULL,
  base_height = NULL,
  base_aspect_ratio = NULL,
  units = NULL,
  dpi = NULL,
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>plot_object</code>	non-ggplot object to plot
<code>save_dir</code>	directory to save to
<code>save_folder</code>	folder in <code>save_dir</code> to save to
<code>save_name</code>	name of plot
<code>save_format</code>	format (e.g. png, tiff, pdf, ...)
<code>show_saved_plot</code>	load & display the saved plot
<code>base_width</code>	width
<code>base_height</code>	height
<code>base_aspect_ratio</code>	aspect ratio
<code>units</code>	units
<code>dpi</code>	Plot resolution

## Examples

```
general_save_function(gobject)
```

---

get10Xmatrix	<i>get10Xmatrix</i>
--------------	---------------------

---

**Description**

This function creates an expression matrix from a 10X structured folder

**Usage**

```
get10Xmatrix(path_to_data)
```

**Arguments**

path\_to\_data    path to the 10X folder

**Details**

A typical 10X folder is named raw\_feature\_bc\_matrix or raw\_feature\_bc\_matrix. It has 3 files:

- barcodes
- features.tsv.gz
- matrix.mtx.gz

**Value**

expression matrix from 10X

**Examples**

```
get10Xmatrix(10Xmatrix)
```

---

getCellProximityGeneScores	<i>getCellProximityGeneScores</i>
----------------------------	-----------------------------------

---

**Description**

Compute cell-cell interaction enrichment (observed vs expected)

**Usage**

```
getCellProximityGeneScores(
  gobject,
  spatial_network_name = "spatial_network",
  cluster_column = "louvain_clus.1",
  selected_genes = NULL,
  expression_values = c("normalized", "scaled", "custom"),
  do_diff_test = TRUE,
  diff_test = c("t.test", "wilcox"),
  minimum_unique_cells = NA,
```

```

    fold_change_addendum = 0.1,
    in_two_directions = TRUE,
    exclude_selected_cells_from_test = F,
    verbose = T
)

```

### Arguments

<code>gobject</code>	giotto object
<code>spatial_network_name</code>	name of spatial network to use
<code>cluster_column</code>	name of column to use for clusters
<code>selected_genes</code>	selection of genes to perform calculations for
<code>expression_values</code>	expression values to use
<code>do_diff_test</code>	perform differential test
<code>diff_test</code>	which differential expression test
<code>minimum_unique_cells</code>	minimum number of cells needed to proceed
<code>fold_change_addendum</code>	constant to add when calculating log2 fold-change
<code>in_two_directions</code>	shows enrichment in both directions: cell1-cell2, cell2-cell1
<code>exclude_selected_cells_from_test</code>	exclude certain cells from test
<code>verbose</code>	verbose

### Details

Function to calculate if genes are differentially expressed in cell types when they interact (according to physical proximity) with other cell types. The results `data.table` contains the following columns:

- `genes`: All or selected list of tested genes
- `cell_expr_1`: average gene expression in cell type 1 from unified\_int cell-cell interaction
- `cell_expr_2`: average gene expression in cell type 2 from unified\_int cell-cell interaction
- `comb_expr`: combined average gene expression in cell type 1 and 2 from unified\_int cell-cell interaction
- `all_cell_expr_1`: average gene expression for all cells from cell type 1
- `all_cell_expr_2`: average gene expression for all cells from cell type 2
- `all_comb_expr`: combined average gene expression for all cells from cell type 1 and 2
- `pval_1`: p-value from test between interacting cells and all cells from cell type 1
- `pval_2`: p-value from test between interacting cells and all cells from cell type 2
- `cell_type_1`: first cell type of cell-cell interaction
- `cell_type_2`: second cell type of cell-cell interaction
- `interaction`: the cell-cell interaction, based on physical proximity
- `nr_1`: number of cell type 1 in the unified cell-cell interaction
- `nr_2`: number of cell type 2 in the unified cell-cell interaction

- all\_nr\_1: number of all cell type 1 in the whole dataset
- all\_nr\_2: number of all cell type 2 in the whole dataset
- diff\_spat: difference between comb\_expr and all\_comb\_expr
- diff\_spat\_1: difference between cell\_expr\_1 and all\_cell\_expr\_1
- diff\_spat\_2: difference between cell\_expr\_1 and all\_cell\_expr\_1
- log2fc\_spat\_1: fold-change of diff\_spat\_1
- log2fc\_spat\_2: fold-change of diff\_spat\_2
- log2fc\_spat: fold-change of diff\_spat
- type\_int: type of interaction
- unified\_int: interaction with alphabetically sorted cell type 1 and cell type 2
- unif\_int\_rank: 1 or 2
- fdr\_1: fdr from test between interacting cells and all cells from cell type 1
- fdr\_2: fdr from test between interacting cells and all cells from cell type 2

### Value

Cell Proximity Gene scores (CPGscores) in data.table format

### Examples

```
getCellProximityGeneScores(gobject)
```

---

```
getClusterSimilarity    getClusterSimilarity
```

---

### Description

Creates data.table with pairwise correlation scores between each cluster.

### Usage

```
getClusterSimilarity(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  cor = c("pearson", "spearman")
)
```

### Arguments

gobject	giotto object
expression_values	expression values to use
cluster_column	name of column to use for clusters
cor	correlation score to calculate distance

**Details**

Creates data.table with pairwise correlation scores between each cluster and the group size (# of cells) for each cluster. This information can be used together with mergeClusters to combine very similar or small clusters into bigger clusters.

**Value**

data.table

**Examples**

```
getClusterSimilarity(gobject)
```

---

getDendrogramSplits	<i>getDendrogramSplits</i>
---------------------	----------------------------

---

**Description**

Split dendrogram at each node and keep the leave (label) information..

**Usage**

```
getDendrogramSplits(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  cluster_column,  
  cor = c("pearson", "spearman"),  
  distance = "ward.D",  
  h = NULL,  
  h_color = "red",  
  show_dend = TRUE,  
  verbose = TRUE  
)
```

**Arguments**

gobject	giotto object
expression_values	expression values to use
cluster_column	name of column to use for clusters
cor	correlation score to calculate distance
distance	distance method to use for hierarchical clustering
h	height of horizontal lines to plot
h_color	color of horizontal lines
show_dend	show dendrogram
verbose	be verbose

**Details**

Creates a data.table with three columns and each row represents a node in the dendrogram. For each node the height of the node is given together with the two subdendrograms. This information can be used to determine in a hierarchical manner differentially expressed marker genes at each node.

**Value**

data.table object

**Examples**

```
getDendrogramSplits(gobject)
```

---

getDistinctColors	<i>getDistinctColors</i>
-------------------	--------------------------

---

**Description**

Returns a number of distinct colors based on the RGB scale

**Usage**

```
getDistinctColors(n)
```

**Arguments**

n	number of colors wanted
---	-------------------------

**Value**

number of distinct colors

---

getGeneToGeneScores	<i>getGeneToGeneScores</i>
---------------------	----------------------------

---

**Description**

Compute gene-gene enrichment scores.

**Usage**

```
getGeneToGeneScores(  
  CPGscore,  
  selected_genes = NULL,  
  specific_genes_1 = NULL,  
  specific_genes_2 = NULL,  
  min_cells = 5,  
  min_fdr = 0.05,  
  min_spat_diff = 0.2,  
  min_log2_fc = 0.5,
```

```
direction = c("both", "up", "down"),
fold_change_addendum = 0.1,
verbose = TRUE
)
```

**Arguments**

- CPGscore CPGscore, output from getCellProximityGeneScores()
- selected\_genes select subset of genes
- specific\_genes\_1 specific source genes (see details)
- specific\_genes\_2 specific target genes (see details)
- min\_cells min number of cells threshold
- min\_spat\_diff spatial difference threshold
- min\_log2\_fc log2 fold-change threshold
- direction up or downregulation or both
- fold\_change\_addendum constant to add when calculating log2 fold-change
- verbose verbose
- min\_pval p-value threshold

**Details**

This converts the single gene cell proximityscores into pairwise combinations of genes, which allows you to determine if 2 genes are differentially expressed in interacting cell types.

**Value**

Gene to gene scores in data.table format

**Examples**

```
getGeneToGeneScores(CPGscore)
```

---

```
get_cell_to_cell_sorted_name_conversion
get_cell_to_cell_sorted_name_conversion
```

---

**Description**

creates unified cell-cell interaction names

**Usage**

```
get_cell_to_cell_sorted_name_conversion(all_cell_types)
```

**Examples**

```
get_cell_to_cell_sorted_name_conversion()
```



---

```
get_interaction_gene_enrichment  
    get_interaction_gene_enrichment
```

---

**Description**

Computes gene enrichment between all interactions

**Usage**

```
get_interaction_gene_enrichment(  
  spatial_network,  
  unified_int_col = "unified_int",  
  source_col = "source_clus",  
  source_IDs = "from",  
  neighb_col = "neighb_clus",  
  neighb_IDs = "to",  
  expression_matrix,  
  cell_annotation,  
  annotation_ID = "uniq_ID",  
  cell_type_col,  
  do_diff_test = T,  
  diff_test = c("t.test", "wilcox"),  
  minimum_unique_cells = NA,  
  exclude_selected_cells_from_test = T,  
  verbose = T  
)
```

**Examples**

```
get_interaction_gene_enrichment()
```

---

```
get_specific_interaction_gene_enrichment  
    get_specific_interaction_gene_enrichment
```

---

**Description**

Computes gene enrichment between specified interaction

**Usage**

```
get_specific_interaction_gene_enrichment(  
  sub_spatial_network,  
  source_col = "source_clus",  
  source_IDs = "from",  
  neighb_col = "neighb_clus",  
  neighb_IDs = "to",  
  expression_matrix,
```

```

interaction_name = "to_specify",
cell_annotation,
annotation_ID = "uniq_ID",
cell_type_col,
do_diff_test = T,
diff_test = c("t.test", "wilcox"),
minimum_unique_cells = NA,
exclude_selected_cells_from_test = T
)

```

### Examples

```
get_specific_interaction_gene_enrichment()
```

---

```
ggplot_save_function  ggplot_save_function
```

---

### Description

Function to automatically save plots to directory of interest

### Usage

```

ggplot_save_function(
  gobject,
  plot_object,
  save_dir = NULL,
  save_folder = NULL,
  save_name = NULL,
  default_save_name = "giotto_plot",
  save_format = NULL,
  show_saved_plot = F,
  ncol = 1,
  nrow = 1,
  scale = 1,
  base_width = NULL,
  base_height = NULL,
  base_aspect_ratio = NULL,
  units = NULL,
  dpi = NULL,
  limitsize = TRUE,
  ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_object</code>	ggplot object to plot
<code>save_dir</code>	directory to save to
<code>save_folder</code>	folder in <code>save_dir</code> to save to

save_name	name of plot
save_format	format (e.g. png, tiff, pdf, ...)
show_saved_plot	load & display the saved plot
ncol	number of columns
nrow	number of rows
scale	scale
base_width	width
base_height	height
base_aspect_ratio	aspect ratio
units	units
dpi	Plot resolution
limitsize	When TRUE (the default), ggsave will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.

**See Also**

[cowplot::save\\_plot](#)

**Examples**

```
ggplot_save_function(gobject)
```

---

giotto-class	<i>S4 giotto Class</i>
--------------	------------------------

---

**Description**

Framework of giotto object to store and work with spatial expression data

**Slots**

raw_exprs	raw expression counts
norm_expr	normalized expression counts
norm_scaled_expr	normalized and scaled expression counts
custom_expr	custom normalized counts
spatial_locs	spatial location coordinates for cells
cell_metadata	metadata for cells
gene_metadata	metadata for genes
cell_ID	unique cell IDs
gene_ID	unique gene IDs
spatial_network	spatial network in data.table/data.frame format
spatial_grid	spatial grid in data.table/data.frame format
dimension_reduction	slot to save dimension reduction coordinates

nn\_network nearest neighbor network in igraph format  
 parameters slot to save parameters that have been used  
 instructions slot for global function instructions  
 offset\_file offset file used to stitch together image fields  
 OS\_platform Operating System to run Giotto analysis on

---

hyperGeometricEnrich    *hyperGeometricEnrich*

---

## Description

Function to calculate gene signature enrichment scores per spatial position using a hypergeometric test.

## Usage

```
hyperGeometricEnrich(
  gobject,
  sign_matrix,
  expression_values = c("normalized", "scaled", "custom"),
  reverse_log_scale = TRUE,
  logbase = 2,
  output_enrichment = c("original", "zscore")
)
```

## Arguments

gobject	Giotto object
sign_matrix	Matrix of signature genes for each cell type / process
expression_values	expression values to use
reverse_log_scale	reverse expression values from log scale
logbase	log base to use if reverse_log_scale = TRUE
output_enrichment	how to return enrichment output

## Details

The enrichment score is calculated based on the p-value from the hypergeometric test,  $-\log_{10}(\text{p-value})$ .

## Value

data.table with enrichment results

## Examples

```
hyperGeometricEnrich(gobject)
```

---

kmeans_binarize	<i>kmeans_binarize</i>
-----------------	------------------------

---

**Description**

create binarized scores using kmeans

**Usage**

```
kmeans_binarize(x, nstart = 3, iter.max = 10)
```

---

loadHMRF	<i>loadHMRF</i>
----------	-----------------

---

**Description**

load previous HMRF

**Usage**

```
loadHMRF(  
  name_used = "test",  
  output_folder_used,  
  k_used = 10,  
  betas_used,  
  python_path_used  
)
```

**Arguments**

name_used	name of HMRF that was run
output_folder_used	output folder that was used
k_used	number of HMRF domains that was tested
betas_used	betas that were tested
python_path_used	python path that was used

**Details**

Description of HMRF parameters ...

**Value**

reloads a previous ran HMRF from doHRMF

**Examples**

```
loadHMRF(gobject)
```

---

makeSignMatrixPAGE	<i>makeSignMatrixPAGE</i>
--------------------	---------------------------

---

**Description**

Function to convert list of signature genes (e.g. for cell types or processes) into a binary matrix format that can be used with the PAGE enrichment option.

**Usage**

```
makeSignMatrixPAGE(sign_names, sign_list)
```

**Arguments**

- sign\_names      vector with names for each provided gene signature
- sign\_list      list of genes (signature)

**Value**

matrix

**See Also**

[PAGEEnrich](#)

**Examples**

```
makeSignMatrixPAGE()
```

---

makeSignMatrixRank	<i>makeSignMatrixRank</i>
--------------------	---------------------------

---

**Description**

Function to convert a single-cell count matrix and a corresponding single-cell cluster vector into a rank matrix that can be used with the Rank enrichment option.

**Usage**

```
makeSignMatrixRank(sc_matrix, sc_cluster_ids, gobject = NULL)
```

**Arguments**

- sign\_names      vector with names for each provided gene signature
- sign\_list      list of genes (signature)

**Value**

matrix

**See Also**[rankEnrich](#)**Examples**

```
makeSignMatrixRank()
```

---

```
make_simulated_network
```

```
make_simulated_network
```

---

**Description**

Simulate random network.

**Usage**

```
make_simulated_network(  
  gobject,  
  spatial_network_name = "spatial_network",  
  cluster_column,  
  number_of_simulations = 100  
)
```

**Examples**

```
make_simulated_network(gobject)
```

---

```
mergeClusters
```

```
mergeClusters
```

---

**Description**

Merge selected clusters based on pairwise correlation scores and size of cluster.

**Usage**

```
mergeClusters(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  cluster_column,  
  cor = c("pearson", "spearman"),  
  new_cluster_name = "merged_cluster",  
  min_cor_score = 0.8,  
  max_group_size = 20,  
  force_min_group_size = 10,  
  return_gobject = TRUE,  
  verbose = TRUE  
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>cluster_column</code>	name of column to use for clusters
<code>cor</code>	correlation score to calculate distance
<code>new_cluster_name</code>	new name for merged clusters
<code>min_cor_score</code>	min correlation score to merge pairwise clusters
<code>max_group_size</code>	max cluster size that can be merged
<code>force_min_group_size</code>	size of clusters that will be merged with their most similar neighbor(s)
<code>return_gobject</code>	return giotto object
<code>verbose</code>	be verbose

**Details**

Merge selected clusters based on pairwise correlation scores and size of cluster. To avoid large clusters to merge the `max_group_size` can be lowered. Small clusters can be forcibly merged with their most similar pairwise cluster by adjusting the `force_min_group_size` parameter. Clusters smaller than this value will be merged independent on the provided `min_cor_score` value.

A giotto object is returned by default, if FALSE then the merging vector will be returned.

**Value**

Giotto object

**Examples**

```
mergeClusters(gobject)
```

---

mygini\_fun

*mygini\_fun*


---

**Description**

calculate gini coefficient

**Usage**

```
mygini_fun(x, weights = rep(1, length(x)))
```

**Value**

gini coefficient



---

nnDT\_to\_kNN

*nnDT\_to\_kNN*


---

**Description**

Convert a nearest network data.table to a kNN object

**Usage**

```
nnDT_to_kNN(nnDT)
```

**Arguments**

nnDT                      nearest neighbor network in data.table format

**Value**

kNN object

---

node\_clusters

*node\_clusters*


---

**Description**

Merge selected clusters based on pairwise correlation scores and size of cluster.

**Usage**

```
node_clusters(hclus_obj, verbose = TRUE)
```

**Arguments**

hclus\_obj                hclus object  
 verbose                be verbose

**Value**

list of splitted dendrogram nodes from high to low node height

**Examples**

```
node_clusters(hclus_obj)
```

---

normalizeGiotto	<i>normalizeGiotto</i>
-----------------	------------------------

---

## Description

normalize and/or scale expression values of Giotto object

## Usage

```
normalizeGiotto(
  gobject,
  norm_methods = c("standard", "osmFISH"),
  library_size_norm = TRUE,
  scalefactor = 6000,
  log_norm = TRUE,
  logbase = 2,
  scale_genes = T,
  scale_cells = T,
  scale_order = c("first_genes", "first_cells"),
  verbose = F
)
```

## Arguments

gobject	giotto object
norm_methods	normalization method to use
library_size_norm	normalize cells by library size
scalefactor	scale factor to use after library size normalization
log_norm	transform values to log-scale
logbase	log base to use to log normalize expression values
scale_genes	z-score genes over all cells
scale_cells	z-score cells over all genes
scale_order	order to scale genes and cells
verbose	be verbose

## Details

Currently there are two 'methods' to normalize your raw counts data.

A. The standard method follows the standard protocol which can be adjusted using the provided parameters and follows the following order:

- 1. Data normalization for total library size and scaling by a custom scale-factor.
- 2. Log transformation of data.
- 3. Z-scoring of data by genes and/or cells.

B. The normalization method as provided by the osmFISH paper is also implemented:

- 1. First normalize genes, for each gene divide the counts by the total gene count and multiply by the total number of genes.
- 2. Next normalize cells, for each cell divide the normalized gene counts by the total counts per cell and multiply by the total number of cells.

This data will be saved in the Giotto slot for custom expression.

### Value

giotto object

### Examples

```
normalizeGiotto(gobject)
```

---

OR_function2	<i>OR_function2</i>
--------------	---------------------

---

### Description

calculate odds-ratio

### Usage

```
OR_function2(A, B, C, D)
```

---

PAGEEnrich	<i>PAGEEnrich</i>
------------	-------------------

---

### Description

Function to calculate gene signature enrichment scores per spatial position using PAGE.

### Usage

```
PAGEEnrich(
  gobject,
  sign_matrix,
  expression_values = c("normalized", "scaled", "custom"),
  reverse_log_scale = TRUE,
  logbase = 2,
  output_enrichment = c("original", "zscore")
)
```

Arguments

- gobject            Giotto object
- sign\_matrix      Matrix of signature genes for each cell type / process
- expression\_values            expression values to use
- reverse\_log\_scale            reverse expression values from log scale
- logbase           log base to use if reverse\_log\_scale = TRUE
- output\_enrichment            how to return enrichment output

Details

The enrichment Z score is calculated by using method (PAGE) from Kim SY et al., BMC bioinformatics, 2005 as  $Z = ((Sm^\mu * m^{(1/2)})/\delta)$ . For each gene in each spot, mu is the fold change values versus the mean expression and delta is the standard deviation. Sm is the mean fold change value of a specific marker gene set and m is the size of a given marker gene set.

Value

data.table with enrichment results

Examples

```
PAGEEnrich(gobject)
```

---

pDataDT	<i>pDataDT</i>
---------	----------------

---

Description

show cell metadata

Usage

```
pDataDT(gobject)
```

Arguments

- gobject            giotto object

Value

data.table with cell metadata

Examples

```
pDataDT(gobject)
```

---

plotCPGscores	<i>plotCPGscores</i>
---------------	----------------------

---

## Description

Create heatmap from cell-cell proximity scores

## Usage

```
plotCPGscores(
  CPGscores,
  selected_interactions = NULL,
  selected_genes = NULL,
  detail_plot = T,
  simple_plot = F,
  simple_plot_facet = c("interaction", "genes"),
  facet_scales = "fixed",
  facet_ncol = length(selected_genes),
  facet_nrow = length(selected_interactions),
  show_plot = F
)
```

## Arguments

CPGscores	CPGscores, output from <code>getCellProximityGeneScores()</code>
selected_interactions	interactions to show
selected_genes	genes to show
detail_plot	show detailed info in both interacting cell types
simple_plot	show a simplified plot
simple_plot_facet	facet on interactions or genes with simple plot
facet_scales	ggplot facet scales paramter
facet_ncol	ggplot facet ncol parameter
facet_nrow	ggplot facet nrow parameter
show_plot	show plot

## Details

Give more details ...

## Value

ggplot barplot

## Examples

```
plotCPGscores(CPGscores)
```

---

plotGTGscores	<i>plotGTGscores</i>
---------------	----------------------

---

## Description

Create heatmap from cell-cell proximity scores

## Usage

```
plotGTGscores(
  gobject,
  GTGscore,
  selected_interactions = NULL,
  selected_gene_to_gene = NULL,
  detail_plot = T,
  simple_plot = F,
  simple_plot_facet = c("interaction", "genes"),
  facet_scales = "fixed",
  facet_ncol = length(selected_gene_to_gene),
  facet_nrow = length(selected_interactions),
  colors = c("blue", "red"),
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "plotGTGscores"
)
```

## Arguments

<code>gobject</code>	giotto object
<code>GTGscore</code>	GTGscore, output from <code>getGeneToGeneScores()</code>
<code>selected_interactions</code>	interactions to show
<code>detail_plot</code>	show detailed info in both interacting cell types
<code>simple_plot</code>	show a simplified plot
<code>simple_plot_facet</code>	facet on interactions or genes with simple plot
<code>facet_scales</code>	ggplot facet scales paramter
<code>facet_ncol</code>	ggplot facet ncol parameter
<code>facet_nrow</code>	ggplot facet nrow parameter
<code>colors</code>	vector with 2 colors to represent respectively all and selected cells
<code>show_plot</code>	show plots
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>selected_genes</code>	genes to show

**Details**

Give more details ...

**Value**

ggplot barplot

**Examples**

```
plotGTGscores(GTGscore)
```

---

plotHeatmap

*plotHeatmap*


---

**Description**

Creates heatmap for genes and clusters.

**Usage**

```
plotHeatmap(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  cluster_column = NULL,
  cluster_order = c("size", "correlation", "custom"),
  cluster_custom_order = NULL,
  cluster_color_code = NULL,
  cluster_cor_method = "pearson",
  cluster_hclust_method = "ward.D",
  gene_order = c("custom", "correlation"),
  gene_custom_order = NULL,
  gene_cor_method = "pearson",
  gene_hclust_method = "complete",
  show_values = c("rescaled", "z-scaled", "original"),
  size_vertical_lines = 1.1,
  gradient_colors = c("blue", "yellow", "red"),
  gene_label_selection = NULL,
  axis_text_y_size = NULL,
  legend_nrows = 1,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "plotHeatmap"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>genes</code>	genes to use
<code>cluster_column</code>	name of column to use for clusters
<code>cluster_order</code>	method to determine cluster order
<code>cluster_custom_order</code>	custom order for clusters
<code>cluster_color_code</code>	color code for clusters
<code>cluster_cor_method</code>	method for cluster correlation
<code>cluster_hclust_method</code>	method for hierarchical clustering of clusters
<code>gene_order</code>	method to determine gene order
<code>gene_custom_order</code>	custom order for genes
<code>gene_cor_method</code>	method for gene correlation
<code>gene_hclust_method</code>	method for hierarchical clustering of genes
<code>show_values</code>	which values to show on heatmap
<code>size_vertical_lines</code>	sizes for vertical lines
<code>gradient_colors</code>	colors for heatmap gradient
<code>gene_label_selection</code>	subset of genes to show on y-axis
<code>axis_text_y_size</code>	size for y-axis text
<code>legend_nrows</code>	number of rows for the cluster legend
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name

**Details**

If you want to display many genes there are 2 ways to proceed:

- 1. set `axis_text_y_size` to a really small value and show all genes
- 2. provide a subset of genes to display to `gene_label_selection`



## Value

ggplot

## Examples

```
plotHeatmap(gobject)
```

---

plotly_axis_scale_2D	<i>plotly_axis_scale_2D</i>
----------------------	-----------------------------

---

## Description

adjust the axis scale in 3D plotly plot

## Usage

```
plotly_axis_scale_2D(
  cell_locations,
  sdimx = NULL,
  sdimy = NULL,
  mode = c("cube", "real", "custom"),
  custom_ratio = NULL
)
```

## Arguments

cell_locations	spatial_loc in giotto object
sdimx	x axis of cell spatial location
sdimy	y axis of cell spatial location
mode	axis adjustment mode
custom_ratio	set the ratio artificially

## Value

edges in spatial grid as data.table()

## Examples

```
plotly_axis_scale_2D(gobject)
```

---

plotly_axis_scale_3D	<i>plotly_axis_scale_3D</i>
----------------------	-----------------------------

---

**Description**

adjust the axis scale in 3D plotly plot

**Usage**

```
plotly_axis_scale_3D(
  cell_locations,
  sdimx = NULL,
  sdimy = NULL,
  sdimz = NULL,
  mode = c("cube", "real", "custom"),
  custom_ratio = NULL
)
```

**Arguments**

cell_locations	spatial_loc in giotto object
sdimx	x axis of cell spatial location
sdimy	y axis of cell spatial location
sdimz	z axis of cell spatial location
mode	axis adjustment mode
custom_ratio	set the ratio artificially

**Value**

edges in spatial grid as data.table()

**Examples**

```
plotly_axis_scale_3D(gobject)
```

---

plotly_grid	<i>plotly_grid</i>
-------------	--------------------

---

**Description**

provide grid segment to draw in plot\_ly()

**Usage**

```
plotly_grid(
  spatial_grid,
  x_start = "x_start",
  y_start = "y_start",
  x_end = "x_end",
  y_end = "y_end"
)
```

**Arguments**

spatial\_grid      spatial\_grid in giotto object

**Value**

edges in spatial grid as data.table()

**Examples**

```
plotly_grid(gobject)
```

---

plotly_network	<i>plotly_network</i>
----------------	-----------------------

---

**Description**

provide network segment to draw in 3D plot\_ly()

**Usage**

```
plotly_network(  
  network,  
  x = "sdimx_begin",  
  y = "sdimy_begin",  
  z = "sdimz_begin",  
  x_end = "sdimx_end",  
  y_end = "sdimy_end",  
  z_end = "sdimz_end"  
)
```

**Arguments**

gobject              network in giotto object

**Value**

edges in network as data.table()

**Examples**

```
plotly_network(gobject)
```

---

plotMetaDataCellsHeatmap

*plotMetaDataCellsHeatmap*


---

## Description

Creates heatmap for numeric cell metadata within aggregated clusters.

## Usage

```
plotMetaDataCellsHeatmap(
  gobject,
  metadata_cols = NULL,
  spat_enr_names = NULL,
  value_cols = NULL,
  first_meta_col = NULL,
  second_meta_col = NULL,
  show_values = c("zscores", "original", "zscores_rescaled"),
  custom_cluster_order = NULL,
  clus_cor_method = "pearson",
  clus_cluster_method = "complete",
  custom_values_order = NULL,
  values_cor_method = "pearson",
  values_cluster_method = "complete",
  midpoint = 0,
  x_text_size = 8,
  x_text_angle = 45,
  y_text_size = 8,
  strip_text_size = 8,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "plotMetaDataCellsHeatmap"
)
```

## Arguments

gobject	giotto object
metadata_cols	annotation columns found in pDataDT(gobject)
spat_enr_names	spatial enrichment results to include
value_cols	value columns to use
first_meta_col	if more than 1 metadata column, select the x-axis factor
second_meta_col	if more than 1 metadata column, select the facetting factor
show_values	which values to show on heatmap
custom_cluster_order	custom cluster order (default = NULL)

clus_cor_method	correlation method for clusters
clus_cluster_method	hierarchical cluster method for the clusters
midpoint	midpoint of show_values
x_text_size	size of x-axis text
x_text_angle	angle of x-axis text
y_text_size	size of y-axis text
strip_text_size	size of strip text
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
custom_gene_order	custom gene order (default = NULL)
gene_cor_method	correlation method for genes
gene_cluster_method	hierarchical cluster method for the genes

## Details

Creates heatmap for the average values of selected value columns in the different annotation groups.

## Value

ggplot or data.table

## See Also

[plotMetaDataHeatmap](#) for gene expression instead of numeric cell annotation data.

## Examples

```
plotMetaDataCellsHeatmap(gobject)
```

---

plotMetaDataHeatmap	<i>plotMetaDataHeatmap</i>
---------------------	----------------------------

---

## Description

Creates heatmap for genes within aggregated clusters.

## Usage

```
plotMetaDataHeatmap(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  metadata_cols = NULL,
  selected_genes = NULL,
  first_meta_col = NULL,
  second_meta_col = NULL,
  show_values = c("zscores", "original", "zscores_rescaled"),
  custom_cluster_order = NULL,
  clus_cor_method = "pearson",
  clus_cluster_method = "complete",
  custom_gene_order = NULL,
  gene_cor_method = "pearson",
  gene_cluster_method = "complete",
  midpoint = 0,
  x_text_size = 10,
  x_text_angle = 45,
  y_text_size = 10,
  strip_text_size = 8,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "plotMetaDataHeatmap"
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>metadata_cols</code>	annotation columns found in <code>pDataDT(gobject)</code>
<code>selected_genes</code>	subset of genes to use
<code>first_meta_col</code>	if more than 1 metadata column, select the x-axis factor
<code>second_meta_col</code>	if more than 1 metadata column, select the facetting factor
<code>show_values</code>	which values to show on heatmap
<code>custom_cluster_order</code>	custom cluster order (default = NULL)

clus_cor_method	correlation method for clusters
clus_cluster_method	hierarchical cluster method for the clusters
custom_gene_order	custom gene order (default = NULL)
gene_cor_method	correlation method for genes
gene_cluster_method	hierarchical cluster method for the genes
midpoint	midpoint of show_values
x_text_size	size of x-axis text
x_text_angle	angle of x-axis text
y_text_size	size of y-axis text
strip_text_size	size of strip text
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name

## Details

Creates heatmap for the average expression of selected genes in the different annotation/cluster groups

## Value

ggplot or data.table

## See Also

[plotMetaDataCellsHeatmap](#) for numeric cell annotation instead of gene expression.

## Examples

```
plotMetaDataHeatmap(gobject)
```

plotPCA

*plotPCA***Description**

Short wrapper for PCA visualization

**Usage**

```
plotPCA(gobject, dim_reduction_name = "pca", default_save_name = "PCA", ...)
```

**Arguments**

gobject	giotto object
dim_reduction_name	dimension reduction name
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column
group_by_subset	subset the group_by factor column
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
spat_enr_names	names of spatial enrichment results to include
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells



other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
title	title for plot, defaults to cell_color parameter
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

## Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotPCA\\_3D](#)

## Value

ggplot

## Examples

```
plotPCA(gobject)
```

---

plotPCA\_2D

*plotPCA\_2D*


---

## Description

Short wrapper for PCA visualization

## Usage

```
plotPCA_2D(
  gobject,
  dim_reduction_name = "pca",
  default_save_name = "PCA_2D",
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_name</code>	dimension reduction name
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>groub_by</code>	create multiple plots based on cell annotation column
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter

<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>title</code>	title for plot, defaults to <code>cell_color</code> parameter
<code>show_legend</code>	show legend
<code>legend_text</code>	size of legend text
<code>axis_text</code>	size of axis text
<code>axis_title</code>	size of axis title
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>

### Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotPCA\\_3D](#)

### Value

ggplot

### Examples

```
plotPCA_2D(gobject)
```

plotPCA\_3D

*plotPCA\_3D***Description**

Visualize cells according to 3D PCA dimension reduction

**Usage**

```
plotPCA_3D(
  gobject,
  dim_reduction_name = "pca",
  default_save_name = "PCA_3D",
  ...
)
```

**Arguments**

gobject	giotto object
dim_reduction_name	pca dimension reduction name
default_save_name	default save name for saving, ideally change save_name in save_param
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
dim3_to_use	dimension to use on z-axis
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters

show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

**Details**

Description of parameters.

**Value**

plotly

**Examples**

```
plotPCA_3D(gobject)
```

---

plotTSNE	<i>plotTSNE</i>
----------	-----------------

---

**Description**

Short wrapper for tSNE visualization

**Usage**

```
plotTSNE(gobject, dim_reduction_name = "tsne", default_save_name = "tSNE", ...)
```

**Arguments**

gobject	giotto object
dim_reduction_name	dimension reduction name
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column
group_by_subset	subset the group_by factor column
dim1_to_use	dimension to use on x-axis

dim2_to_use	dimension to use on y-axis
spat_enr_names	names of spatial enrichment results to include
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
title	title for plot, defaults to cell_color parameter
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title

cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

### Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotTSNE\\_3D](#)

### Value

ggplot

### Examples

```
plotTSNE(gobject)
```

---

plotTSNE\_2D

*plotTSNE\_2D*


---

### Description

Short wrapper for tSNE visualization

### Usage

```
plotTSNE_2D(
  gobject,
  dim_reduction_name = "tsne",
  default_save_name = "tSNE_2D",
  ...
)
```

### Arguments

gobject	giotto object
dim_reduction_name	dimension reduction name
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column
group_by_subset	subset the group_by factor column
dim1_to_use	dimension to use on x-axis

dim2_to_use	dimension to use on y-axis
spat_enr_names	names of spatial enrichment results to include
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
title	title for plot, defaults to cell_color parameter
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title



cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

### Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotTSNE\\_3D](#)

### Value

ggplot

### Examples

```
plotTSNE_2D(gobject)
```

---

plotTSNE_3D	<i>plotTSNE_3D</i>
-------------	--------------------

---

### Description

Visualize cells according to dimension reduction coordinates

### Usage

```
plotTSNE_3D(
  gobject,
  dim_reduction_name = "tsne",
  default_save_name = "TSNE_3D",
  ...
)
```

### Arguments

gobject	giotto object
dim_reduction_name	tsne dimension reduction name
default_save_name	default save name for saving, don't change, change save_name in save_param
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
dim3_to_use	dimension to use on z-axis
show_NN_network	show underlying NN network

nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

### Details

Description of parameters.

### Value

plotly

### Examples

```
plotTSNE_3D(gobject)
```

---

plotUMAP

*plotUMAP*


---

**Description**

Short wrapper for UMAP visualization

**Usage**

```
plotUMAP(gobject, dim_reduction_name = "umap", default_save_name = "UMAP", ...)
```

**Arguments**

gobject	giotto object
dim_reduction_name	dimension reduction name
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column
group_by_subset	subset the group_by factor column
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
spat_enr_names	names of spatial enrichment results to include
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells

other_point_size	size of not selected cells
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
title	title for plot, defaults to cell_color parameter
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

## Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotUMAP\\_3D](#)

## Value

ggplot

## Examples

```
plotUMAP(gobject)
```

---

plotUMAP_2D	<i>plotUMAP_2D</i>
-------------	--------------------

---

## Description

Short wrapper for UMAP visualization

## Usage

```
plotUMAP_2D(
  gobject,
  dim_reduction_name = "umap",
  default_save_name = "UMAP_2D",
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_name</code>	dimension reduction name
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>groub_by</code>	create multiple plots based on cell annotation column
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter

<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>title</code>	title for plot, defaults to <code>cell_color</code> parameter
<code>show_legend</code>	show legend
<code>legend_text</code>	size of legend text
<code>axis_text</code>	size of axis text
<code>axis_title</code>	size of axis title
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>

### Details

Description of parameters, see [dimPlot2D](#). For 3D plots see [plotUMAP\\_3D](#)

### Value

ggplot

### Examples

```
plotUMAP_2D(gobject)
```

---

plotUMAP\_3D

*plotUMAP\_3D*


---

## Description

Visualize cells according to dimension reduction coordinates

## Usage

```
plotUMAP_3D(
  gobject,
  dim_reduction_name = "umap",
  default_save_name = "UMAP_3D",
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_name</code>	umap dimension reduction name
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network</code> = TRUE
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters

show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()

**Details**

Description of parameters.

**Value**

plotly

**Examples**

```
plotUMAP_3D(gobject)
```

---

```
plot_network_layer_ggplot
      plot_network_layer_ggplot
```

---

**Description**

Visualize cells in network layer according to dimension reduction coordinates

**Usage**

```
plot_network_layer_ggplot(
  gobject,
  annotated_network_DT,
  edge_alpha = NULL,
  show_legend = T
)
```

**Arguments**

annotated_network_DT	annotated network data.table of selected cells
edge_alpha	alpha of network edges
show_legend	show legend
gobject	giotto object



**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
plot_network_layer_ggplot(gobject)
```

---

```
plot_point_layer_ggplot
```

```
plot_point_layer_ggplot
```

---

**Description**

Visualize cells in point layer according to dimension reduction coordinates

**Usage**

```
plot_point_layer_ggplot(
  ggobject,
  annotated_DT_selected,
  annotated_DT_other,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = 0,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  edge_alpha = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_legend = T
)
```

**Arguments**

<code>annotated_DT_selected</code>	annotated data.table of selected cells
<code>annotated_DT_other</code>	annotated data.table of not selected cells
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_legend</code>	show legend
<code>gobject</code>	giotto object

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
plot_point_layer_ggplot(gobject)
```

---

```
plot_spat_point_layer_ggplot
      plot_spat_point_layer_ggplot
```

---

**Description**

creat ggplot point layer for spatial coordinates

**Usage**

```
plot_spat_point_layer_ggplot(
  gobject,
  sdimx = NULL,
  sdimy = NULL,
  cell_locations_metadata_selected,
  cell_locations_metadata_other,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  point_size = 2,
  point_border_col = "lightgrey",
  point_border_stroke = 0.1,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 1,
  show_legend = TRUE
)
```

**Arguments**

sdimx	x-axis dimension name (default = 'sdimx')
sdimy	y-axis dimension name (default = 'sdimy')
cell_locations_metadata_selected	annotated location from selected cells

<code>cell_locations_metadata_other</code>	annotated location from non-selected cells
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color for not selected cells
<code>other_point_size</code>	point size for not selected cells
<code>show_legend</code>	show legend
<code>gobject</code>	giotto object

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
plot_spat_point_layer_ggplot(gobject)
```

---

print.giotto	<i>print method for giotto class</i>
--------------	--------------------------------------

---

### Description

print method for giotto class. Prints the chosen number of genes (rows) and cells (columns) from the raw count matrix. Also print the spatial locations for the chosen number of cells.

### Usage

```
print.giotto(object, ...)
```

### Arguments

nr_genes	number of genes (rows) to print
nr_cells	number of cells (columns) to print

---

rankEnrich	<i>rankEnrich</i>
------------	-------------------

---

### Description

Function to calculate gene signature enrichment scores per spatial position using a rank based approach.

### Usage

```
rankEnrich(
  gobject,
  sign_matrix,
  expression_values = c("normalized", "scaled", "custom"),
  reverse_log_scale = TRUE,
  logbase = 2,
  output_enrichment = c("original", "zscore")
)
```

### Arguments

gobject	Giotto object
sign_matrix	Matrix of signature genes for each cell type / process
expression_values	expression values to use
reverse_log_scale	reverse expression values from log scale
logbase	log base to use if reverse_log_scale = TRUE
output_enrichment	how to return enrichment output

**Details**

First a new rank is calculated as  $R = (R1 * R2)^{(1/2)}$ , where R1 is the rank of fold-change for each gene in each spot and R2 is the rank of each marker in each cell type. The Rank-Biased Precision is then calculated as:  $RBP = (1 - 0.99) * (0.99)^{(R - 1)}$  and the final enrichment score is then calculated as the sum of top 100 RBPs.

**Value**

data.table with enrichment results

**Examples**

```
rankEnrich(gobject)
```

---

rank_binarize	<i>rank_binarize</i>
---------------	----------------------

---

**Description**

create binarized scores using arbitrary rank of top genes

**Usage**

```
rank_binarize(x, max_rank = 200)
```

---

readGiottoInstructions	<i>readGiottoInstructions</i>
------------------------	-------------------------------

---

**Description**

Retrieves the instruction associated with the provided parameter

**Usage**

```
readGiottoInstructions(giotto_instructions, param = NULL)
```

**Arguments**

giotto_instructions	
	giotto object or result from createGiottoInstructions()
param	parameter to retrieve

**Value**

specific parameter

**Examples**

```
readGiottoInstructions()
```

---

removeCellAnnotation	<i>removeCellAnnotation</i>
----------------------	-----------------------------

---

**Description**

removes cell annotation of giotto object

**Usage**

```
removeCellAnnotation(gobject, columns = NULL, return_gobject = TRUE)
```

**Arguments**

gobject	giotto object
columns	names of columns to remove
return_gobject	boolean: return giotto object (default = TRUE)

**Details**

if return\_gobject = FALSE, it will return the cell metadata

**Value**

giotto object

**Examples**

```
removeCellAnnotation(gobject)
```

---

removeGeneAnnotation	<i>removeGeneAnnotation</i>
----------------------	-----------------------------

---

**Description**

removes gene annotation of giotto object

**Usage**

```
removeGeneAnnotation(gobject, columns = NULL, return_gobject = TRUE)
```

**Arguments**

gobject	giotto object
columns	names of columns to remove
return_gobject	boolean: return giotto object (default = TRUE)

**Details**

if return\_gobject = FALSE, it will return the gene metadata

**Value**

giotto object

**Examples**

```
removeGeneAnnotation(gobject)
```

---

```
replaceGiottoInstructions
      replaceGiottoInstructions
```

---

**Description**

Function to replace all instructions from giotto object

**Usage**

```
replaceGiottoInstructions(gobject, instructions = NULL)
```

**Arguments**

<code>gobject</code>	giotto object
<code>instructions</code>	new instructions (e.g. result from <code>createGiottoInstructions</code> )

**Value**

named vector with giotto instructions

**Examples**

```
replaceGiottoInstructions()
```

---

<code>runPCA</code>	<i>runPCA</i>
---------------------	---------------

---

**Description**

runs a Principal Component Analysis

**Usage**

```
runPCA(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  reduction = c("cells", "genes"),
  name = "pca",
  genes_to_use = NULL,
  return_gobject = TRUE,
  scale_unit = F,
  ncp = 200,
  ...
)
```



**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>reduction</code>	cells or genes
<code>name</code>	arbitrary name for PCA run
<code>genes_to_use</code>	subset of genes to use for PCA
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>scale_unit</code>	scale features before PCA
<code>ncp</code>	number of principal components to calculate
<code>...</code>	additional parameters for PCA (see details)

**Details**

See [PCA](#) for more information about other parameters.

**Value**

giotto object with updated PCA dimension reduction

**Examples**

```
runPCA(gobject)
```

---

runtSNE

*runtSNE*


---

**Description**

run tSNE

**Usage**

```
runtSNE(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  reduction = c("cells", "genes"),
  dim_reduction_to_use = "pca",
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  name = "tsne",
  genes_to_use = NULL,
  return_gobject = TRUE,
  dims = 2,
  perplexity = 30,
  theta = 0.5,
  do_PCA_first = F,
  set_seed = T,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>reduction</code>	cells or genes
<code>dim_reduction_to_use</code>	use another dimension reduction set as input
<code>dim_reduction_name</code>	name of dimension reduction set to use
<code>dimensions_to_use</code>	number of dimensions to use as input
<code>name</code>	arbitrary name for tSNE run
<code>genes_to_use</code>	if <code>dim_reduction_to_use = NULL</code> , which genes to use
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>dims</code>	tSNE param: number of dimensions to return
<code>perplexity</code>	tSNE param: perplexity
<code>theta</code>	tSNE param: theta
<code>do_PCA_first</code>	tSNE param: do PCA before tSNE (default = FALSE)
<code>set_seed</code>	use of seed
<code>seed_number</code>	seed number to use
<code>...</code>	additional tSNE parameters

**Details**

See [Rtsne](#) for more information about these and other parameters.

- Input for tSNE dimension reduction can be another dimension reduction (default = 'pca')
- To use gene expression as input set `dim_reduction_to_use = NULL`
- multiple tSNE results can be stored by changing the *name* of the analysis

**Value**

giotto object with updated tSNE dimension reduction

**Examples**

```
runtSNE(gobject)
```

runUMAP

*runUMAP***Description**

run UMAP

**Usage**

```
runUMAP(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  reduction = c("cells", "genes"),
  dim_reduction_to_use = "pca",
  dim_reduction_name = "pca",
  dimensions_to_use = 1:10,
  name = "umap",
  genes_to_use = NULL,
  return_gobject = TRUE,
  n_neighbors = 40,
  n_components = 2,
  n_epochs = 400,
  min_dist = 0.01,
  n_threads = 1,
  spread = 5,
  set_seed = T,
  seed_number = 1234,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>reduction</code>	cells or genes
<code>dim_reduction_to_use</code>	use another dimension reduction set as input
<code>dim_reduction_name</code>	name of dimension reduction set to use
<code>dimensions_to_use</code>	number of dimensions to use as input
<code>name</code>	arbitrary name for UMAP run
<code>genes_to_use</code>	if <code>dim_reduction_to_use = NULL</code> , which genes to use
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>n_neighbors</code>	UMAP param: number of neighbors
<code>n_components</code>	UMAP param: number of components
<code>n_epochs</code>	UMAP param: number of epochs

min_dist	UMAP param: minimum distance
n_threads	UMAP param: threads to use
spread	UMAP param: spread
set_seed	use of seed
seed_number	seed number to use
...	additional UMAP parameters

### Details

See [umap](#) for more information about these and other parameters.

- Input for UMAP dimension reduction can be another dimension reduction (default = 'pca')
- To use gene expression as input set `dim_reduction_to_use = NULL`
- multiple UMAP results can be stored by changing the *name* of the analysis

### Value

giotto object with updated UMAP dimension reduction

### Examples

```
runUMAP(gobject)
```

---

selectPatternGenes	<i>selectPatternGenes</i>
--------------------	---------------------------

---

### Description

Select genes correlated with spatial patterns

### Usage

```
selectPatternGenes(
  spatPatObj,
  dimensions = 1:5,
  top_pos_genes = 10,
  top_neg_genes = 10,
  min_pos_cor = 0.5,
  min_neg_cor = -0.5,
  return_top_selection = FALSE
)
```

### Arguments

spatPatObj	Output from detectSpatialPatterns
dimensions	dimensions to identify correlated genes for.
top_pos_genes	Top positively correlated genes.
top_neg_genes	Top negatively correlated genes.
min_pos_cor	Minimum positive correlation score to include a gene.
min_neg_cor	Minimum negative correlation score to include a gene.

**Details**

Description.

**Value**

Data.table with genes associated with selected dimension (PC).

**Examples**

```
selectPatternGenes(gobject)
```

---

```
select_expression_values  
      select_expression_values
```

---

**Description**

helper function to select expression values

**Usage**

```
select_expression_values(gobject, values)
```

**Arguments**

gobject	giotto object
values	expression values to extract

**Value**

expression matrix

---

```
show,giotto-method      show method for giotto class
```

---

**Description**

show method for giotto class

**Usage**

```
## S4 method for signature 'giotto'  
show(object)
```

---

showClusterDendrogram *showClusterDendrogram*


---

## Description

Creates dendrogram for selected clusters.

## Usage

```
showClusterDendrogram(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  cluster_column,
  cor = c("pearson", "spearman"),
  distance = "ward.D",
  h = NULL,
  h_color = "red",
  rotate = FALSE,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "showClusterDendrogram",
  ...
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>cluster_column</code>	name of column to use for clusters
<code>cor</code>	correlation score to calculate distance
<code>distance</code>	distance method to use for hierarchical clustering
<code>h</code>	height of horizontal lines to plot
<code>h_color</code>	color of horizontal lines
<code>rotate</code>	rotate dendrogram 90 degrees
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>...</code>	additional parameters for <code>ggdendrogram()</code>

## Details

Expression correlation dendrogram for selected clusters.

**Value**

ggplot

**Examples**

```
showClusterDendrogram(gobject)
```

---

showClusterHeatmap	<i>showClusterHeatmap</i>
--------------------	---------------------------

---

**Description**

Creates heatmap based on identified clusters

**Usage**

```
showClusterHeatmap(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = "all",
  cluster_column,
  cor = c("pearson", "spearman"),
  distance = "ward.D",
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "showClusterHeatmap",
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	expression values to use
<code>genes</code>	vector of genes to use, default to 'all'
<code>cluster_column</code>	name of column to use for clusters
<code>cor</code>	correlation score to calculate distance
<code>distance</code>	distance method to use for hierarchical clustering
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>...</code>	additional parameters for the Heatmap function from <code>ComplexHeatmap</code>

Details

Correlation heatmap of selected clusters.

Value

ggplot

Examples

```
showClusterHeatmap(gobject)
```

---

showCPGscores	<i>showCPGscores</i>
---------------	----------------------

---

Description

visualize Cell Proximity Gene enrichment scores

Usage

```
showCPGscores(  
  gobject,  
  CPGscore,  
  method = c("volcano", "cell_barplot", "cell-cell", "cell_sankey"),  
  min_cells = 5,  
  min_fdr = 0.05,  
  min_spat_diff = 0.2,  
  min_log2_fc = 0.5,  
  keep_int_duplicates = TRUE,  
  direction = c("both", "up", "down"),  
  cell_color_code = NULL,  
  show_plot = NA,  
  return_plot = NA,  
  save_plot = NA,  
  save_param = list(),  
  default_save_name = "showCPGscores"  
)
```

Arguments

CPGscore	CPGscore, output from getCellProximityGeneScores()
method	visualization method
min_cells	min number of cells threshold
min_fdr	fdr threshold
min_spat_diff	spatial difference threshold
min_log2_fc	min log2 fold-change
keep_int_duplicates	keep both cell_A-cell_B and cell_B-cell_A
direction	up or downregulation or both



cell_color_code	color code for cell types
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

**Details**

Give more details ...

**Value**

Gene to gene scores in data.table format

**Examples**

```
showCPGscores(CPGscore)
```

---

```
showGeneExpressionProximityScore
```

```
showGeneExpressionProximityScore
```

---

**Description**

Create heatmap from cell-cell proximity scores

**Usage**

```
showGeneExpressionProximityScore(
  scores,
  selected_gene,
  sort_column = "diff_spat"
)
```

**Arguments**

scores	CPscore, output from getAverageCellProximityGeneScores()
selected_gene	gene to show
sort_column	column name to use for sorting

**Details**

Give more details ...

**Value**

ggplot barplot

**Examples**

```
showGeneExpressionProximityScore(scores)
```

---

showGiottoInstructions	<i>showGiottoInstructions</i>
------------------------	-------------------------------

---

**Description**

Function to display all instructions from giotto object

**Usage**

```
showGiottoInstructions(gobject)
```

**Arguments**

gobject                  giotto object

**Value**

named vector with giotto instructions

**Examples**

```
showGiottoInstructions()
```

---

showGTGscores	<i>showGTGscores</i>
---------------	----------------------

---

**Description**

visualize Cell Proximity Gene enrichment scores

**Usage**

```
showGTGscores(  
  GTGscore,  
  method = c("cell_barplot", "cell-cell", "cell_sankey"),  
  min_cells = 5,  
  min_pval = 0.05,  
  min_spat_diff = 0.2,  
  min_log2_fc = 0.5,  
  direction = c("both", "up", "down"),  
  cell_color_code = NULL,  
  show_plot = T,  
  specific_genes_1 = NULL,  
  specific_genes_2 = NULL,  
  first_cell_name = "ligand cell",  
  second_cell_name = "receptor cell",  
  return_DT = F  
)
```

**Arguments**

method	visualization method
min_cells	min number of cells threshold
min_pval	p-value threshold
min_spat_diff	spatial difference threshold
min_log2_fc	log2 fold-change threshold
direction	up or downregulation or both
cell_color_code	color code for cell types
show_plot	print plot
specific_genes_1	subset of genes, matched with specific_genes_2
specific_genes_2	subset of genes, matched with specific_genes_1
first_cell_name	name for first cells
second_cell_name	name for second cells
CPGscore	CPGscore, output from getCellProximityGeneScores()

**Details**

Give more details ...

**Value**

ggplot

**Examples**

```
showGTGscores(CPGscore)
```

---

```
showIntExpressionProximityScore
```

```
showIntExpressionProximityScore
```

---

**Description**

Create heatmap from cell-cell proximity scores

**Usage**

```
showIntExpressionProximityScore(
  scores,
  selected_interaction,
  sort_column = "diff_spat",
  show_enriched_n = 5,
  show_depleted_n = 5
)
```

Arguments

scores	scores, output from getAverageCellProximityGeneScores()
selected_interaction	interaction to show
sort_column	column name to use for sorting
show_enriched_n	show top enriched interactions
show_depleted_n	show top depleted interactions

Details

Give more details ...

Value

ggplot barplot

Examples

showIntExpressionProximityScore(scores)

---

showPattern	<i>showPattern</i>
-------------	--------------------

---

Description

create a spatial grid  
show patterns for 2D spatial data

Usage

showPattern(gobject, spatPatObj, ...)  
  
showPattern(gobject, spatPatObj, ...)

Arguments

gobject	giotto object
spatPatObj	Output from detectSpatialPatterns
dimension	dimension to plot
trim	Trim ends of the PC values.
background_color	background color for plot
grid_border_color	color for grid
show_legend	show legend of ggplot
show_plot	show plot

return\_plot      return ggplot object  
save\_plot        directly save the plot [boolean]  
save\_param       list of saving parameters from all\_plots\_save\_function()  
default\_save\_name  
                  default save name for saving, don't change, change save\_name in save\_param

**Details**

Description.

**Value**

ggplot  
ggplot

**See Also**

[showPattern2D](#)

**Examples**

```
showPattern(gobject)  
showPattern(gobject)
```

---

showPattern2D	<i>showPattern2D</i>
---------------	----------------------

---

**Description**

show patterns for 2D spatial data

**Usage**

```
showPattern2D(  
  gobject,  
  spatPatObj,  
  dimension = 1,  
  trim = c(0.02, 0.98),  
  background_color = "white",  
  grid_border_color = "grey",  
  show_legend = T,  
  point_size = 1,  
  show_plot = NA,  
  return_plot = NA,  
  save_plot = NA,  
  save_param = list(),  
  default_save_name = "showPattern2D"  
)
```

Arguments

<code>gobject</code>	giotto object
<code>spatPatObj</code>	Output from <code>detectSpatialPatterns</code>
<code>dimension</code>	dimension to plot
<code>trim</code>	Trim ends of the PC values.
<code>background_color</code>	background color for plot
<code>grid_border_color</code>	color for grid
<code>show_legend</code>	show legend of ggplot
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>

Value

ggplot

Examples

`showPattern2D(gobject)`

---

<code>showPattern3D</code>	<i>showPattern3D</i>
----------------------------	----------------------

---

Description

show patterns for 3D spatial data

Usage

```
showPattern3D(  
  gobject,  
  spatPatObj,  
  dimension = 1,  
  trim = c(0.02, 0.98),  
  background_color = "white",  
  grid_border_color = "grey",  
  show_legend = T,  
  point_size = 1,  
  axis_scale = c("cube", "real", "custom"),  
  custom_ratio = NULL,  
  x_ticks = NULL,  
  y_ticks = NULL,  
  z_ticks = NULL,
```

```

    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "showPattern3D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>spatPatObj</code>	Output from <code>detectSpatialPatterns</code>
<code>dimension</code>	dimension to plot
<code>trim</code>	Trim ends of the PC values.
<code>background_color</code>	background color for plot
<code>grid_border_color</code>	color for grid
<code>show_legend</code>	show legend of plot
<code>point_size</code>	adjust the point size
<code>axis_scale</code>	scale the axis
<code>custom_ratio</code>	customize the scale of the axis
<code>x_ticks</code>	the tick number of <code>x_axis</code>
<code>y_ticks</code>	the tick number of <code>y_axis</code>
<code>z_ticks</code>	the tick number of <code>z_axis</code>
<code>show_plot</code>	show plot
<code>return_plot</code>	return plot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>

### Value

plotly

### Examples

```
showPattern3D(gobject)
```

---

showPatternGenes	<i>showPatternGenes</i>
------------------	-------------------------

---

## Description

show genes correlated with spatial patterns

## Usage

```
showPatternGenes(
  gobject,
  spatPatObj,
  dimension = 1,
  top_pos_genes = 5,
  top_neg_genes = 5,
  point_size = 1,
  return_DT = FALSE,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "showPatternGenes"
)
```

## Arguments

gobject	giotto object
spatPatObj	Output from detectSpatialPatterns
dimension	dimension to plot genes for.
top_pos_genes	Top positively correlated genes.
top_neg_genes	Top negatively correlated genes.
point_size	size of points
return_DT	if TRUE, it will return the data.table used to generate the plots
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Value

ggplot

## Examples

```
showPatternGenes(gobject)
```



---

showProcessingSteps	<i>showProcessingSteps</i>
---------------------	----------------------------

---

**Description**

shows the sequential processing steps that were performed in a summarized format

**Usage**

```
showProcessingSteps(gobject)
```

**Arguments**

gobject	giotto object
---------	---------------

**Value**

list of processing steps and names

**Examples**

```
showProcessingSteps(gobject)
```

---

showTopGeneToGene	<i>showTopGeneToGene</i>
-------------------	--------------------------

---

**Description**

Show enriched/depleted gene-gene enrichments

**Usage**

```
showTopGeneToGene(
  GTGscore,
  top_interactions = 10,
  direction = c("increased", "decreased"),
  complement_data = T,
  subset_cell_ints = NULL,
  subset_genes = NULL
)
```

**Arguments**

GTGscore	GTGscore, output from getGeneToGeneScores()
top_interactions	number of top gene-gene enrichments to show
direction	show top increased or decreased gene-gene enrichments
complement_data	include non-enriched gene-gene scores from other cell-cell interactions
subset_cell_ints	subset cell-cell interactions to show
subset_genes	subset genes to show

Details

Give more details ...

Value

ggplot barplot

Examples

showTopGeneToGene(scores)

---

signPCA	<i>signPCA</i>
---------	----------------

---

Description

identify significant prinicipal components (PCs)

Usage

```
signPCA(  
  gobject,  
  method = c("screeplot", "jackstraw"),  
  expression_values = c("normalized", "scaled", "custom"),  
  reduction = c("cells", "genes"),  
  genes_to_use = NULL,  
  scale_unit = T,  
  ncp = 50,  
  scree_labels = T,  
  scree_ylim = c(0, 10),  
  jack_iter = 10,  
  jack_threshold = 0.01,  
  jack_verbose = T,  
  show_plot = NA,  
  return_plot = NA,  
  save_plot = NA,  
  save_param = list(),  
  default_save_name = "signPCA",  
  ...  
)
```

Arguments

gobject	giotto object
method	method to use to identify significant PCs
expression_values	expression values to use
reduction	cells or genes
genes_to_use	subset of genes to use for PCA
scale_unit	scale features before PCA

ncp	number of principal components to calculate
scree_labels	show labels on scree plot
scree_ylim	y-axis limits on scree plot
jack_iter	number of iterations for jackstraw
jack_threshold	p-value threshold to call a PC significant
jack_verbose	show progress of jackstraw method
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
...	additional parameters for PCA

### Details

Two different methods can be used to assess the number of relevant or significant principal components (PC's).

1. Screeplot works by plotting the explained variance of each individual PC in a barplot allowing you to identify which PC does not show a significant contribution anymore (= 'elbow method').

2. The Jackstraw method uses the [permutationPA](#) function. By systematically permuting genes it identifies robust, and thus significant, PCs.

multiple PCA results can be stored by changing the *name* parameter

### Value

ggplot object for scree method and maxtrix of p-values for jackstraw

### Examples

```
signPCA(gobject)
```

---

spatCellPlot	<i>spatCellPlot</i>
--------------	---------------------

---

### Description

Visualize cells according to spatial coordinates

### Usage

```
spatCellPlot(
  gobject,
  sdimx = "sdimx",
  sdimy = "sdimy",
  spat_enr_names = NULL,
  cell_annotation_values,
  cell_color_gradient = c("blue", "white", "red"),
```

```

gradient_midpoint = NULL,
gradient_limits = NULL,
select_cell_groups = NULL,
select_cells = NULL,
point_size = 3,
point_border_col = "black",
point_border_stroke = 0.1,
show_cluster_center = F,
show_center_label = F,
center_point_size = 4,
center_point_border_col = "black",
center_point_border_stroke = 0.1,
label_size = 4,
label_fontface = "bold",
show_network = F,
spatial_network_name = "spatial_network",
network_color = NULL,
network_alpha = 1,
show_grid = F,
spatial_grid_name = "spatial_grid",
grid_color = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 1,
other_cells_alpha = 0.1,
coord_fix_ratio = NULL,
show_legend = T,
legend_text = 8,
axis_text = 8,
axis_title = 8,
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spatCellPlot"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_annotation_values</code>	numeric cell annotation columns
<code>cell_color_gradient</code>	vector with 3 colors for numeric data

gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
show_network	show underlying spatial network
spatial_network_name	name of spatial network to use
network_color	color of spatial network
network_alpha	alpha of spatial network
show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	point size of not selected cells
other_cells_alpha	alpha of not selected cells
coord_fix_ratio	fix ratio between x and y-axis
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

spatCellPlot(gobject)

---

spatCellPlot2D	<i>spatCellPlot2D</i>
----------------	-----------------------

---

**Description**

Visualize cells according to spatial coordinates

**Usage**

```
spatCellPlot2D(  
  gobject,  
  sdimx = "sdimx",  
  sdimy = "sdimy",  
  spat_enr_names = NULL,  
  cell_annotation_values,  
  cell_color_gradient = c("blue", "white", "red"),  
  gradient_midpoint = NULL,  
  gradient_limits = NULL,  
  select_cell_groups = NULL,  
  select_cells = NULL,  
  point_size = 3,  
  point_border_col = "black",  
  point_border_stroke = 0.1,  
  show_cluster_center = F,  
  show_center_label = F,  
  center_point_size = 4,  
  center_point_border_col = "black",  
  center_point_border_stroke = 0.1,  
  label_size = 4,  
  label_fontface = "bold",  
  show_network = F,  
  spatial_network_name = "spatial_network",  
  network_color = NULL,  
  network_alpha = 1,  
  show_grid = F,  
  spatial_grid_name = "spatial_grid",  
  grid_color = NULL,  
  show_other_cells = T,  
  other_cell_color = "lightgrey",  
  other_point_size = 1,
```

```

    other_cells_alpha = 0.1,
    coord_fix_ratio = NULL,
    show_legend = T,
    legend_text = 8,
    axis_text = 8,
    axis_title = 8,
    cow_n_col = 2,
    cow_rel_h = 1,
    cow_rel_w = 1,
    cow_align = "h",
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatCellPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_annotation_values</code>	numeric cell annotation columns
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>show_network</code>	show underlying spatial network

spatial_network_name	name of spatial network to use
network_color	color of spatial network
network_alpha	alpha of spatial network
show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	point size of not selected cells
other_cells_alpha	alpha of not selected cells
coord_fix_ratio	fix ratio between x and y-axis
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## Examples

```
spatCellPlot2D(gobject)
```



---

spatDimCellPlot	<i>spatDimCellPlot</i>
-----------------	------------------------

---

## Description

Visualize numerical features of cells according to spatial AND dimension reduction coordinates in 2D

## Usage

```
spatDimCellPlot(
  gobject,
  plot_alignment = c("vertical", "horizontal"),
  spat_enr_names = NULL,
  cell_annotation_values,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  sdimx = "sdimx",
  sdimy = "sdimy",
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  dim_point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  dim_show_cluster_center = F,
  dim_show_center_label = T,
  dim_center_point_size = 4,
  dim_center_point_border_col = "black",
  dim_center_point_border_stroke = 0.1,
  dim_label_size = 4,
  dim_label_fontface = "bold",
  spat_show_cluster_center = F,
  spat_show_center_label = F,
  spat_center_point_size = 4,
  spat_center_point_border_col = "black",
  spat_center_point_border_stroke = 0.1,
  spat_label_size = 4,
  spat_label_fontface = "bold",
  show_NN_network = F,
  nn_network_to_use = "sNN",
  nn_network_name = "sNN.pca",
  dim_edge_alpha = 0.5,
  spat_show_network = F,
```

```

    spatial_network_name = "spatial_network",
    spat_network_color = "red",
    spat_network_alpha = 0.5,
    spat_show_grid = F,
    spatial_grid_name = "spatial_grid",
    spat_grid_color = "green",
    show_other_cells = TRUE,
    other_cell_color = "grey",
    dim_other_point_size = 0.5,
    spat_other_point_size = 0.5,
    spat_other_cells_alpha = 0.5,
    coord_fix_ratio = NULL,
    cow_n_col = 2,
    cow_rel_h = 1,
    cow_rel_w = 1,
    cow_align = "h",
    show_legend = T,
    legend_text = 8,
    axis_text = 8,
    axis_title = 8,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatDimCellPlot"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_annotation_values</code>	numeric cell annotation columns
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>sdimx</code>	= spatial dimension to use on x-axis
<code>sdimy</code>	= spatial dimension to use on y-axis
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter

```

select_cells    select subset of cells based on cell IDs
dim_point_size  size of points in dim. reduction space
dim_point_border_col
                border color of points in dim. reduction space
dim_point_border_stroke
                border stroke of points in dim. reduction space
spat_point_size
                size of spatial points
spat_point_border_col
                border color of spatial points
spat_point_border_stroke
                border stroke of spatial points
dim_show_cluster_center
                show the center of each cluster
dim_show_center_label
                provide a label for each cluster
dim_center_point_size
                size of the center point
dim_center_point_border_col
                border color of center point
dim_center_point_border_stroke
                stroke size of center point
dim_label_size  size of the center label
dim_label_fontface
                font of the center label
spat_show_cluster_center
                show the center of each cluster
spat_show_center_label
                provide a label for each cluster
spat_center_point_size
                size of the center point
spat_label_size
                size of the center label
spat_label_fontface
                font of the center label
show_NN_network
                show underlying NN network
nn_network_to_use
                type of NN network to use (kNN vs sNN)
nn_network_name
                name of NN network to use, if show_NN_network = TRUE
dim_edge_alpha  column to use for alpha of the edges
spat_show_network
                show spatial network
spatial_network_name
                name of spatial network to use
spat_network_color
                color of spatial network

```

spat_show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
spat_grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
dim_other_point_size	size of not selected dim cells
spat_other_point_size	size of not selected spat cells
spat_other_cells_alpha	alpha of not selected spat cells
coord_fix_ratio	ratio for coordinates
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## Examples

```
spatDimCellPlot(gobject)
```

---

spatDimCellPlot2D	<i>spatDimCellPlot2D</i>
-------------------	--------------------------

---

## Description

Visualize numerical features of cells according to spatial AND dimension reduction coordinates in 2D

## Usage

```
spatDimCellPlot2D(
  gobject,
  plot_alignment = c("vertical", "horizontal"),
  spat_enr_names = NULL,
  cell_annotation_values,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  sdimx = "sdimx",
  sdimy = "sdimy",
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  dim_point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  dim_show_cluster_center = F,
  dim_show_center_label = T,
  dim_center_point_size = 4,
  dim_center_point_border_col = "black",
  dim_center_point_border_stroke = 0.1,
  dim_label_size = 4,
  dim_label_fontface = "bold",
  spat_show_cluster_center = F,
  spat_show_center_label = F,
  spat_center_point_size = 4,
  spat_center_point_border_col = "black",
  spat_center_point_border_stroke = 0.1,
  spat_label_size = 4,
  spat_label_fontface = "bold",
  show_NN_network = F,
  nn_network_to_use = "sNN",
  nn_network_name = "sNN.pca",
  dim_edge_alpha = 0.5,
  spat_show_network = F,
```

```

    spatial_network_name = "spatial_network",
    spat_network_color = "red",
    spat_network_alpha = 0.5,
    spat_show_grid = F,
    spatial_grid_name = "spatial_grid",
    spat_grid_color = "green",
    show_other_cells = TRUE,
    other_cell_color = "grey",
    dim_other_point_size = 0.5,
    spat_other_point_size = 0.5,
    spat_other_cells_alpha = 0.5,
    coord_fix_ratio = NULL,
    cow_n_col = 2,
    cow_rel_h = 1,
    cow_rel_w = 1,
    cow_align = "h",
    show_legend = T,
    legend_text = 8,
    axis_text = 8,
    axis_title = 8,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatDimCellPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_annotation_values</code>	numeric cell annotation columns
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>sdimx</code>	= spatial dimension to use on x-axis
<code>sdimy</code>	= spatial dimension to use on y-axis
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter

```

select_cells    select subset of cells based on cell IDs
dim_point_size  size of points in dim. reduction space
dim_point_border_col
                border color of points in dim. reduction space
dim_point_border_stroke
                border stroke of points in dim. reduction space
spat_point_size
                size of spatial points
spat_point_border_col
                border color of spatial points
spat_point_border_stroke
                border stroke of spatial points
dim_show_cluster_center
                show the center of each cluster
dim_show_center_label
                provide a label for each cluster
dim_center_point_size
                size of the center point
dim_center_point_border_col
                border color of center point
dim_center_point_border_stroke
                stroke size of center point
dim_label_size  size of the center label
dim_label_fontface
                font of the center label
spat_show_cluster_center
                show the center of each cluster
spat_show_center_label
                provide a label for each cluster
spat_center_point_size
                size of the center point
spat_label_size
                size of the center label
spat_label_fontface
                font of the center label
show_NN_network
                show underlying NN network
nn_network_to_use
                type of NN network to use (kNN vs sNN)
nn_network_name
                name of NN network to use, if show_NN_network = TRUE
dim_edge_alpha  column to use for alpha of the edges
spat_show_network
                show spatial network
spatial_network_name
                name of spatial network to use
spat_network_color
                color of spatial network

```

spat_show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
spat_grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
dim_other_point_size	size of not selected dim cells
spat_other_point_size	size of not selected spat cells
spat_other_cells_alpha	alpha of not selected spat cells
coord_fix_ratio	ratio for coordinates
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## Examples

```
spatDimCellPlot2D(gobject)
```



---

spatDimGenePlot	<i>spatDimGenePlot</i>
-----------------	------------------------

---

## Description

Visualize cells according to spatial AND dimension reduction coordinates in ggplot mode

## Usage

```
spatDimGenePlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  plot_alignment = c("vertical", "horizontal"),
  genes,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  show_NN_network = F,
  show_spatial_network = F,
  show_spatial_grid = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  edge_alpha_dim = NULL,
  scale_alpha_with_expression = FALSE,
  spatial_network_name = "spatial_network",
  spatial_grid_name = "spatial_grid",
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  midpoint = 0,
  genes_high_color = "red",
  genes_mid_color = "white",
  genes_low_color = "blue",
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_legend = T,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "spatDimGenePlot"
)
```

## Arguments

gobject	giotto object
---------	---------------

expression_values	gene expression values to use
plot_alignment	direction to align plot
genes	genes to show
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimension reduction name
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
point_size	size of point (cell)
dim_point_border_col	color of border around points
dim_point_border_stroke	stroke size of border around points
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
edge_alpha_dim	dim reduction plot: column to use for alpha of the edges
scale_alpha_with_expression	scale expression with ggplot alpha parameter
spatial_network_name	name of spatial network to use
spatial_grid_name	name of spatial grid to use
spat_point_size	spatial plot: point size
spat_point_border_col	color of border around points
spat_point_border_stroke	stroke size of border around points
midpoint	size of point (cell)
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
dim_point_size	dim reduction plot: point size

## Details

Description of parameters.

## Value

ggplot

## See Also

[spatDimGenePlot3D](#)

## Examples

```
spatDimGenePlot(gobject)
```

---

spatDimGenePlot2D	<i>spatDimGenePlot2D</i>
-------------------	--------------------------

---

## Description

Visualize cells according to spatial AND dimension reduction coordinates in ggplot mode

## Usage

```
spatDimGenePlot2D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  plot_alignment = c("vertical", "horizontal"),
  genes,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  show_NN_network = F,
  show_spatial_network = F,
  show_spatial_grid = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  edge_alpha_dim = NULL,
  scale_alpha_with_expression = FALSE,
  spatial_network_name = "spatial_network",
  spatial_grid_name = "spatial_grid",
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  midpoint = 0,
  genes_high_color = "red",
  genes_mid_color = "white",
```

```

genes_low_color = "blue",
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_legend = T,
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spatDimGenePlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>plot_alignment</code>	direction to align plot
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>point_size</code>	size of point (cell)
<code>dim_point_border_col</code>	color of border around points
<code>dim_point_border_stroke</code>	stroke size of border around points
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha_dim</code>	dim reduction plot: column to use for alpha of the edges
<code>scale_alpha_with_expression</code>	scale expression with ggplot alpha parameter
<code>spatial_network_name</code>	name of spatial network to use
<code>spatial_grid_name</code>	name of spatial grid to use
<code>spat_point_size</code>	spatial plot: point size
<code>spat_point_border_col</code>	color of border around points
<code>spat_point_border_stroke</code>	stroke size of border around points

midpoint	size of point (cell)
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
dim_point_size	dim reduction plot: point size

Details

Description of parameters.

Value

ggplot

See Also

[spatDimGenePlot3D](#)

Examples

spatDimGenePlot2D(gobject)

---

spatDimGenePlot3D	<i>spatDimGenePlot3D</i>
-------------------	--------------------------

---

Description

Visualize cells according to spatial AND dimension reduction coordinates in ggplot mode

Usage

```
spatDimGenePlot3D(  
  gobject,  
  expression_values = c("normalized", "scaled", "custom"),  
  plot_alignment = c("horizontal", "vertical"),  
  dim_reduction_to_use = "umap",  
  dim_reduction_name = "umap",  
  dim1_to_use = 1,  
  dim2_to_use = 2,  
  dim3_to_use = NULL,  
  sdimx = "sdimx",
```

```

sdimy = "sdimy",
sdimz = "sdimz",
genes,
cluster_column = NULL,
select_cell_groups = NULL,
select_cells = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 1.5,
show_NN_network = F,
nn_network_to_use = "sNN",
network_name = "sNN.pca",
label_size = 16,
genes_low_color = "blue",
genes_mid_color = "white",
genes_high_color = "red",
dim_point_size = 3,
nn_network_alpha = 0.5,
show_spatial_network = F,
spatial_network_name = "spatial_network",
network_color = "lightgray",
spatial_network_alpha = 0.5,
show_spatial_grid = F,
spatial_grid_name = "spatial_grid",
spatial_grid_color = NULL,
spatial_grid_alpha = 0.5,
spatial_point_size = 3,
legend_text_size = 12,
axis_scale = c("cube", "real", "custom"),
custom_ratio = NULL,
x_ticks = NULL,
y_ticks = NULL,
z_ticks = NULL,
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spatDimGenePlot3D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis

dim3_to_use	dimension to use on z-axis
genes	genes to show
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
dim_point_size	dim reduction plot: point size
spatial_network_name	name of spatial network to use
spatial_grid_name	name of spatial grid to use
spatial_point_size	spatial plot: point size
show_plot	show plots
return_plot	return plotly object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
edge_alpha_dim	dim reduction plot: column to use for alpha of the edges
scale_alpha_with_expression	scale expression with ggplot alpha parameter
point_size	size of point (cell)
show_legend	show legend

## Details

Description of parameters.

## Value

plotly

## Examples

```
spatDimGenePlot3D(gobject)
```

---

spatDimPlot

*spatDimPlot*


---

## Description

Visualize cells according to spatial AND dimension reduction coordinates 2D

## Usage

```
spatDimPlot(
  gobject,
  plot_alignment = c("vertical", "horizontal"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  sdimx = "sdimx",
  sdimy = "sdimy",
  spat_enr_names = NULL,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  dim_point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  dim_show_cluster_center = F,
  dim_show_center_label = T,
  dim_center_point_size = 4,
  dim_center_point_border_col = "black",
  dim_center_point_border_stroke = 0.1,
  dim_label_size = 4,
  dim_label_fontface = "bold",
  spat_show_cluster_center = F,
  spat_show_center_label = F,
  spat_center_point_size = 4,
  spat_label_size = 4,
  spat_label_fontface = "bold",
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  nn_network_alpha = 0.05,
  show_spatial_network = F,
  spat_network_name = "spatial_network",
```



```

    spat_network_color = "blue",
    spat_network_alpha = 0.5,
    show_spatial_grid = F,
    spat_grid_name = "spatial_grid",
    spat_grid_color = "blue",
    show_other_cells = T,
    other_cell_color = "lightgrey",
    dim_other_point_size = 1,
    spat_other_point_size = 1,
    spat_other_cells_alpha = 0.5,
    dim_show_legend = F,
    spat_show_legend = F,
    legend_text = 8,
    axis_text = 8,
    axis_title = 8,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatDimPlot"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>sdmx</code>	= spatial dimension to use on x-axis
<code>sdmy</code>	= spatial dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs

```

dim_point_size  size of points in dim. reduction space
dim_point_border_col
                border color of points in dim. reduction space
dim_point_border_stroke
                border stroke of points in dim. reduction space
spat_point_size
                size of spatial points
spat_point_border_col
                border color of spatial points
spat_point_border_stroke
                border stroke of spatial points
dim_show_cluster_center
                show the center of each cluster
dim_show_center_label
                provide a label for each cluster
dim_center_point_size
                size of the center point
dim_center_point_border_col
                border color of center point
dim_center_point_border_stroke
                stroke size of center point
dim_label_size  size of the center label
dim_label_fontface
                font of the center label
spat_show_cluster_center
                show the center of each cluster
spat_show_center_label
                provide a label for each cluster
spat_center_point_size
                size of the center point
spat_label_size
                size of the center label
spat_label_fontface
                font of the center label
show_NN_network
                show underlying NN network
nn_network_to_use
                type of NN network to use (kNN vs sNN)
network_name    name of NN network to use, if show_NN_network = TRUE
nn_network_alpha
                column to use for alpha of the edges
show_spatial_network
                show spatial network
spat_network_name
                name of spatial network to use
spat_network_color
                color of spatial network
show_spatial_grid
                show spatial grid

```

spat_grid_name	name of spatial grid to use
spat_grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
dim_other_point_size	size of not selected dim cells
spat_other_point_size	size of not selected spat cells
spat_other_cells_alpha	alpha of not selected spat cells
dim_show_legend	show legend of dimension reduction plot
spat_show_legend	show legend of spatial plot
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## See Also

[spatDimPlot2D](#) and [spatDimPlot3D](#) for 3D visualization.

## Examples

```
spatDimPlot(gobject)
```

---

spatDimPlot2D

*spatDimPlot2D*


---

## Description

Visualize cells according to spatial AND dimension reduction coordinates 2D

## Usage

```
spatDimPlot2D(
  gobject,
  plot_alignment = c("vertical", "horizontal"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  sdimx = "sdimx",
  sdimy = "sdimy",
  spat_enr_names = NULL,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  dim_point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  spat_point_size = 1,
  spat_point_border_col = "black",
  spat_point_border_stroke = 0.1,
  dim_show_cluster_center = F,
  dim_show_center_label = T,
  dim_center_point_size = 4,
  dim_center_point_border_col = "black",
  dim_center_point_border_stroke = 0.1,
  dim_label_size = 4,
  dim_label_fontface = "bold",
  spat_show_cluster_center = F,
  spat_show_center_label = F,
  spat_center_point_size = 4,
  spat_label_size = 4,
  spat_label_fontface = "bold",
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  nn_network_alpha = 0.05,
  show_spatial_network = F,
  spat_network_name = "spatial_network",
```

```

    spat_network_color = "blue",
    spat_network_alpha = 0.5,
    show_spatial_grid = F,
    spat_grid_name = "spatial_grid",
    spat_grid_color = "blue",
    show_other_cells = T,
    other_cell_color = "lightgrey",
    dim_other_point_size = 1,
    spat_other_point_size = 1,
    spat_other_cells_alpha = 0.5,
    dim_show_legend = F,
    spat_show_legend = F,
    legend_text = 8,
    axis_text = 8,
    axis_title = 8,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatDimPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>sdmx</code>	= spatial dimension to use on x-axis
<code>sdmy</code>	= spatial dimension to use on y-axis
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs

```

dim_point_size  size of points in dim. reduction space
dim_point_border_col
                border color of points in dim. reduction space
dim_point_border_stroke
                border stroke of points in dim. reduction space
spat_point_size
                size of spatial points
spat_point_border_col
                border color of spatial points
spat_point_border_stroke
                border stroke of spatial points
dim_show_cluster_center
                show the center of each cluster
dim_show_center_label
                provide a label for each cluster
dim_center_point_size
                size of the center point
dim_center_point_border_col
                border color of center point
dim_center_point_border_stroke
                stroke size of center point
dim_label_size  size of the center label
dim_label_fontface
                font of the center label
spat_show_cluster_center
                show the center of each cluster
spat_show_center_label
                provide a label for each cluster
spat_center_point_size
                size of the center point
spat_label_size
                size of the center label
spat_label_fontface
                font of the center label
show_NN_network
                show underlying NN network
nn_network_to_use
                type of NN network to use (kNN vs sNN)
network_name    name of NN network to use, if show_NN_network = TRUE
nn_network_alpha
                column to use for alpha of the edges
show_spatial_network
                show spatial network
spat_network_name
                name of spatial network to use
spat_network_color
                color of spatial network
show_spatial_grid
                show spatial grid

```

spat_grid_name	name of spatial grid to use
spat_grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
dim_other_point_size	size of not selected dim cells
spat_other_point_size	size of not selected spat cells
spat_other_cells_alpha	alpha of not selected spat cells
dim_show_legend	show legend of dimension reduction plot
spat_show_legend	show legend of spatial plot
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

## Details

Description of parameters.

## Value

ggplot

## See Also

[spatDimPlot3D](#)

## Examples

```
spatDimPlot2D(gobject)
```

---

spatDimPlot3D

*spatDimPlot3D*


---

## Description

Visualize cells according to spatial AND dimension reduction coordinates in plotly mode

## Usage

```
spatDimPlot3D(
  gobject,
  plot_alignment = c("horizontal", "vertical"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = 3,
  sdimx = "sdimx",
  sdimy = "sdimy",
  sdimz = "sdimz",
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  label_size = 16,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 1.5,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  dim_point_size = 3,
  nn_network_alpha = 0.5,
  show_spatial_network = F,
  spatial_network_name = "spatial_network",
  network_color = "lightgray",
  spatial_network_alpha = 0.5,
  show_spatial_grid = F,
  spatial_grid_name = "spatial_grid",
  spatial_grid_color = NULL,
  spatial_grid_alpha = 0.5,
  spatial_point_size = 3,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
```



```

    legend_text_size = 12,
    show_plot = NA,
    return_plot = NA,
    save_plot = NA,
    save_param = list(),
    default_save_name = "spatDimPlot3D"
)

```

## Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>sdimx</code>	= spatial dimension to use on x-axis
<code>sdimy</code>	= spatial dimension to use on y-axis
<code>sdimz</code>	= spatial dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>show_cluster_center</code>	show the center of each cluster
<code>show_center_label</code>	provide a label for each cluster
<code>center_point_size</code>	size of the center point
<code>label_size</code>	size of the center label
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors

<code>dim_point_size</code>	size of points in dim. reduction space
<code>nn_network_alpha</code>	column to use for alpha of the edges
<code>show_spatial_network</code>	show spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>spatial_network_alpha</code>	alpha of spatial network
<code>show_spatial_grid</code>	show spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>spatial_grid_color</code>	color of spatial grid
<code>spatial_point_size</code>	size of spatial points
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_param</code>	list of saving parameters from <code>all_plots_save_function()</code>
<code>default_save_name</code>	default save name for saving, don't change, change <code>save_name</code> in <code>save_param</code>
<code>dim_point_border_col</code>	border color of points in dim. reduction space
<code>dim_point_border_stroke</code>	border stroke of points in dim. reduction space
<code>spatial_network_color</code>	color of spatial network
<code>spatial_other_point_size</code>	size of not selected spatial points
<code>spatial_other_cells_alpha</code>	alpha of not selected spatial points
<code>dim_other_point_size</code>	size of not selected dim. reduction points
<code>show_legend</code>	show legend

### Details

Description of parameters.

### Value

plotly

### Examples

```
spatDimPlot3D(gobject)
```

spatGenePlot

*spatGenePlot***Description**

Visualize cells and gene expression according to spatial coordinates

**Usage**

```
spatGenePlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  genes_high_color = "darkred",
  genes_mid_color = "white",
  genes_low_color = "darkblue",
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  midpoint = 0,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "spatGenePlot"
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>genes_high_color</code>	color represents high gene expression
<code>genes_mid_color</code>	color represents middle gene expression

genes_low_color	color represents low gene expression
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
midpoint	expression midpoint
scale_alpha_with_expression	scale expression with ggplot alpha parameter
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
...	parameters for cowplot::save_plot()

**Details**

Description of parameters.

**Value**

ggplot

**See Also**

[spatGenePlot3D](#) and [spatGenePlot2D](#)

**Examples**

```
spatGenePlot(gobject)
```

---

spatGenePlot2D

*spatGenePlot2D*


---

## Description

Visualize cells and gene expression according to spatial coordinates

## Usage

```
spatGenePlot2D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  genes_high_color = "darkred",
  genes_mid_color = "white",
  genes_low_color = "darkblue",
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  midpoint = 0,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "spatGenePlot2D"
)
```

## Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>genes_high_color</code>	color represents high gene expression
<code>genes_mid_color</code>	color represents middle gene expression

genes_low_color	color represents low gene expression
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
midpoint	expression midpoint
scale_alpha_with_expression	scale expression with ggplot alpha parameter
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
...	parameters for cowplot::save_plot()

**Details**

Description of parameters.

**Value**

ggplot

**See Also**

[spatGenePlot3D](#)

**Examples**

```
spatGenePlot2D(gobject)
```

---

spatGenePlot3D	<i>spatGenePlot3D</i>
----------------	-----------------------

---

## Description

Visualize cells and gene expression according to spatial coordinates

## Usage

```
spatGenePlot3D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  cluster_column = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 1,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  spatial_grid_name = "spatial_grid",
  point_size = 2,
  show_legend = T,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "spatGenePlot3D"
)
```

## Arguments

gobject	giotto object
expression_values	gene expression values to use
genes	genes to show
show_network	show underlying spatial network
network_color	color of spatial network

spatial_network_name	name of spatial network to use
show_grid	show spatial grid
genes_high_color	color represents high gene expression
genes_mid_color	color represents middle gene expression
genes_low_color	color represents low gene expression
spatial_grid_name	name of spatial grid to use
point_size	size of point (cell)
show_legend	show legend
show_plot	show plots
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
grid_color	color of spatial grid
midpoint	expression midpoint
scale_alpha_with_expression	scale expression with ggplot alpha parameter
...	parameters for cowplot::save_plot()

Details

Description of parameters.

Value

ggplot

Examples

spatGenePlot3D(gobject)

---

Spatial_AEH	<i>Spatial_AEH</i>
-------------	--------------------

---

Description

calculate automatic expression histology with spatialDE method



Usage

```
Spatial_AEH(  
  gobject = NULL,  
  results = NULL,  
  pattern_num = 5,  
  l = 1.05,  
  show_AEH = T,  
  sdimx = NULL,  
  sdimy = NULL,  
  point_size = 3,  
  point_alpha = 1,  
  low_color = "blue",  
  mid_color = "white",  
  high_color = "red",  
  midpoint = 0,  
  python_path = NULL  
)
```

Arguments

gobject	Giotto object
results	output from spatial_DE
pattern_num	the number of gene expression patterns
show_AEH	show AEH plot
python_path	specify specific path to python if required

Details

Description.

Value

a list or a dataframe of SVs

Examples

```
Spatial_DE(gobject)
```

---

Spatial_DE	<i>Spatial_DE</i>
------------	-------------------

---

Description

calculate spatial variable genes with spatialDE method

Usage

```
Spatial_DE(  
  gobject = NULL,  
  show_plot = T,  
  size = c(4, 2, 1),  
  color = c("blue", "green", "red"),  
  sig_alpha = 0.5,  
  unsig_alpha = 0.5,  
  python_path = NULL  
)
```

Arguments

- gobject            Giotto object
- show\_plot        show FSV plot
- python\_path      specify specific path to python if required

Details

Description.

Value

a list or a dataframe of SVs

Examples

```
Spatial_DE(gobject)
```

---

spatPlot	<i>spatPlot</i>
----------	-----------------

---

Description

Visualize cells according to spatial coordinates

Usage

```
spatPlot(  
  gobject,  
  group_by = NULL,  
  group_by_subset = NULL,  
  sdimx = "sdimx",  
  sdimy = "sdimy",  
  spat_enr_names = NULL,  
  cell_color = NULL,  
  color_as_factor = T,  
  cell_color_code = NULL,  
  cell_color_gradient = c("blue", "white", "red"),  
  gradient_midpoint = NULL,  
  gradient_limits = NULL,
```

```

select_cell_groups = NULL,
select_cells = NULL,
point_size = 3,
point_border_col = "black",
point_border_stroke = 0.1,
show_cluster_center = F,
show_center_label = F,
center_point_size = 4,
center_point_border_col = "black",
center_point_border_stroke = 0.1,
label_size = 4,
label_fontface = "bold",
show_network = F,
spatial_network_name = "spatial_network",
network_color = NULL,
network_alpha = 1,
show_grid = F,
spatial_grid_name = "spatial_grid",
grid_color = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 1,
other_cells_alpha = 0.1,
coord_fix_ratio = NULL,
title = NULL,
show_legend = T,
legend_text = 8,
axis_text = 8,
axis_title = 8,
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spatPlot"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor

cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
show_network	show underlying spatial network
spatial_network_name	name of spatial network to use
network_color	color of spatial network
network_alpha	alpha of spatial network
show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	point size of not selected cells
other_cells_alpha	alpha of not selected cells
coord_fix_ratio	fix ratio between x and y-axis
title	title of plot
show_legend	show legend
legend_text	size of legend text

axis_text	size of axis text
axis_title	size of axis title
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column

Details

Description of parameters.

Value

ggplot

See Also

[spatPlot3D](#)

Examples

spatPlot(gobject)

---

spatPlot2D	<i>spatPlot2D</i>
------------	-------------------

---

Description

Visualize cells according to spatial coordinates

Usage

```
spatPlot2D(  
  gobject,  
  group_by = NULL,  
  group_by_subset = NULL,  
  sdimx = "sdimx",  
  sdimy = "sdimy",  
  spat_enr_names = NULL,  
  cell_color = NULL,  
  color_as_factor = T,  
  cell_color_code = NULL,
```

```

cell_color_gradient = c("blue", "white", "red"),
gradient_midpoint = NULL,
gradient_limits = NULL,
select_cell_groups = NULL,
select_cells = NULL,
point_size = 3,
point_border_col = "black",
point_border_stroke = 0.1,
show_cluster_center = F,
show_center_label = F,
center_point_size = 4,
center_point_border_col = "black",
center_point_border_stroke = 0.1,
label_size = 4,
label_fontface = "bold",
show_network = F,
spatial_network_name = "spatial_network",
network_color = NULL,
network_alpha = 1,
show_grid = F,
spatial_grid_name = "spatial_grid",
grid_color = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 1,
other_cells_alpha = 0.1,
coord_fix_ratio = NULL,
title = NULL,
show_legend = T,
legend_text = 8,
axis_text = 8,
axis_title = 8,
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spatPlot2D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>group_by_subset</code>	subset the <code>group_by</code> factor column
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_color</code>	color for cells (see details)

color_as_factor	convert color column to factor
cell_color_code	named vector with colors
cell_color_gradient	vector with 3 colors for numeric data
gradient_midpoint	midpoint for color gradient
gradient_limits	vector with lower and upper limits
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
label_fontface	font of labels
show_network	show underlying spatial network
spatial_network_name	name of spatial network to use
network_color	color of spatial network
network_alpha	alpha of spatial network
show_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
grid_color	color of spatial grid
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
other_point_size	point size of not selected cells
other_cells_alpha	alpha of not selected cells
coord_fix_ratio	fix ratio between x and y-axis
title	title of plot

show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param
groub_by	create multiple plots based on cell annotation column

Details

Description of parameters.

Value

ggplot

See Also

[spatPlot3D](#)

Examples

spatPlot2D(gobject)

---

spatPlot2D_single	<i>spatPlot2D_single</i>
-------------------	--------------------------

---

Description

Visualize cells according to spatial coordinates



**Usage**

```

spatPlot2D_single(
  gobject,
  sdimx = "sdimx",
  sdimy = "sdimy",
  spat_enr_names = NULL,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  cell_color_gradient = c("blue", "white", "red"),
  gradient_midpoint = NULL,
  gradient_limits = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  point_size = 3,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_cluster_center = F,
  show_center_label = F,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
  show_network = F,
  spatial_network_name = "spatial_network",
  network_color = NULL,
  network_alpha = 1,
  show_grid = F,
  spatial_grid_name = "spatial_grid",
  grid_color = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 1,
  other_cells_alpha = 0.1,
  coord_fix_ratio = NULL,
  title = NULL,
  show_legend = T,
  legend_text = 8,
  axis_text = 8,
  axis_title = 8,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "spatPlot2D_single"
)

```

**Arguments**

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')

<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>spat_enr_names</code>	names of spatial enrichment results to include
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>cell_color_gradient</code>	vector with 3 colors for numeric data
<code>gradient_midpoint</code>	midpoint for color gradient
<code>gradient_limits</code>	vector with lower and upper limits
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>show_network</code>	show underlying spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>network_color</code>	color of spatial network
<code>network_alpha</code>	alpha of spatial network
<code>show_grid</code>	show spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>grid_color</code>	color of spatial grid
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	point size of not selected cells
<code>other_cells_alpha</code>	alpha of not selected cells

coord_fix_ratio	fix ratio between x and y-axis
title	title of plot
show_legend	show legend
legend_text	size of legend text
axis_text	size of axis text
axis_title	size of axis title
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

Details

Description of parameters.

Value

ggplot

See Also

[spatPlot3D](#)

Examples

```
spatPlot2D_single(gobject)
```

---

spatPlot3D	<i>spatPlot3D</i>
------------	-------------------

---

Description

Visualize cells according to spatial coordinates

Usage

```
spatPlot3D(  
  gobject,  
  sdimx = "sdimx",  
  sdimy = "sdimy",  
  sdimz = "sdimz",  
  point_size = 3,  
  cell_color = NULL,  
  cell_color_code = NULL,  
  select_cell_groups = NULL,  
  select_cells = NULL,
```

```

show_other_cells = T,
other_cell_color = "lightgrey",
other_point_size = 0.5,
show_network = F,
network_color = NULL,
network_alpha = 1,
other_cell_alpha = 0.5,
spatial_network_name = "spatial_network",
show_grid = F,
grid_color = NULL,
spatial_grid_name = "spatial_grid",
title = "",
show_legend = T,
axis_scale = c("cube", "real", "custom"),
custom_ratio = NULL,
x_ticks = NULL,
y_ticks = NULL,
z_ticks = NULL,
show_plot = NA,
return_plot = NA,
save_plot = NA,
save_param = list(),
default_save_name = "spat3D"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimy')
<code>point_size</code>	size of point (cell)
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid

spatial_grid_name	name of spatial grid to use
title	title of plot
show_legend	show legend
axis_scale	the way to scale the axis
custom_ratio	customize the scale of the plot
x_ticks	set the number of ticks on the x-axis
y_ticks	set the number of ticks on the y-axis
z_ticks	set the number of ticks on the z-axis
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

Details

Description of parameters.

Value

ggplot

Examples

spatPlot3D(gobject)

---

specificCellCellcommunicationScores  
specificCellCellcommunicationScores

---

Description

Specific Cell-Cell communication scores based on spatial expression of interacting cells

Usage

```
specificCellCellcommunicationScores(  
  gobject,  
  spatial_network_name = "spatial_network",  
  cluster_column = "cell_types",  
  random_iter = 100,  
  cell_type_1 = "astrocyte",  
  cell_type_2 = "endothelial",  
  gene_set_1,  
  gene_set_2,
```

```
log2FC_addendum = 0.1,  
min_observations = 2,  
verbose = T  
)
```

**Arguments**

<code>gobject</code>	giotto object to use
<code>spatial_network_name</code>	spatial network to use for identifying interacting cells
<code>cluster_column</code>	cluster column with cell type information
<code>random_iter</code>	number of iterations
<code>cell_type_1</code>	first cell type
<code>cell_type_2</code>	second cell type
<code>gene_set_1</code>	first specific gene set from gene pairs
<code>gene_set_2</code>	second specific gene set from gene pairs
<code>log2FC_addendum</code>	addendum to add when calculating log2FC
<code>min_observations</code>	minimum number of interactions needed to be considered
<code>verbose</code>	verbose

**Details**

Details will follow.

**Value**

Cell-Cell communication scores for gene pairs based on spatial interaction

**Examples**

```
specificCellCellcommunicationScores(gobject)
```

---

<code>split_dendrogram_in_two</code>
<i>split_dendrogram_in_two</i>

---

**Description**

Merge selected clusters based on pairwise correlation scores and size of cluster.

**Usage**

```
split_dendrogram_in_two(dend)
```

**Arguments**

<code>dend</code>	dendrogram object
-------------------	-------------------

## Value

list of two dendrograms and height of node

## Examples

```
split_dendrogram_in_two(dend)
```

---

```
stitchFieldCoordinates
```

```
stitchFieldCoordinates
```

---

## Description

Helper function to stitch field coordinates together to form one complete picture

## Usage

```
stitchFieldCoordinates(
  location_file,
  offset_file,
  cumulate_offset_x = F,
  cumulate_offset_y = F,
  field_col = "Field of View",
  X_coord_col = "X",
  Y_coord_col = "Y",
  reverse_final_x = F,
  reverse_final_y = T
)
```

## Arguments

location_file	location dataframe with X and Y coordinates
offset_file	dataframe that describes the offset for each field (see details)
cumulate_offset_x	(boolean) Do the x-axis offset values need to be cumulated?
cumulate_offset_y	(boolean) Do the y-axis offset values need to be cumulated?
field_col	column that indicates the field within the location_file
X_coord_col	column that indicates the x coordinates
Y_coord_col	column that indicates the x coordinates
reverse_final_x	(boolean) Do the final x coordinates need to be reversed?
reverse_final_y	(boolean) Do the final y coordinates need to be reversed?

**Details**

Stitching of fields:

- 1. have cell locations: at least 3 columns: field, X, Y
- 2. create offset file: offset file has 3 columns: field, x\_offset, y\_offset
- 3. create new cell location file by stitching original cell locations with stitchFieldCoordinates
- 4. provide new cell location file to [createGiottoObject](#)

**Value**

Updated location dataframe with new X ['X\_final'] and Y ['Y\_final'] coordinates

**Examples**

```
stitchFieldCoordinates(gobject)
```

---

subClusterCells	<i>subClusterCells</i>
-----------------	------------------------

---

**Description**

subcluster cells

**Usage**

```
subClusterCells(
  gobject,
  name = "sub_clus",
  cluster_method = c("leiden", "louvain_community", "louvain_multinet"),
  cluster_column = NULL,
  selected_clusters = NULL,
  hvg_param = list(reverse_log_scale = T, difference_in_variance = 1, expression_values
    = "normalized"),
  hvg_min_perc_cells = 5,
  hvg_mean_expr_det = 1,
  use_all_genes_as_hvg = FALSE,
  min_nr_of_hvg = 5,
  pca_param = list(expression_values = "normalized", scale_unit = T),
  nn_param = list(dimensions_to_use = 1:20),
  k_neighbors = 10,
  resolution = 1,
  gamma = 1,
  omega = 1,
  python_path = NULL,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  return_gobject = TRUE,
  verbose = T
)
```



**Arguments**

<code>gobject</code>	giotto object
<code>name</code>	name for new clustering result
<code>cluster_method</code>	clustering method to use
<code>cluster_column</code>	cluster column to subcluster
<code>selected_clusters</code>	only do subclustering on these clusters
<code>hvg_param</code>	parameters for calculateHVG
<code>hvg_min_perc_cells</code>	threshold for detection in min percentage of cells
<code>hvg_mean_expr_det</code>	threshold for mean expression level in cells with detection
<code>use_all_genes_as_hvg</code>	forces all genes to be HVG and to be used as input for PCA
<code>min_nr_of_hvg</code>	minimum number of HVG, or all genes will be used as input for PCA
<code>pca_param</code>	parameters for runPCA
<code>nn_param</code>	parameters for createNearestNetwork
<code>k_neighbors</code>	number of k for createNearestNetwork
<code>resolution</code>	resolution
<code>gamma</code>	gamma
<code>omega</code>	omega
<code>python_path</code>	specify specific path to python if required
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use
<code>return_gobject</code>	boolean: return giotto object (default = TRUE)
<code>verbose</code>	verbose

**Details**

This function performs subclustering on selected clusters. The systematic steps are:

- 1. subset Giotto object
- 2. identify highly variable genes
- 3. run PCA
- 4. create nearest neighbouring network
- 5. do clustering

**Value**

giotto object with new subclusters appended to cell metadata

**See Also**

[doLouvainCluster\\_multinet](#), [doLouvainCluster\\_community](#) and @seealso [doLeidenCluster](#)

**Examples**

```
subClusterCells(gobject)
```

---

subsetGiotto

*subsetGiotto*


---

**Description**

subsets Giotto object including previous analyses.

**Usage**

```
subsetGiotto(gobject, cell_ids = NULL, gene_ids = NULL, verbose = FALSE)
```

**Arguments**

gobject	giotto object
cell_ids	cell IDs to keep
gene_ids	gene IDs to keep
verbose	be verbose

**Value**

giotto object

**Examples**

```
subsetGiotto(gobject)
```

---

subsetGiottoLocs

*subsetGiottoLocs*


---

**Description**

subsets Giotto object based on spatial locations

**Usage**

```
subsetGiottoLocs(
  gobject,
  x_max = NULL,
  x_min = NULL,
  y_max = NULL,
  y_min = NULL,
  z_max = NULL,
  z_min = NULL,
  return_gobject = T,
  verbose = FALSE
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>x_max</code>	maximum x-coordinate
<code>x_min</code>	minimum x-coordinate
<code>y_max</code>	maximum y-coordinate
<code>y_min</code>	minimum y-coordinate
<code>z_max</code>	maximum z-coordinate
<code>z_min</code>	minimum z-coordinate
<code>return_gobject</code>	return Giotto object

**Details**

if `return_gobject = FALSE`, then a filtered combined metadata `data.table` will be returned

**Value**

giotto object

**Examples**

```
subsetGiottoLocs(gobject)
```

---

viewHMRFresults

*viewHMRFresults*


---

**Description**

View results from doHMRF.

**Usage**

```
viewHMRFresults(
  gobject,
  HMRFoutput,
  k = NULL,
  betas_to_view = NULL,
  third_dim = NULL,
  ...
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>HMRFoutput</code>	HMRF output from doHMRF
<code>k</code>	number of HMRF domains
<code>betas_to_view</code>	results from different betas that you want to view
<code>...</code>	paramters to <code>visPlot()</code>

**Details**

Description ...

**Value**

spatial plots with HMRF domains

**See Also**

[visPlot](#)

**Examples**

`viewHMRFresults(gobject)`

---

<code>viewHMRFresults2D</code>	<i><code>viewHMRFresults2D</code></i>
--------------------------------	---------------------------------------

---

**Description**

View results from doHMRF.

**Usage**

```
viewHMRFresults2D(  
  gobject,  
  HMRFoutput,  
  k = NULL,  
  betas_to_view = NULL,  
  third_dim = NULL,  
  ...  
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>HMRFoutput</code>	HMRF output from doHMRF
<code>k</code>	number of HMRF domains
<code>betas_to_view</code>	results from different betas that you want to view
<code>...</code>	paramters to <code>visPlot()</code>

**Details**

Description ...

**Value**

spatial plots with HMRF domains

**See Also**

[spatPlot2D](#)

Examples

```
viewHMRResults2D(gobject)
```

---

viewHMRResults3D	<i>viewHMRResults3D</i>
------------------	-------------------------

---

Description

View results from doHMRF.

Usage

```
viewHMRResults3D(  
  gobject,  
  HMRFoutput,  
  k = NULL,  
  betas_to_view = NULL,  
  third_dim = NULL,  
  ...  
)
```

Arguments

gobject	giotto object
HMRFoutput	HMRF output from doHMRF
k	number of HMRF domains
betas_to_view	results from different betas that you want to view
...	paramters to visPlot()

Details

Description ...

Value

spatial plots with HMRF domains

See Also

[spatPlot3D](#)

Examples

```
viewHMRResults3D(gobject)
```

---

violinPlot

*violinPlot*


---

## Description

Creates violinplot for selected clusters

## Usage

```
violinPlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  cluster_column,
  cluster_custom_order = NULL,
  color_violin = c("genes", "cluster"),
  cluster_color_code = NULL,
  strip_position = c("top", "right", "left", "bottom"),
  strip_text = 7,
  axis_text_x_size = 10,
  axis_text_y_size = 6,
  show_plot = NA,
  return_plot = NA,
  save_plot = NA,
  save_param = list(),
  default_save_name = "violinPlot"
)
```

## Arguments

gobject	giotto object
expression_values	expression values to use
genes	genes to plot
cluster_column	name of column to use for clusters
cluster_custom_order	custom order of clusters
color_violin	color violin according to genes or clusters
cluster_color_code	color code for clusters
strip_position	position of gene labels
strip_text	size of strip text
axis_text_x_size	size of x-axis text
axis_text_y_size	size of y-axis text
show_plot	show plot
return_plot	return ggplot object

save_plot	directly save the plot [boolean]
save_param	list of saving parameters from all_plots_save_function()
default_save_name	default save name for saving, don't change, change save_name in save_param

**Value**

ggplot

**Examples**

```
violinPlot(gobject)
```

---

visDimGenePlot	<i>visDimGenePlot</i>
----------------	-----------------------

---

**Description**

Visualize cells and gene expression according to dimension reduction coordinates

**Usage**

```
visDimGenePlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  edge_alpha = NULL,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  point_border_col = "black",
  point_border_stroke = 0.1,
  midpoint = 0,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_legend = T,
  plot_method = c("ggplot", "plotly"),
  show_plots = F
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha</code>	column to use for alpha of the edges
<code>scale_alpha_with_expression</code>	scale expression with ggplot alpha parameter
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>midpoint</code>	size of point (cell)
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>show_legend</code>	show legend
<code>show_plots</code>	show plots

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
visDimGenePlot(gobject)
```



---

```
visDimGenePlot_2D_ggplot
      visDimGenePlot_2D_ggplot
```

---

## Description

Visualize cells and gene expression according to dimension reduction coordinates

## Usage

```
visDimGenePlot_2D_ggplot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  edge_alpha = NULL,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  genes_high_color = "red",
  genes_mid_color = "white",
  genes_low_color = "blue",
  point_border_col = "black",
  point_border_stroke = 0.1,
  midpoint = 0,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_legend = T,
  show_plots = F
)
```

## Arguments

gobject	giotto object
expression_values	gene expression values to use
genes	genes to show
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimension reduction name
dim1_to_use	dimension to use on x-axis

dim2_to_use	dimension to use on y-axis
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
edge_alpha	column to use for alpha of the edges
scale_alpha_with_expression	scale expression with ggplot alpha parameter
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
midpoint	size of point (cell)
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
show_plots	show plots

### Details

Description of parameters.

### Value

ggplot

### Examples

```
visDimGenePlot_2D_ggplot(gobject)
```

---

```
visDimGenePlot_3D_plotly
visDimGenePlot_3D_plotly
```

---

### Description

Visualize cells and gene expression according to dimension reduction coordinates

**Usage**

```
visDimGenePlot_3D_plotly(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = 3,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  network_color = "lightgray",
  edge_alpha = NULL,
  point_size = 1,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  show_legend = T,
  show_plots = F
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>show_legend</code>	show legend
<code>show_plots</code>	show plots

**Details**

Description of parameters.

Value

ggplot

Examples

```
visDimGenePlot_3D_plotly(gobject)
```

---

visDimPlot	<i>visDimPlot</i>
------------	-------------------

---

Description

Visualize cells according to dimension reduction coordinates

Usage

```
visDimPlot(  
  gobject,  
  dim_reduction_to_use = "umap",  
  dim_reduction_name = "umap",  
  dim1_to_use = 1,  
  dim2_to_use = 2,  
  dim3_to_use = NULL,  
  show_NN_network = F,  
  nn_network_to_use = "sNN",  
  network_name = "sNN.pca",  
  cell_color = NULL,  
  color_as_factor = T,  
  cell_color_code = NULL,  
  select_cell_groups = NULL,  
  select_cells = NULL,  
  show_other_cells = T,  
  other_cell_color = "lightgrey",  
  other_point_size = 0.5,  
  show_cluster_center = F,  
  show_center_label = T,  
  center_point_size = 4,  
  center_point_border_col = "black",  
  center_point_border_stroke = 0.1,  
  label_size = 4,  
  label_fontface = "bold",  
  edge_alpha = NULL,  
  point_size = 3,  
  point_border_col = "black",  
  point_border_stroke = 0.1,  
  plot_method = c("ggplot", "plotly"),  
  show_legend = T,  
  show_plot = F,  
  return_plot = TRUE,  
  save_plot = F,  
  save_dir = NULL,
```

```

    save_folder = NULL,
    save_name = NULL,
    save_format = NULL,
    show_saved_plot = F,
    ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>label_fontface</code>	font of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_legend</code>	show legend
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_dir</code>	directory to save the plot

save_folder	(optional) folder in directory to save the plot
save_name	name of plot
save_format	format of plot (e.g. tiff, png, pdf, ...)
show_saved_plot	load & display the saved plot

### Details

Description of parameters.

### Value

ggplot or plotly

### Examples

```
visDimPlot(gobject)
```

---

visDimPlot_2D_ggplot	<i>visDimPlot_2D_ggplot</i>
----------------------	-----------------------------

---

### Description

Visualize cells according to dimension reduction coordinates

### Usage

```
visDimPlot_2D_ggplot(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  center_point_border_col = "black",
  center_point_border_stroke = 0.1,
  label_size = 4,
  label_fontface = "bold",
```

```

    edge_alpha = NULL,
    point_size = 1,
    point_border_col = "black",
    point_border_stroke = 0.1,
    show_legend = T,
    show_plot = F,
    return_plot = TRUE,
    save_plot = F,
    save_dir = NULL,
    save_folder = NULL,
    save_name = NULL,
    save_format = NULL,
    show_saved_plot = F,
    ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>other_point_size</code>	size of not selected cells
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points

label_size	size of labels
label_fontface	font of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend

### Details

Description of parameters.

### Value

ggplot

### Examples

```
visDimPlot_2D_ggplot(gobject)
```

---

visDimPlot\_2D\_plotly    *visDimPlot\_2D\_plotly*

---

### Description

Visualize cells according to dimension reduction coordinates

### Usage

```
visDimPlot_2D_plotly(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  color_as_factor = T,
  cell_color = NULL,
  cell_color_code = NULL,
  show_cluster_center = F,
  show_center_label = T,
```



```

    center_point_size = 4,
    label_size = 4,
    edge_alpha = NULL,
    point_size = 5
  )

```

### Arguments

<code>gobject</code>	giotto object
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>color_as_factor</code>	convert color column to factor
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>show_cluster_center</code>	plot center of selected clusters
<code>show_center_label</code>	plot label of selected clusters
<code>center_point_size</code>	size of center points
<code>label_size</code>	size of labels
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)

### Details

Description of parameters.

### Value

plotly

### Examples

```
visDimPlot_2D_plotly(gobject)
```

---

visDimPlot\_3D\_plotly    *visDimPlot\_3D\_plotly*


---

## Description

Visualize cells according to dimension reduction coordinates

## Usage

```
visDimPlot_3D_plotly(
  gobject,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = 3,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  color_as_factor = T,
  cell_color = NULL,
  cell_color_code = NULL,
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  label_size = 4,
  edge_alpha = NULL,
  point_size = 1
)
```

## Arguments

gobject	giotto object
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimension reduction name
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
dim3_to_use	dimension to use on z-axis
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)

network_name	name of NN network to use, if show_NN_network = TRUE
color_as_factor	convert color column to factor
cell_color	color for cells (see details)
cell_color_code	named vector with colors
show_cluster_center	plot center of selected clusters
show_center_label	plot label of selected clusters
center_point_size	size of center points
label_size	size of labels
edge_alpha	column to use for alpha of the edges
point_size	size of point (cell)

### Details

Description of parameters.

### Value

plotly

### Examples

```
visDimPlot_3D_plotly(gobject)
```

---

visForceLayoutPlot	<i>visForceLayoutPlot</i>
--------------------	---------------------------

---

### Description

Visualize cells according to forced layout algorithm coordinates

### Usage

```
visForceLayoutPlot(
  gobject,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  layout_name = "layout",
  dim1_to_use = 1,
  dim2_to_use = 2,
  show_NN_network = T,
  cell_color = NULL,
  color_as_factor = TRUE,
  cell_color_code = NULL,
  edge_alpha = NULL,
  point_size = 1,
```

```

    point_border_col = "black",
    point_border_stroke = 0.1,
    show_legend = T,
    show_plot = F,
    return_plot = TRUE,
    save_plot = F,
    save_dir = NULL,
    save_folder = NULL,
    save_name = NULL,
    save_format = NULL,
    show_saved_plot = F,
    ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	NN network to use
<code>layout_name</code>	name of layout to use
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>edge_alpha</code>	column to use for alpha of the edges
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_legend</code>	show legend
<code>show_plot</code>	show plot
<code>return_plot</code>	return ggplot object
<code>save_plot</code>	directly save the plot [boolean]
<code>save_dir</code>	directory to save the plot
<code>save_folder</code>	(optional) folder in directory to save the plot
<code>save_name</code>	name of plot
<code>save_format</code>	format of plot (e.g. tiff, png, pdf, ...)
<code>show_saved_plot</code>	load & display the saved plot

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
visForceLayoutPlot(gobject)
```

---

visGenePlot	<i>visGenePlot</i>
-------------	--------------------

---

**Description**

Visualize cells and gene expression according to spatial coordinates

**Usage**

```
visGenePlot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  midpoint = 0,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  plot_method = c("ggplot", "plotly"),
  show_plots = F
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>genes_high_color</code>	color represents high gene expression
<code>genes_mid_color</code>	color represents middle gene expression
<code>genes_low_color</code>	color represents low gene expression
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>midpoint</code>	expression midpoint
<code>scale_alpha_with_expression</code>	scale expression with ggplot alpha parameter
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>show_legend</code>	show legend
<code>cow_n_col</code>	cowplot param: how many columns
<code>cow_rel_h</code>	cowplot param: relative height
<code>cow_rel_w</code>	cowplot param: relative width
<code>cow_align</code>	cowplot param: how to align
<code>axis_scale</code>	three mode to adjust axis scale
<code>x_ticks</code>	number of ticks on x axis
<code>y_ticks</code>	number of ticks on y axis
<code>z_ticks</code>	number of ticks on z axis
<code>plot_method</code>	two methods of plot
<code>show_plots</code>	show plots

**Details**

Description of parameters.

**Value**

ggplot or plotly

**Examples**

```
visGenePlot(gobject)
```

---

```
visGenePlot_2D_ggplot  visGenePlot_2D_ggplot
```

---

**Description**

Visualize cells and gene expression according to spatial coordinates

**Usage**

```
visGenePlot_2D_ggplot(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  genes_high_color = "darkred",
  genes_mid_color = "white",
  genes_low_color = "darkblue",
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  grid_color = NULL,
  spatial_grid_name = "spatial_grid",
  midpoint = 0,
  scale_alpha_with_expression = FALSE,
  point_size = 1,
  point_border_col = "black",
  point_border_stroke = 0.1,
  show_legend = T,
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  show_plots = F
)
```

**Arguments**

gobject	giotto object
expression_values	gene expression values to use
genes	genes to show
genes_high_color	color represents high gene expression
genes_mid_color	color represents middle gene expression

genes_low_color	color represents low gene expression
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
midpoint	expression midpoint
scale_alpha_with_expression	scale expression with ggplot alpha parameter
point_size	size of point (cell)
point_border_col	color of border around points
point_border_stroke	stroke size of border around points
show_legend	show legend
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_plots	show plots

### Details

Description of parameters.

### Value

ggplot

### Examples

```
visGenePlot_2D_ggplot(gobject)
```

---

visGenePlot\_3D\_plotly    *visGenePlot\_3D\_plotly*

---

### Description

Visualize cells and gene expression according to spatial coordinates



**Usage**

```
visGenePlot_3D_plotly(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  genes,
  show_network = F,
  network_color = NULL,
  spatial_network_name = "spatial_network",
  edge_alpha = NULL,
  show_grid = F,
  genes_high_color = NULL,
  genes_mid_color = "white",
  genes_low_color = "blue",
  spatial_grid_name = "spatial_grid",
  point_size = 1,
  show_legend = T,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  show_plots = F
)
```

**Arguments**

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>genes</code>	genes to show
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>show_grid</code>	show spatial grid
<code>genes_high_color</code>	color represents high gene expression
<code>genes_mid_color</code>	color represents middle gene expression
<code>genes_low_color</code>	color represents low gene expression
<code>spatial_grid_name</code>	name of spatial grid to use
<code>point_size</code>	size of point (cell)
<code>show_legend</code>	show legend
<code>axis_scale</code>	three mode to adjust axis scale
<code>x_ticks</code>	number of ticks on x axis
<code>y_ticks</code>	number of ticks on y axis

z_ticks	number of ticks on z axis
show_plots	show plots
grid_color	color of spatial grid
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align

Details

Description of parameters.

Value

plotly

Examples

```
visGenePlot_3D_plotly(gobject)
```

---

visPlot	<i>visPlot</i>
---------	----------------

---

Description

Visualize cells according to spatial coordinates

Usage

```
visPlot(  
  gobject,  
  sdimx = NULL,  
  sdimy = NULL,  
  sdimz = NULL,  
  point_size = 3,  
  point_border_col = "black",  
  point_border_stroke = 0.1,  
  cell_color = NULL,  
  cell_color_code = NULL,  
  color_as_factor = T,  
  select_cell_groups = NULL,  
  select_cells = NULL,  
  show_other_cells = T,  
  other_cell_color = "lightgrey",  
  show_network = F,  
  network_color = NULL,  
  network_alpha = 1,  
  other_cell_alpha = 0.1,  
  spatial_network_name = "spatial_network",  
  show_grid = F,  
)
```

```

    grid_color = NULL,
    grid_alpha = 1,
    spatial_grid_name = "spatial_grid",
    coord_fix_ratio = 0.6,
    title = "",
    show_legend = T,
    axis_scale = c("cube", "real", "custom"),
    custom_ratio = NULL,
    x_ticks = NULL,
    y_ticks = NULL,
    z_ticks = NULL,
    plot_method = c("ggplot", "plotly"),
    show_plot = F,
    return_plot = TRUE,
    save_plot = F,
    save_dir = NULL,
    save_folder = NULL,
    save_name = NULL,
    save_format = NULL,
    show_saved_plot = F,
    ...
)

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use

show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
coord_fix_ratio	fix ratio between x and y-axis
title	title of plot
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_dir	directory to save the plot
save_folder	(optional) folder in directory to save the plot
save_name	name of plot
save_format	format of plot (e.g. tiff, png, pdf, ...)
show_saved_plot	load & display the saved plot

### Details

Description of parameters.

### Value

ggplot

### Examples

```
visPlot(gobject)
```

---

visPlot_2D_ggplot	<i>visPlot_2D_ggplot</i>
-------------------	--------------------------

---

### Description

Visualize cells according to spatial coordinates

### Usage

```
visPlot_2D_ggplot(
  gobject,
  sdimx = NULL,
  sdimy = NULL,
  point_size = 3,
  point_border_col = "black",
  point_border_stroke = 0.1,
  cell_color = NULL,
  cell_color_code = NULL,
```

```

    color_as_factor = T,
    select_cell_groups = NULL,
    select_cells = NULL,
    show_other_cells = T,
    other_cell_color = "lightgrey",
    show_network = F,
    network_color = NULL,
    network_alpha = 1,
    other_cells_alpha = 0.1,
    spatial_network_name = "spatial_network",
    show_grid = F,
    grid_color = NULL,
    spatial_grid_name = "spatial_grid",
    coord_fix_ratio = 0.6,
    title = "",
    show_legend = T,
    axis_scale = c("cube", "real", "custom"),
    custom_ratio = NULL,
    x_ticks = NULL,
    y_ticks = NULL,
    z_ticks = NULL,
    show_plot = F,
    return_plot = TRUE,
    save_plot = F,
    save_dir = NULL,
    save_folder = NULL,
    save_name = NULL,
    save_format = NULL,
    show_saved_plot = F,
    ...
  )

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>point_size</code>	size of point (cell)
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs

show_other_cells	display not selected cells
other_cell_color	color of not selected cells
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
spatial_grid_name	name of spatial grid to use
coord_fix_ratio	fix ratio between x and y-axis
title	title of plot
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_dir	directory to save the plot
save_folder	(optional) folder in directory to save the plot
save_name	name of plot
save_format	format of plot (e.g. tiff, png, pdf, ...)
show_saved_plot	load & display the saved plot

## Details

Description of parameters.

## Value

ggplot

## Examples

```
visPlot_2D_ggplot(gobject)
```

---

visPlot_2D_plotly	<i>visPlot_2D_plotly</i>
-------------------	--------------------------

---

## Description

Visualize cells according to spatial coordinates

## Usage

```
visPlot_2D_plotly(
  gobject,
  sdimx = NULL,
  sdimy = NULL,
  point_size = 3,
  cell_color = NULL,
  cell_color_code = NULL,
  color_as_factor = T,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  other_point_size = 0.5,
  show_network = F,
  network_color = "lightgray",
  network_alpha = 1,
  other_cell_alpha = 0.5,
  spatial_network_name = "spatial_network",
  show_grid = F,
  grid_color = NULL,
  grid_alpha = 1,
  spatial_grid_name = "spatial_grid",
  show_legend = T,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  show_plot = F
)
```

## Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>point_size</code>	size of point (cell)
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>color_as_factor</code>	convert color column to factor

select_cell_groups	select a subset of the groups from cell_color
show_network	show underlying spatial network
network_color	color of spatial network
spatial_network_name	name of spatial network to use
show_grid	show spatial grid
grid_color	color of spatial grid
grid_alpha	alpha of spatial grid
spatial_grid_name	name of spatial grid to use
show_legend	show legend
show_plot	show plot

Details

Description of parameters.

Value

plotly

Examples

visPlot\_2D\_plotly(gobject)

---

visPlot_3D_plotly	<i>visPlot_3D_plotly</i>
-------------------	--------------------------

---

Description

Visualize cells according to spatial coordinates

Usage

```
visPlot_3D_plotly(  
  gobject,  
  sdimx = NULL,  
  sdimy = NULL,  
  sdimz = NULL,  
  point_size = 3,  
  cell_color = NULL,  
  cell_color_code = NULL,  
  select_cell_groups = NULL,  
  select_cells = NULL,  
  show_other_cells = T,  
  other_cell_color = "lightgrey",  
  other_point_size = 0.5,  
  show_network = F,
```



```

    network_color = NULL,
    network_alpha = 1,
    other_cell_alpha = 0.5,
    spatial_network_name = "spatial_network",
    spatial_grid_name = "spatial_grid",
    title = "",
    show_legend = T,
    axis_scale = c("cube", "real", "custom"),
    custom_ratio = NULL,
    x_ticks = NULL,
    y_ticks = NULL,
    z_ticks = NULL,
    show_plot = F
)

```

### Arguments

<code>gobject</code>	giotto object
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>point_size</code>	size of point (cell)
<code>cell_color</code>	color for cells (see details)
<code>cell_color_code</code>	named vector with colors
<code>select_cell_groups</code>	select a subset of the groups from <code>cell_color</code>
<code>show_network</code>	show underlying spatial network
<code>network_color</code>	color of spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>spatial_grid_name</code>	name of spatial grid to use
<code>title</code>	title of plot
<code>show_legend</code>	show legend
<code>show_plot</code>	show plot
<code>point_border_col</code>	color of border around points
<code>point_border_stroke</code>	stroke size of border around points
<code>color_as_factor</code>	convert color column to factor
<code>show_grid</code>	show spatial grid
<code>grid_color</code>	color of spatial grid
<code>coord_fix_ratio</code>	fix ratio between x and y-axis

**Details**

Description of parameters.

**Value**

ggplot

**Examples**

```
visPlot_3D_plotly(gobject)
```

---

visSpatDimGenePlot	<i>visSpatDimGenePlot</i>
--------------------	---------------------------

---

**Description**

integration of visSpatDimGenePlot\_2D(ggplot) and visSpatDimGenePlot\_3D(plotly)

**Usage**

```
visSpatDimGenePlot(
  gobject,
  plot_method = c("ggplot", "plotly"),
  expression_values = c("normalized", "scaled", "custom"),
  plot_alignment = c("horizontal", "vertical"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = NULL,
  sdimx = NULL,
  sdimy = NULL,
  sdimz = NULL,
  genes,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  edge_alpha_dim = NULL,
  scale_alpha_with_expression = FALSE,
  label_size = 16,
  genes_low_color = "blue",
  genes_mid_color = "white",
  genes_high_color = "red",
  dim_point_size = 3,
  nn_network_alpha = 0.5,
  show_spatial_network = F,
  spatial_network_name = "spatial_network",
  network_color = "lightgray",
  spatial_network_alpha = 0.5,
```

```

show_spatial_grid = F,
spatial_grid_name = "spatial_grid",
spatial_grid_color = NULL,
spatial_grid_alpha = 0.5,
spatial_point_size = 3,
spatial_point_border_col = "black",
spatial_point_border_stroke = 0.1,
legend_text_size = 12,
axis_scale = c("cube", "real", "custom"),
custom_ratio = NULL,
x_ticks = NULL,
y_ticks = NULL,
z_ticks = NULL,
midpoint = 0,
point_size = 1,
cow_n_col = 2,
cow_rel_h = 1,
cow_rel_w = 1,
cow_align = "h",
show_legend = T,
show_plots = F
)

```

### Arguments

<code>gobject</code>	giotto object
<code>expression_values</code>	gene expression values to use
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>sdimx</code>	x-axis dimension name (default = 'sdimx')
<code>sdimy</code>	y-axis dimension name (default = 'sdimy')
<code>sdimz</code>	z-axis dimension name (default = 'sdimz')
<code>genes</code>	genes to show
<code>dim_point_border_col</code>	color of border around points
<code>dim_point_border_stroke</code>	stroke size of border around points
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>

edge_alpha_dim	dim reduction plot: column to use for alpha of the edges
scale_alpha_with_expression	scale expression with ggplot alpha parameter
label_size	size for the label
genes_low_color	color to represent low expression of gene
genes_high_color	color to represent high expression of gene
dim_point_size	dim reduction plot: point size
spatial_network_name	name of spatial network to use
spatial_grid_name	name of spatial grid to use
spatial_point_size	spatial plot: point size
spatial_point_border_col	color of border around points
spatial_point_border_stroke	stroke size of border around points
legend_text_size	the size of the text in legend
axis_scale	three modes to adjust axis scale ratio
custom_ratio	set the axis scale ratio on custom
x_ticks	number of ticks on x axis
y_ticks	number of ticks on y axis
z_ticks	number of ticks on z axis
midpoint	size of point (cell)
point_size	size of point (cell)
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
show_plot	show plot

## Details

Description of parameters.

## Value

ggplot or plotly

## Examples

```
visSpatDimGenePlot(gobject)
```

---

visSpatDimGenePlot\_2D    *visSpatDimGenePlot\_2D*


---

## Description

Visualize cells according to spatial AND dimension reduction coordinates in ggplot mode

## Usage

```
visSpatDimGenePlot_2D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  plot_alignment = c("horizontal", "vertical"),
  genes,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  point_size = 1,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  show_NN_network = F,
  show_spatial_network = F,
  show_spatial_grid = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  edge_alpha_dim = NULL,
  scale_alpha_with_expression = FALSE,
  spatial_network_name = "spatial_network",
  spatial_grid_name = "spatial_grid",
  spatial_point_size = 1,
  spatial_point_border_col = "black",
  spatial_point_border_stroke = 0.1,
  midpoint = 0,
  genes_high_color = "red",
  genes_mid_color = "white",
  genes_low_color = "blue",
  cow_n_col = 2,
  cow_rel_h = 1,
  cow_rel_w = 1,
  cow_align = "h",
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  show_legend = T,
  show_plots = F
)
```

## Arguments

gobject                      giotto object

expression_values	gene expression values to use
plot_alignment	direction to align plot
genes	genes to show
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimension reduction name
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
point_size	size of point (cell)
dim_point_border_col	color of border around points
dim_point_border_stroke	stroke size of border around points
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
edge_alpha_dim	dim reduction plot: column to use for alpha of the edges
scale_alpha_with_expression	scale expression with ggplot alpha parameter
spatial_network_name	name of spatial network to use
spatial_grid_name	name of spatial grid to use
spatial_point_size	spatial plot: point size
spatial_point_border_col	color of border around points
spatial_point_border_stroke	stroke size of border around points
midpoint	size of point (cell)
cow_n_col	cowplot param: how many columns
cow_rel_h	cowplot param: relative height
cow_rel_w	cowplot param: relative width
cow_align	cowplot param: how to align
show_legend	show legend
dim_point_size	dim reduction plot: point size
show_plot	show plot

## Details

Description of parameters.

**Value**

ggplot

**Examples**

```
visSpatDimGenePlot_2D(gobject)
```

---

```
visSpatDimGenePlot_3D  visSpatDimGenePlot_3D
```

---

**Description**

Visualize cells according to spatial AND dimension reduction coordinates in plotly mode

**Usage**

```
visSpatDimGenePlot_3D(
  gobject,
  expression_values = c("normalized", "scaled", "custom"),
  plot_alignment = c("horizontal", "vertical"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = NULL,
  sdimx = NULL,
  sdimy = NULL,
  sdimz = NULL,
  genes,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  label_size = 16,
  genes_low_color = "blue",
  genes_mid_color = "white",
  genes_high_color = "red",
  dim_point_size = 3,
  nn_network_alpha = 0.5,
  show_spatial_network = F,
  spatial_network_name = "spatial_network",
  network_color = "lightgray",
  spatial_network_alpha = 0.5,
  show_spatial_grid = F,
  spatial_grid_name = "spatial_grid",
  spatial_grid_color = NULL,
  spatial_grid_alpha = 0.5,
  spatial_point_size = 3,
  legend_text_size = 12,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
```

```

    y_ticks = NULL,
    z_ticks = NULL
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>genes_low_color</code>	color represent high gene expression (see details)
<code>genes_high_color</code>	color represent high gene expression (see details)
<code>nn_network_alpha</code>	column to use for alpha of the edges
<code>show_spatial_network</code>	show spatial network
<code>spatial_network_name</code>	name of spatial network to use
<code>network_color</code>	color of spatial/nn network
<code>spatial_network_alpha</code>	alpha of spatial network
<code>show_spatial_grid</code>	show spatial grid
<code>spatial_grid_name</code>	name of spatial grid to use
<code>spatial_grid_color</code>	color of spatial grid
<code>spatial_grid_alpha</code>	alpha of spatial grid
<code>legend_text_size</code>	text size of legend
<code>show_legend</code>	show legend
<code>show_plot</code>	show plot

### Details

Description of parameters.



**Value**

plotly

**Examples**

```
visSpatDimPlot_3D(gobject)
```

---

visSpatDimPlot

*visSpatDimPlot*


---

**Description**

integration of visSpatDimPlot\_2D and visSpatDimPlot\_3D

**Usage**

```
visSpatDimPlot(
  gobject,
  plot_method = c("ggplot", "plotly"),
  plot_alignment = NULL,
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = NULL,
  sdims = NULL,
  sdims = NULL,
  sdims = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  label_size = NULL,
  label_fontface = "bold",
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  select_cell_groups = NULL,
  select_cells = NULL,
  show_other_cells = T,
  other_cell_color = "lightgrey",
  dim_point_size = 3,
  dim_point_border_col = "black",
  dim_point_border_stroke = 0.1,
  nn_network_alpha = NULL,
  show_spatial_network = F,
  spatial_network_name = "spatial_network",
  network_color = "lightgray",
  spatial_network_alpha = 0.5,
```

```

show_spatial_grid = F,
spatial_grid_name = "spatial_grid",
spatial_grid_color = NULL,
spatial_grid_alpha = 0.5,
spatial_point_size = 3,
legend_text_size = 12,
spatial_point_border_col = "black",
spatial_point_border_stroke = 0.1,
show_legend = T,
axis_scale = c("cube", "real", "custom"),
custom_ratio = NULL,
x_ticks = NULL,
y_ticks = NULL,
z_ticks = NULL,
show_plot = F
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>dim3_to_use</code>	dimension to use on z-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)
<code>color_as_factor</code>	convert color column to factor
<code>cell_color_code</code>	named vector with colors
<code>select_cell_groups</code>	select subset of cells/clusters based on <code>cell_color</code> parameter
<code>select_cells</code>	select subset of cells based on cell IDs
<code>show_other_cells</code>	display not selected cells
<code>other_cell_color</code>	color of not selected cells
<code>nn_network_alpha</code>	column to use for alpha of the edges
<code>show_spatial_network</code>	show spatial network

```

spatial_network_name
    name of spatial network to use
spatial_network_alpha
    alpha of spatial network
show_spatial_grid
    show spatial grid
spatial_grid_name
    name of spatial grid to use
spatial_grid_color
    color of spatial grid
spatial_grid_alpha
    alpha of spatial grid
legend_text_size
    text size of legend
show_legend
    show legend
show_plot
    show plot
plot_mode
    choose the mode to draw plot : ggplot or plotly
spatial_network_color
    color of spatial network

```

### Details

Description of parameters.

### Value

ggplot or plotly

### Examples

```
visSpatDimPlot(gobject)
```

---

visSpatDimPlot_2D	<i>visSpatDimPlot_2D</i>
-------------------	--------------------------

---

### Description

Visualize cells according to spatial AND dimension reduction coordinates in ggplot2 mode

### Usage

```

visSpatDimPlot_2D(
  gobject,
  plot_alignment = c("vertical", "horizontal"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  sdimx = NULL,
  sdimy = NULL,

```

```

show_NN_network = F,
nn_network_to_use = "sNN",
network_name = "sNN.pca",
show_cluster_center = F,
show_center_label = T,
center_point_size = 4,
label_size = 4,
label_fontface = "bold",
cell_color = NULL,
color_as_factor = T,
cell_color_code = NULL,
select_cell_groups = NULL,
select_cells = NULL,
show_other_cells = T,
other_cell_color = "lightgrey",
dim_plot_mode = NULL,
dim_point_size = 1,
dim_point_border_col = "black",
dim_point_border_stroke = 0.1,
nn_network_alpha = 0.05,
show_spatial_network = F,
spatial_network_name = "spatial_network",
spatial_network_color = NULL,
show_spatial_grid = F,
spatial_grid_name = "spatial_grid",
spatial_grid_color = NULL,
spatial_point_size = 1,
spatial_point_border_col = "black",
spatial_point_border_stroke = 0.1,
show_legend = T,
show_plot = F,
plot_method = "ggplot"
)

```

### Arguments

<code>gobject</code>	giotto object
<code>plot_alignment</code>	direction to align plot
<code>dim_reduction_to_use</code>	dimension reduction to use
<code>dim_reduction_name</code>	dimension reduction name
<code>dim1_to_use</code>	dimension to use on x-axis
<code>dim2_to_use</code>	dimension to use on y-axis
<code>show_NN_network</code>	show underlying NN network
<code>nn_network_to_use</code>	type of NN network to use (kNN vs sNN)
<code>network_name</code>	name of NN network to use, if <code>show_NN_network = TRUE</code>
<code>cell_color</code>	color for cells (see details)

color_as_factor	convert color column to factor
cell_color_code	named vector with colors
select_cell_groups	select subset of cells/clusters based on cell_color parameter
select_cells	select subset of cells based on cell IDs
show_other_cells	display not selected cells
other_cell_color	color of not selected cells
nn_network_alpha	column to use for alpha of the edges
show_spatial_network	show spatial network
spatial_network_name	name of spatial network to use
spatial_network_color	color of spatial network
show_spatial_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
spatial_grid_color	color of spatial grid
show_legend	show legend
show_plot	show plot
return_plot	return ggplot object
save_plot	directly save the plot [boolean]
save_dir	directory to save the plot
save_folder	(optional) folder in directory to save the plot
save_name	name of plot
save_format	format of plot (e.g. tiff, png, pdf, ...)
show_saved_plot	load & display the saved plot

### Details

Description of parameters.

### Value

ggplot

### Examples

```
visSpatDimPlot_2D(gobject)
```

---

visSpatDimPlot_3D	<i>visSpatDimPlot_3D</i>
-------------------	--------------------------

---

## Description

Visualize cells according to spatial AND dimension reduction coordinates in plotly mode

## Usage

```
visSpatDimPlot_3D(
  gobject,
  plot_alignment = c("horizontal", "vertical"),
  dim_reduction_to_use = "umap",
  dim_reduction_name = "umap",
  dim1_to_use = 1,
  dim2_to_use = 2,
  dim3_to_use = NULL,
  sdims = NULL,
  sdims = NULL,
  sdims = NULL,
  show_NN_network = F,
  nn_network_to_use = "sNN",
  network_name = "sNN.pca",
  show_cluster_center = F,
  show_center_label = T,
  center_point_size = 4,
  label_size = 16,
  cell_color = NULL,
  color_as_factor = T,
  cell_color_code = NULL,
  dim_point_size = 3,
  nn_network_alpha = 0.5,
  show_spatial_network = F,
  spatial_network_name = "spatial_network",
  network_color = "lightgray",
  spatial_network_alpha = 0.5,
  show_spatial_grid = F,
  spatial_grid_name = "spatial_grid",
  spatial_grid_color = NULL,
  spatial_grid_alpha = 0.5,
  spatial_point_size = 3,
  axis_scale = c("cube", "real", "custom"),
  custom_ratio = NULL,
  x_ticks = NULL,
  y_ticks = NULL,
  z_ticks = NULL,
  legend_text_size = 12
)
```

## Arguments

gobject                  giotto object

plot_alignment	direction to align plot
dim_reduction_to_use	dimension reduction to use
dim_reduction_name	dimension reduction name
dim1_to_use	dimension to use on x-axis
dim2_to_use	dimension to use on y-axis
dim3_to_use	dimension to use on z-axis
show_NN_network	show underlying NN network
nn_network_to_use	type of NN network to use (kNN vs sNN)
network_name	name of NN network to use, if show_NN_network = TRUE
cell_color	color for cells (see details)
color_as_factor	convert color column to factor
cell_color_code	named vector with colors
nn_network_alpha	column to use for alpha of the edges
show_spatial_network	show spatial network
spatial_network_name	name of spatial network to use
spatial_network_alpha	alpha of spatial network
show_spatial_grid	show spatial grid
spatial_grid_name	name of spatial grid to use
spatial_grid_color	color of spatial grid
spatial_grid_alpha	alpha of spatial grid
legend_text_size	text size of legend
spatial_network_color	color of spatial network
show_legend	show legend
show_plot	show plot

## Details

Description of parameters.

## Value

plotly

**Examples**

```
visSpatDimPlot_3D(gobject)
```

---

writeHMRResults	<i>writeHMRResults</i>
-----------------	------------------------

---

**Description**

write results from doHMRF to a data.table.

**Usage**

```
writeHMRResults(
  gobject,
  HMRFoutput,
  k = NULL,
  betas_to_view = NULL,
  print_command = F
)
```

**Arguments**

gobject	giotto object
HMRFoutput	HMRF output from doHMRF
k	k to write results for
betas_to_view	results from different betas that you want to view
print_command	see the python command

**Value**

data.table with HMRF results for each b and the selected k

**Examples**

```
writeHMRResults(gobject)
```

---

write_giotto_viewer_annotation	<i>write_giotto_viewer_annotation</i>
--------------------------------	---------------------------------------

---

**Description**

write out factor-like annotation data from a giotto object for the Viewer

**Usage**

```
write_giotto_viewer_annotation(
  annotation,
  annot_name = "test",
  output_directory = getwd()
)
```



**Arguments**

annotation	annotation from the data.table from giotto object
annot_name	name of the annotation
output_directory	directory where to save the files

**Value**

write a .txt and .annot file for the selection annotation

---

```
write_giotto_viewer_dim_reduction
      write_giotto_viewer_dim_reduction
```

---

**Description**

write out dimensional reduction data from a giotto object for the Viewer

**Usage**

```
write_giotto_viewer_dim_reduction(
  dim_reduction_cell,
  dim_red = NULL,
  dim_red_name = NULL,
  dim_red_rounding = NULL,
  dim_red_rescale = c(-20, 20),
  output_directory = getwd()
)
```

**Arguments**

dim_reduction_cell	dimension reduction slot from giotto object
dim_red	high level name of dimension reduction
dim_red_name	specific name of dimension reduction to use
dim_red_rounding	numerical indicating how to round the coordinates
dim_red_rescale	numericals to rescale the coordinates
output_directory	directory where to save the files

**Value**

write a .txt and .annot file for the selection annotation

---

```
write_giotto_viewer_numeric_annotation  
    write_giotto_viewer_numeric_annotation
```

---

**Description**

write out numeric annotation data from a giotto object for the Viewer

**Usage**

```
write_giotto_viewer_numeric_annotation(  
  annotation,  
  annot_name = "test",  
  output_directory = getwd()  
)
```

**Arguments**

annotation	annotation from the data.table from giotto object
annot_name	name of the annotation
output_directory	directory where to save the files

**Value**

write a .txt and .annot file for the selection annotation

# Index

- \*Topic **giotto**,
  - giotto-class, [131](#)
  - print.giotto, [173](#)
  - show,giotto-method, [181](#)
- \*Topic **giotto**
  - createGiottoObject, [46](#)
- \*Topic **object**
  - giotto-class, [131](#)
  - print.giotto, [173](#)
  - show,giotto-method, [181](#)
- [addCellMetadata](#), [6](#), [48](#)
- [addCellStatistics](#), [7](#), [12](#)
- [addGeneMetadata](#), [8](#), [48](#)
- [addGeneStatistics](#), [9](#), [12](#)
- [addHMRF](#), [10](#)
- [addNetworkLayout](#), [10](#)
- [addStatistics](#), [11](#)
- [adjustGiottoMatrix](#), [12](#)
- [aes\\_string2](#), [13](#)
- [all\\_plots\\_save\\_function](#), [14](#)
- [allCellCellcommunicationsScores](#), [13](#)
- [annotate\\_spatlocs\\_with\\_spatgrid\\_2D](#), [17](#)
- [annotate\\_spatlocs\\_with\\_spatgrid\\_3D](#), [17](#)
- [annotateGiotto](#), [15](#)
- [annotateSpatialNetwork](#), [16](#)
- [average\\_gene\\_gene\\_expression\\_in\\_groups](#),  
[18](#)
- [binGetSpatialGenes](#), [18](#)
- [calculate\\_spatial\\_genes\\_python](#), [23](#)
- [calculateHVG](#), [20](#)
- [calculateMetaTable](#), [21](#)
- [calculateMetaTableCells](#), [22](#)
- [cellProximityBarplot](#), [24](#)
- [cellProximityEnrichment](#), [25](#)
- [cellProximityHeatmap](#), [26](#)
- [cellProximityNetwork](#), [27](#)
- [cellProximitySpatPlot](#), [28](#)
- [cellProximitySpatPlot2D](#), [29](#), [30](#)
- [cellProximitySpatPlot3D](#), [29](#), [32](#)
- [cellProximityVisPlot](#), [34](#)
- [cellProximityVisPlot\\_2D\\_ggplot](#), [36](#)
- [cellProximityVisPlot\\_2D\\_plotly](#), [38](#)
- [cellProximityVisPlot\\_3D\\_plotly](#), [39](#)
- [changeGiottoInstructions](#), [41](#)
- [cluster\\_walktrap](#), [99](#)
- [clusterCells](#), [42](#)
- [combineMetadata](#), [44](#)
- [convertEnsemblToGeneSymbol](#), [45](#)
- [cowplot::save\\_plot](#), [131](#)
- [create\\_average\\_detection\\_DT](#), [56](#)
- [create\\_average\\_DT](#), [56](#)
- [create\\_cell\\_type\\_random\\_cell\\_IDs](#), [57](#)
- [create\\_cluster\\_matrix](#), [58](#)
- [create\\_dimObject](#), [58](#)
- [createGiottoInstructions](#), [45](#), [47](#)
- [createGiottoObject](#), [46](#), [248](#)
- [createHeatmap\\_DT](#), [48](#)
- [createNearestNetwork](#), [49](#)
- [createSpatialEnrich](#), [51](#), [102](#)
- [createSpatialGrid](#), [52](#)
- [createSpatialGrid\\_2D](#), [53](#)
- [createSpatialGrid\\_3D](#), [54](#)
- [createSpatialNetwork](#), [55](#)
- [decide\\_cluster\\_order](#), [59](#)
- [detectSpatialPatterns](#), [60](#)
- [dimCellPlot](#), [61](#)
- [dimCellPlot2D](#), [63](#), [63](#)
- [dimGenePlot](#), [66](#)
- [dimGenePlot2D](#), [68](#)
- [dimGenePlot3D](#), [68](#), [70](#), [70](#)
- [dimPlot](#), [72](#)
- [dimPlot2D](#), [74](#), [75](#), [153](#), [155](#), [159](#), [161](#), [164](#),  
[166](#)
- [dimPlot2D\\_single](#), [78](#)
- [dimPlot3D](#), [66](#), [74](#), [77](#), [80](#), [80](#)
- [direction\\_test \(direction\\_test\\_CPG\)](#), [82](#)
- [direction\\_test\\_CPG](#), [82](#)
- [doHclust](#), [44](#), [83](#)
- [doHMRF](#), [84](#)
- [doKmeans](#), [44](#), [85](#)
- [doLeidenCluster](#), [44](#), [87](#), [90](#), [249](#)
- [doLeidenSubCluster](#), [88](#)
- [doLouvainCluster](#), [44](#), [90](#)

- doLouvainCluster\_community, [44](#), [91](#), [91](#),  
[95](#), [96](#), [249](#)
- doLouvainCluster\_multinet, [44](#), [91](#), [92](#), [95](#),  
[98](#), [249](#)
- doLouvainSubCluster, [93](#)
- doLouvainSubCluster\_community, [95](#)
- doLouvainSubCluster\_multinet, [97](#)
- doRandomWalkCluster, [44](#), [98](#)
- doSNNCluster, [44](#), [100](#)
- dt\_to\_matrix, [101](#)
- exportGiottoViewer, [101](#)
- exprOnlyCellCellcommunicationScores,  
[102](#)
- extended\_gini\_fun, [103](#)
- extractNearestNetwork, [104](#)
- fDataDT, [104](#)
- filterCombinations, [105](#), [108](#)
- filterCPGscores, [106](#)
- filterDistributions, [107](#)
- filterGiotto, [108](#)
- find\_grid\_2D, [118](#)
- find\_grid\_3D, [118](#)
- find\_grid\_x, [118](#)
- find\_grid\_y, [119](#)
- find\_grid\_z, [119](#)
- findGiniMarkers, [109](#), [111](#), [112](#)
- findGiniMarkers\_one\_vs\_all, [110](#), [114](#)
- findMarkers, [111](#), [117](#)
- findMarkers\_one\_vs\_all, [113](#)
- findMastMarkers, [112](#), [114](#), [116](#)
- findMastMarkers\_one\_vs\_all, [114](#), [115](#)
- findScranMarkers, [112](#), [116](#), [118](#)
- findScranMarkers\_one\_vs\_all, [114](#), [117](#)
- fish\_function, [119](#)
- fish\_function2, [119](#)
- FSV\_show, [120](#)
- GenePattern\_show, [121](#)
- general\_save\_function, [122](#)
- get10Xmatrix, [123](#)
- get\_cell\_to\_cell\_sorted\_name\_conversion,  
[128](#)
- get\_interaction\_gene\_enrichment, [129](#)
- get\_specific\_interaction\_gene\_enrichment,  
[129](#)
- getCellProximityGeneScores, [123](#)
- getClusterSimilarity, [125](#)
- getDendrogramSplits, [126](#)
- getDistinctColors, [127](#)
- getGeneToGeneScores, [127](#)
- ggplot\_save\_function, [130](#)
- giotto (giotto-class), [131](#)
- giotto-class, [131](#)
- Giotto::general\_save\_function, [15](#)
- glouvain\_ml, [93](#)
- hclust, [84](#)
- hyperGeometricEnrich, [52](#), [132](#)
- kmeans, [86](#)
- kmeans\_binarize, [133](#)
- kNN, [50](#)
- layout\_with\_drl, [11](#)
- loadHMRF, [133](#)
- make\_simulated\_network, [135](#)
- makeSignMatrixPAGE, [134](#)
- makeSignMatrixRank, [134](#)
- mergeClusters, [135](#)
- mygini\_fun, [136](#)
- nnDT\_to\_kNN, [137](#)
- node\_clusters, [137](#)
- normalizeGiotto, [138](#)
- OR\_function2, [139](#)
- PAGEEnrich, [52](#), [134](#), [139](#)
- PCA, [177](#)
- pDataDT, [140](#)
- permutationPA, [195](#)
- plot\_network\_layer\_ggplot, [168](#)
- plot\_point\_layer\_ggplot, [169](#)
- plot\_spat\_point\_layer\_ggplot, [171](#)
- plotCPGscores, [141](#)
- plotGTGscores, [142](#)
- plotHeatmap, [143](#)
- plotly\_axis\_scale\_2D, [145](#)
- plotly\_axis\_scale\_3D, [146](#)
- plotly\_grid, [146](#)
- plotly\_network, [147](#)
- plotMetaDataCellsHeatmap, [148](#), [151](#)
- plotMetaDataHeatmap, [149](#), [150](#)
- plotPCA, [152](#)
- plotPCA\_2D, [154](#)
- plotPCA\_3D, [153](#), [155](#), [156](#)
- plotTSNE, [157](#)
- plotTSNE\_2D, [159](#)
- plotTSNE\_3D, [159](#), [161](#), [161](#)
- plotUMAP, [163](#)
- plotUMAP\_2D, [165](#)
- plotUMAP\_3D, [164](#), [166](#), [167](#)
- print.giotto, [173](#)

- rank\_binarize, [174](#)
- rankEnrich, [52](#), [135](#), [173](#)
- readGiottoInstructions, [174](#)
- removeCellAnnotation, [175](#)
- removeGeneAnnotation, [175](#)
- replaceGiottoInstructions, [176](#)
- Rtsne, [178](#)
- runPCA, [176](#)
- runSNE, [177](#)
- runUMAP, [179](#)
- select\_expression\_values, [181](#)
- selectPatternGenes, [180](#)
- show,giotto-method, [181](#)
- showClusterDendrogram, [182](#)
- showClusterHeatmap, [183](#)
- showCPGscores, [184](#)
- showGeneExpressionProximityScore, [185](#)
- showGiottoInstructions, [186](#)
- showGTGscores, [186](#)
- showIntExpressionProximityScore, [187](#)
- showPattern, [188](#)
- showPattern2D, [189](#), [189](#)
- showPattern3D, [190](#)
- showPatternGenes, [192](#)
- showProcessingSteps, [193](#)
- showTopGeneToGene, [193](#)
- signPCA, [194](#)
- sNN, [50](#)
- sNNclust, [100](#)
- spatCellPlot, [195](#)
- spatCellPlot2D, [198](#)
- spatDimCellPlot, [201](#)
- spatDimCellPlot2D, [205](#)
- spatDimGenePlot, [209](#)
- spatDimGenePlot2D, [211](#)
- spatDimGenePlot3D, [211](#), [213](#), [213](#)
- spatDimPlot, [216](#)
- spatDimPlot2D, [219](#), [220](#)
- spatDimPlot3D, [219](#), [223](#), [224](#)
- spatGenePlot, [227](#)
- spatGenePlot2D, [228](#), [229](#)
- spatGenePlot3D, [228](#), [230](#), [231](#)
- Spatial\_AEH, [232](#)
- Spatial\_DE, [233](#)
- spatPlot, [234](#)
- spatPlot2D, [237](#), [252](#)
- spatPlot2D\_single, [240](#)
- spatPlot3D, [237](#), [240](#), [243](#), [243](#), [253](#)
- specificCellCellcommunicationScores, [245](#)
- split\_dendrogram\_in\_two, [246](#)
- stitchFieldCoordinates, [48](#), [247](#)
- subClusterCells, [248](#)
- subsetGiotto, [250](#)
- subsetGiottoLocs, [250](#)
- umap, [180](#)
- viewHMRResults, [251](#)
- viewHMRResults2D, [252](#)
- viewHMRResults3D, [253](#)
- violinPlot, [254](#)
- visDimGenePlot, [255](#)
- visDimGenePlot\_2D\_ggplot, [257](#)
- visDimGenePlot\_3D\_plotly, [258](#)
- visDimPlot, [260](#)
- visDimPlot\_2D\_ggplot, [262](#)
- visDimPlot\_2D\_plotly, [264](#)
- visDimPlot\_3D\_plotly, [266](#)
- visForceLayoutPlot, [267](#)
- visGenePlot, [269](#)
- visGenePlot\_2D\_ggplot, [271](#)
- visGenePlot\_3D\_plotly, [272](#)
- visPlot, [252](#), [274](#)
- visPlot\_2D\_ggplot, [276](#)
- visPlot\_2D\_plotly, [279](#)
- visPlot\_3D\_plotly, [280](#)
- visSpatDimGenePlot, [282](#)
- visSpatDimGenePlot\_2D, [285](#)
- visSpatDimGenePlot\_3D, [287](#)
- visSpatDimPlot, [289](#)
- visSpatDimPlot\_2D, [291](#)
- visSpatDimPlot\_3D, [294](#)
- write\_giotto\_viewer\_annotation, [296](#)
- write\_giotto\_viewer\_dim\_reduction, [297](#)
- write\_giotto\_viewer\_numeric\_annotation, [298](#)
- writeHMRResults, [296](#)
- zlm, [115](#)