Package: MOSuite (via r-universe)

April 23, 2026

Title R package for differential multi-omics analysis

Version 0.3.0.9001

Description Multi-Omics Suite provides a suite of functions to clean, filter, batch-correct, normalize, visualize, and perform differential analysis. While the package is designed for differential RNA-seq analysis and multi-omics datasets, it can be used for any data represented in a counts table. See the website for more information, documentation, and examples at <https://ccbr.github.io/MOSuite/>.

License MIT + file LICENSE

URL https://github.com/CCBR/MOSuite, https://ccbr.github.io/MOSuite/

BugReports https://github.com/CCBR/MOSuite/issues

Depends R (>= 4.0.0)

Imports assertthat, dendextend, DESeq2, dplyr, edgeR, ggplot2 (< 4.0.0), glue, htmlwidgets, jsonlite, limma, matrixStats, methods, options, plotly, purrr, reshape2, rlang, S7, stats, stringr, tibble, tidyr, tidyselect

Suggests amap, argparse, broom, cffr, colorspace, ComplexHeatmap, dendsort, docopt, EnhancedVolcano, ggrepel, gridExtra, knitr, lobstr, patchwork, plotrix, RColorBrewer, Rd2md, readr, rmarkdown, roxygen2, scales, styler, sva, testthat (>= 3.0.0), UpSetR, usethis, V8, VennDiagram, withr

VignetteBuilder knitr

Config/Needs/dev cffr, covr, goodpractice, here, lintr, pkgdown, rcmdcheck, xml2

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs make libicu-dev libssl-dev zlib1g-dev

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

Date/Publication 2026-04-23 14:49:54 UTC

RemoteUrl https://github.com/CCBR/MOSuite

RemoteRef HEAD

RemoteSha 0f9674a3e030ba9aac130a9a4be53311beb92086

Contents

`batch_correct_counts` *Perform batch correction*

Description

Perform batch correction using sva::ComBat()

Usage

```
batch_correct_counts(
  moo,
  count_type = "norm",
  sub_count_type = "voom",
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  samples_to_include = NULL,
  covariates_colnames = "Group",
  batch_colname = "Batch",
  label_colname = NULL,
  colors_for_plots = NULL,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "batch"
)
```

Arguments

moo	multiOmicDataSet object (see create_multiOmicDataSet_from_dataframes())
count_type	the type of counts to use – must be a name in the counts slot (moo@counts)
sub_count_type	if count_type is a list, specify the sub count type within the list. (Default: "voom")
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: NULL - first column in the sample metadata will be used.)
feature_id_colname	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)
samples_to_include	Which samples would you like to include? Usually, you will choose all sample columns, or you could choose to remove certain samples. Samples excluded here will be removed in this step and from further analysis downstream of this step. (Default: NULL - all sample IDs in moo@sample_meta will be used.)
covariates_colnames	The column name(s) from the sample metadata containing variable(s) of interest, such as phenotype. Most commonly this will be the same column selected for your Groups Column. Some experimental designs may require that you add additional covariate columns here. Do not include the batch_colname here.
batch_colname	The column from the sample metadata containing the batch information. Samples extracted, prepared, or sequenced at separate times or using separate materials/staff/equipment may belong to different batches. Not all data sets have batches, in which case you do not need batch correction. If your data set has no batches, you can provide a batch column with the same value in every row to skip batch correction (alternatively, simply do not run this function).
label_colname	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: NULL – sample_id_colname will be used.)
colors_for_plots	Colors for the PCA and histogram will be picked, in order, from this list. Colors must either be names in grDevices::colors() or valid hex codes.
print_plots	Whether to print plots during analysis (Defaults to FALSE, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')
save_plots	Whether to save plots to files during analysis (Defaults to TRUE, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir subdirectory in figures/ where plots will be saved if save_plots is TRUE

Value

multiOmicDataSet with batch-corrected counts

See Also

Other moo methods: clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = as.data.frame(nidap_raw_counts),
    "clean" = as.data.frame(nidap_clean_raw_counts),
    "filt" = as.data.frame(nidap_filtered_counts),
    "norm" = list(
      "voom" = as.data.frame(nidap_norm_counts)
    )
  )
) |>
  batch_correct_counts(
    count_type = "norm",
    sub_count_type = "voom",
    covariates_colnames = "Group",
    batch_colname = "Batch",
    label_colname = "Label"
  )

head(moo@counts[["batch"]])
```

---

bind_dfs_long *Bind dataframes in named list to long dataframe*

Description

The dataframes must have all of the same columns

Usage

```
bind_dfs_long(df_list, outcolname = contrast)
```

Arguments

df_list	named list of dataframes
outcolname	column name in output dataframe for the names from the named list

Value

long dataframe with new column outcolname from named list

Examples

```
dfs <- list(
  "a_vs_b" = data.frame(id = c("a1", "b2", "c3"), score = runif(3)),
  "b_vs_c" = data.frame(id = c("a1", "b2", "c3"), score = rnorm(3))
)
dfs |> bind_dfs_long()
```

---

calc_cpm *Calculate counts-per-million (CPM) on raw counts in a multiOmicDataSet*

Description

Calculate counts-per-million (CPM) on raw counts in a multiOmicDataSet

Usage

```
calc_cpm(moo, ...)
```

Arguments

moo	multiOmicDataSet object
...	additional arguments to pass to edgeR::cpm()

Value

multiOmicDataSet with cpm-transformed counts

Examples

```
sample_meta <- data.frame(
  sample_id = c("KO_S3", "KO_S4", "WT_S1", "WT_S2"),
  condition = factor(
    c("knockout", "knockout", "wildtype", "wildtype"),
    levels = c("wildtype", "knockout")
  )
)
moo <- create_multiOmicDataSet_from_dataframes(sample_meta, gene_counts) |>
  calc_cpm()
head(moo@counts$cpm)
```

`calc_pca` *Perform principal components analysis*

Description

Perform principal components analysis

Usage

```
calc_pca(
  counts_dat,
  sample_metadata,
  sample_id_colname = NULL,
  feature_id_colname = NULL
)
```

Arguments

`counts_dat`	data frame of feature counts (e.g. from the counts slot of a `multiOmicDataSet`).
`sample_metadata`	sample metadata as a data frame or tibble.
`sample_id_colname`	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)
`feature_id_colname`	The column from the counts dataa containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)

Value

data frame with statistics for each principal component

See Also

Other PCA functions: `plot_pca()`, `plot_pca_2d()`, `plot_pca_3d()`

Examples

```
calc_pca(nidap_raw_counts, nidap_sample_metadata) |> head()
```

clean_raw_counts *Clean Raw Counts*

Description

This function checks the input raw counts matrix for common formatting problems with feature identifiers and sample names. If feature IDs contain multiple IDs separated by special characters (| - , or space) they will be split into multiple columns. If duplicate feature IDs are detected the counts are summed across duplicate feature ID rows within each sample. Invalid sample names will also be reported and can be automatically corrected. If your sample names are corrected here, be sure to make equivalent changes to your metadata table.

Usage

```
clean_raw_counts(
  moo,
  count_type = "raw",
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  samples_to_rename = "",
  cleanup_column_names = TRUE,
  split_gene_name = TRUE,
  aggregate_rows_with_duplicate_gene_names = TRUE,
  gene_name_column_to_use_for_collapsing_duplicates = "",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "clean"
)
```

Arguments

moo	multiOmicDataSet object (see `create_multiOmicDataSet_from_dataframes()`)
count_type	the type of counts to use – must be a name in the counts slot (`moo@counts`)
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)
feature_id_colname	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)
samples_to_rename	If you do not have a Plot Labels Column in your sample metadata table, you can use this parameter to rename samples manually for display on the PCA plot.

Use "Add item" to add each additional sample for renaming. Use the following format to describe which old name (in your sample metadata table) you want to rename to which new name: old_name: new_name

`cleanup_column_names`
: Invalid raw counts column names can cause errors in the downstream analysis. If this is `TRUE`, any invalid column names will be automatically altered to a correct format. These format changes will include adding an "X" as the first character in any column name that began with a numeral and replacing some special characters ("-,:. ") with underscores ("_"). Invalid sample names and any changes made will be detailed.

`split_gene_name`
: If `TRUE`, split the gene name column by any of these special characters: `,|_-:`

`aggregate_rows_with_duplicate_gene_names`
: If a Feature ID (from the "Cleanup Column Names" parameter above) is found to be duplicated on multiple rows of the raw counts, the Log will report these Feature IDs. Using the default behavior (`TRUE`), the counts for all rows with a duplicate Feature IDs are aggregated into a single row. Counts are summed across duplicate Feature ID rows within each sample. Additional identifier columns, if present (e.g. Ensembl IDs), will be preserved and multiple matching identifiers in such additional columns will appear as comma-separated values in an aggregated row.

`gene_name_column_to_use_for_collapsing_duplicates`
: Select the column with Feature IDs to use as grouping elements to collapse the counts matrix. The log output will list the columns available to identify duplicate row IDs in order to aggregate information. If left blank your "Feature ID" Column will be used to Aggregate Rows. If "Feature ID" column can be split into multiple IDs the non Ensembl ID name will be used to aggregate duplicate IDs. If "Feature ID" column does not contain Ensembl IDs the split Feature IDs will be named 'Feature_id_1' and 'Feature_id_2'. For this case an error will occur and you will have to manually enter the Column ID for this field.

`print_plots`
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

`save_plots`
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

`plots_subdir`
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Value

`multiOmicDataSet` with cleaned counts

See Also

Other moo methods: `batch_correct_counts()`, `diff_counts()`, `filter_counts()`, `filter_diff()`, `normalize_counts()`, `plot_corr_heatmap()`, `plot_expr_heatmap()`, `plot_histogram()`, `plot_pca()`, `plot_read_depth()`, `run_deseq2()`, `set_color_pal()`

Examples

```
moo <- create_multiOmicDataSet_from_dataframes(
  as.data.frame(nidap_sample_metadata),
  as.data.frame(nidap_raw_counts),
  sample_id_colname = "Sample",
) |>
  clean_raw_counts(sample_id_colname = "Sample", feature_id_colname = "GeneName")
head(moo@counts$clean)
```

`create_multiOmicDataSet_from_dataframes`
Construct a multiOmicDataSet object from data frames

Description

Construct a multiOmicDataSet object from data frames

Usage

```
create_multiOmicDataSet_from_dataframes(
  sample_metadata,
  counts_dat,
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  count_type = "raw"
)
```

Arguments

`sample_metadata`	sample metadata as a data frame or tibble. The first column is assumed to contain the sample IDs which must correspond to column names in the raw counts.
`counts_dat`	data frame of feature counts (e.g. expected feature counts from RSEM).
`sample_id_colname`	name of the column in `sample_metadata` that contains the sample IDs. (Default: `NULL` - first column in the sample metadata will be used.)
`feature_id_colname`	name of the column in `counts_dat` that contains feature/gene IDs. (Default: `NULL` - first column in the count data will be used.)
`count_type`	type to assign the values of `counts_dat` to in the `counts` slot

Value

multiOmicDataSet object

See Also

Other moo constructors: create_multiOmicDataSet_from_files(), multiOmicDataSet()

Examples

```
sample_meta <- data.frame(
  sample_id = c("KO_S3", "KO_S4", "WT_S1", "WT_S2"),
  condition = factor(
    c("knockout", "knockout", "wildtype", "wildtype"),
    levels = c("wildtype", "knockout")
  )
)
moo <- create_multiOmicDataSet_from_dataframes(sample_meta, gene_counts)
head(moo@sample_meta)
head(moo@counts$raw)
head(moo@annotation)

sample_meta_nidap <- readr::read_csv(system.file("extdata", "nidap",
  "Sample_Metadata_Bulk_RNA-seq_Training_Dataset_CCBR.csv.gz",
  package = "MOSuite"
))
raw_counts_nidap <- readr::read_csv(system.file("extdata", "nidap", "Raw_Counts.csv.gz",
  package = "MOSuite"
))
moo_nidap <- create_multiOmicDataSet_from_dataframes(sample_meta_nidap, raw_counts_nidap)
```

---

create_multiOmicDataSet_from_files

Construct a multiOmicDataSet object from text files (e.g. TSV, CSV).

Description

Construct a multiOmicDataSet object from text files (e.g. TSV, CSV).

Usage

```
create_multiOmicDataSet_from_files(
  sample_meta_filepath,
  feature_counts_filepath,
  count_type = "raw",
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  delim = NULL,
  ...
)
```

Arguments

sample_meta_filepath	path to text file with sample IDs and metadata for differential analysis.
feature_counts_filepath	path to text file of expected feature counts (e.g. gene counts from RSEM).
count_type	type to assign the values of counts_dat to in the counts slot
sample_id_colname	name of the column in sample_metadata that contains the sample IDs. (Default: NULL - first column in the sample metadata will be used.)
feature_id_colname	name of the column in counts_dat that contains feature/gene IDs. (Default: NULL - first column in the count data will be used.)
delim	Delimiter used in the input files. Any delimiter accepted by readr::read_delim() can be used. If the files are in CSV format, set delim = ','; for TSV format, set delim = '\t'.
...	additional arguments forwarded to readr::read_delim().

Value

multiOmicDataSet object

See Also

Other moo constructors: create_multiOmicDataSet_from_dataframes(), multiOmicDataSet()

Examples

```
moo <- create_multiOmicDataSet_from_files(
  sample_meta_filepath = system.file("extdata",
    "sample_metadata.tsv.gz",
    package = "MOSuite"
  ),
  feature_counts_filepath = system.file("extdata",
    "RSEM.genes.expected_count.all_samples.txt.gz",
    package = "MOSuite"
  ),
  delim = "\t"
)
moo@counts$raw |> head()
moo@sample_meta

moo_nidap <- create_multiOmicDataSet_from_files(
  system.file("extdata", "nidap",
    "Sample_Metadata_Bulk_RNA-seq_Training_Dataset_CCBR.csv.gz",
    package = "MOSuite"
  ),
  system.file("extdata", "nidap", "Raw_Counts.csv.gz", package = "MOSuite"),
  delim = ","
)
```

diff_counts *Differential expression analysis*

Description

Differential expression analysis

Usage

```
diff_counts(
  moo,
  count_type = "filt",
  sub_count_type = NULL,
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  samples_to_include = NULL,
  covariates_colnames = NULL,
  contrast_colname = NULL,
  contrasts = NULL,
  input_in_log_counts = FALSE,
  return_mean_and_sd = FALSE,
  voom_normalization_method = "quantile",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "diff"
)
```

Arguments

moo	multiOmicDataSet object (see create_multiOmicDataSet_from_dataframes())
count_type	the type of counts to use – must be a name in the counts slot (moo@counts)
sub_count_type	if count_type is a list, specify the sub count type within the list. (Default: NULL)
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: NULL - first column in the sample metadata will be used.)
feature_id_colname	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)
samples_to_include	Which samples would you like to include? Usually, you will choose all sample columns, or you could choose to remove certain samples. Samples excluded

here will be removed in this step and from further analysis downstream of this step. (Default: NULL - all sample IDs in moo@sample_meta will be used.)

covariates_colnames
: Columns to be used as covariates in linear modeling. Must include column from "Contrast Variable". Most commonly your covariate will be group and batch (if you have different batches in your data).

contrast_colname
: The column in the metadata that contains the group variables you wish to find differential expression between. Up to 2 columns (2-factor analysis) can be used.

contrasts
: Specify each contrast in the format group1-group2, e.g. treated-control

input_in_log_counts
: set this to TRUE if counts are already log2-transformed

return_mean_and_sd
: if TRUE, return Mean and Standard Deviation of groups in addition to DEG estimates for contrast(s)

voom_normalization_method
: Normalization method to be applied to the logCPM values when using limma::voom

print_plots
: Whether to print plots during analysis (Defaults to FALSE, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to TRUE, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir
: subdirectory in figures/ where plots will be saved if save_plots is TRUE

Value

multiOmicDataSet with diff added to the analyses slot (i.e. moo@analyses$diff)

See Also

Other moo methods: batch_correct_counts(), clean_raw_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = as.data.frame(nidap_raw_counts),
    "clean" = as.data.frame(nidap_clean_raw_counts),
    "filt" = as.data.frame(nidap_filtered_counts)
  )
) |>
  diff_counts(
    count_type = "filt",
    sub_count_type = NULL,
```

```
    sample_id_colname = "Sample",
    feature_id_colname = "Gene",
    covariates_colnames = c("Group", "Batch"),
    contrast_colname = c("Group"),
    contrasts = c("B-A", "C-A", "B-C"),
    voom_normalization_method = "quantile",
  )
head(moo@analyses$diff)
```

`extract_counts` *Extract count data*

Description

Extract count data

Usage

```
extract_counts(moo, count_type, sub_count_type = NULL)
```

Arguments

`moo`	multiOmicDataSet containing `count_type` & `sub_count_type` in the counts slot
`count_type`	the type of counts to use – must be a name in the counts slot (`moo@counts[[count_type]]`)
`sub_count_type`	if `count_type` is a list, specify the sub count type within the list (`moo@counts[[count_type]][[sub_cou` (Default: `NULL`)

Examples

```
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = as.data.frame(nidap_raw_counts),
    "clean" = as.data.frame(nidap_clean_raw_counts),
    "filt" = as.data.frame(nidap_filtered_counts),
    "norm" = list(
      "voom" = as.data.frame(nidap_norm_counts)
    )
  )
)

moo |>
  extract_counts("filt") |>
  head()

moo |>
  extract_counts("norm", "voom") |>
```

```
head()
```

`filter_counts` *Filter low counts*

Description

This is often the first step in the QC portion of an analysis to filter out features that have very low raw counts across most or all of your samples.

Usage

```
filter_counts(
  moo,
  count_type = "clean",
  feature_id_colname = NULL,
  sample_id_colname = NULL,
  group_colname = "Group",
  label_colname = NULL,
  samples_to_include = NULL,
  minimum_count_value_to_be_considered_nonzero = 8,
  minimum_number_of_samples_with_nonzero_counts_in_total = 7,
  minimum_number_of_samples_with_nonzero_counts_in_a_group = 3,
  use_cpm_counts_to_filter = TRUE,
  use_group_based_filtering = FALSE,
  principal_component_on_x_axis = 1,
  principal_component_on_y_axis = 2,
  legend_position_for_pca = "top",
  point_size_for_pca = 1,
  add_label_to_pca = TRUE,
  label_font_size = 3,
  label_offset_y_ = 2,
  label_offset_x_ = 2,
  samples_to_rename = c(""),
  color_histogram_by_group = FALSE,
  set_min_max_for_x_axis_for_histogram = FALSE,
  minimum_for_x_axis_for_histogram = -1,
  maximum_for_x_axis_for_histogram = 1,
  legend_position_for_histogram = "top",
  legend_font_size_for_histogram = 10,
  number_of_histogram_legend_columns = 6,
  colors_for_plots = NULL,
  plot_corr_matrix_heatmap = TRUE,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  interactive_plots = FALSE,
```

```
  plots_subdir = "filt"
)
```

Arguments

moo	multiOmicDataSet object (see `create_multiOmicDataSet_from_dataframes()`)
count_type	the type of counts to use – must be a name in the counts slot (`moo@counts`)
feature_id_colname	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)
group_colname	The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).
label_colname	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: `NULL` – `sample_id_colname` will be used.)
samples_to_include	Which samples would you like to include? Usually, you will choose all sample columns, or you could choose to remove certain samples. Samples excluded here will be removed in this step and from further analysis downstream of this step. (Default: `NULL` - all sample IDs in `moo@sample_meta` will be used.)
minimum_count_value_to_be_considered_nonzero	Minimum count value to be considered non-zero for a sample
minimum_number_of_samples_with_nonzero_counts_in_total	Minimum number of samples (total) with non-zero counts
minimum_number_of_samples_with_nonzero_counts_in_a_group	Only keeps genes that have at least this number of samples with nonzero CPM counts in at least one group
use_cpm_counts_to_filter	If no transformation has been been performed on counts matrix (eg Raw Counts) set to TRUE. If TRUE counts will be transformed to CPM and filtered based on given criteria. If gene counts matrix has been transformed (eg log2, CPM, FPKM or some form of Normalization) set to FALSE. If FALSE no further transformation will be applied and features will be filtered as is. For RNAseq data RAW counts should be transformed to CPM in order to properly filter.

use_group_based_filtering
: If TRUE, only keeps features (e.g. genes) that have at least a certain number of samples with nonzero CPM counts in at least one group

principal_component_on_x_axis
: The principal component to plot on the x-axis for the PCA plot. Choices include 1, 2, 3, ... (default: 1)

principal_component_on_y_axis
: The principal component to plot on the y-axis for the PCA plot. Choices include 1, 2, 3, ... (default: 2)

legend_position_for_pca
: legend position for the PCA plot

point_size_for_pca
: geom point size for the PCA plot

add_label_to_pca
: label points on the PCA plot

label_font_size
: label font size for the PCA plot

label_offset_y_
: label offset y for the PCA plot

label_offset_x_
: label offset x for the PCA plot

samples_to_rename
: If you do not have a Plot Labels Column in your sample metadata table, you can use this parameter to rename samples manually for display on the PCA plot. Use "Add item" to add each additional sample for renaming. Use the following format to describe which old name (in your sample metadata table) you want to rename to which new name: old_name: new_name

color_histogram_by_group
: Set to FALSE to label histogram by Sample Names, or set to TRUE to label histogram by the column you select in the "Group Column Used to Color Histogram" parameter (below). Default is FALSE.

set_min_max_for_x_axis_for_histogram
: whether to set min/max value for histogram x-axis

minimum_for_x_axis_for_histogram
: x-axis minimum for histogram plot

maximum_for_x_axis_for_histogram
: x-axis maximum for histogram plot

legend_position_for_histogram
: legend position for the histogram plot. consider setting to 'none' for a large number of samples.

legend_font_size_for_histogram
: legend font size for the histogram plot

number_of_histogram_legend_columns
: number of columns for the histogram legend

colors_for_plots
: Colors for the PCA and histogram will be picked, in order, from this list. Colors must either be names in `grDevices::colors()` or valid hex codes.

plot_corr_matrix_heatmap
: Datasets with a large number of samples may be too large to create a correlation matrix heatmap. If this function takes longer than 5 minutes to run, Set to `FALSE` and the correlation matrix will not be be created. Default is `TRUE`.

print_plots
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

interactive_plots
: set to TRUE to make PCA and Histogram plots interactive with `plotly`, allowing you to hover your mouse over a point or line to view sample information. The similarity heat map will not display if this toggle is set to `TRUE`. Default is `FALSE`.

plots_subdir
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Details

This function takes a multiOmicDataSet containing clean raw counts and a sample metadata table, and returns the multiOmicDataSet object with filtered counts. It also produces an image consisting of three QC plots.

You can tune the threshold for tuning how low counts for a given gene are before they are deemed "too low" and filtered out of downstream analysis. By default, this parameter is set to 1, meaning any raw count value less than 1 will count as "too low".

The QC plots are provided to help you assess: (1) PCA Plot: the within and between group variance in expression after dimensionality reduction; (2) Count Density Histogram: the dis/similarity of count distributions between samples; and (3) Similarity Heatmap: the overall similarity of samples to one another based on unsupervised clustering.

Value

`multiOmicDataSet` with filtered counts

See Also

Other moo methods: `batch_correct_counts()`, `clean_raw_counts()`, `diff_counts()`, `filter_diff()`, `normalize_counts()`, `plot_corr_heatmap()`, `plot_expr_heatmap()`, `plot_histogram()`, `plot_pca()`, `plot_read_depth()`, `run_deseq2()`, `set_color_pal()`

Examples

```
moo <- create_multiOmicDataSet_from_dataframes(
  as.data.frame(nidap_sample_metadata),
  as.data.frame(nidap_clean_raw_counts),
  sample_id_colname = "Sample",
  feature_id_colname = "Gene"
```

```
) |>
  filter_counts(
    count_type = "raw"
  )
head(moo@counts$filt)
```

`filter_diff` *Filter features from differential analysis based on statistical significance*

Description

Outputs dataset of significant genes from DEG table; filters genes based on statistical significance (p-value or adjusted p-value) and change (fold change, log2 fold change, or t-statistic); in addition allows for selection of DEG estimates and for sub-setting of contrasts and groups included in the output gene list.

Usage

```
filter_diff(
  moo,
  feature_id_colname = NULL,
  significance_column = "adjpval",
  significance_cutoff = 0.05,
  change_column = "logFC",
  change_cutoff = 1,
  filtering_mode = "any",
  include_estimates = c("FC", "logFC", "tstat", "pval", "adjpval"),
  round_estimates = TRUE,
  rounding_decimal_for_percent_cells = 0,
  contrast_filter = "none",
  contrasts = c(),
  groups = c(),
  groups_filter = "none",
  label_font_size = 6,
  label_distance = 1,
  y_axis_expansion = 0.08,
  fill_colors = c("steelblue1", "whitesmoke"),
  pie_chart_in_3d = TRUE,
  bar_width = 0.4,
  draw_bar_border = TRUE,
  plot_type = "bar",
  plot_titles_fontsize = 12,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = file.path("diff", "filt")
)
```

Arguments

moo	multiOmicDataSet object (see create_multiOmicDataSet_from_dataframes())
feature_id_colname	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)
significance_column	Column name for significance, e.g. "pval" or "pvaladj" (default)
significance_cutoff	Features will only be kept if their significance_column is less then this cutoff threshold
change_column	Column name for change, e.g. "logFC" (default)
change_cutoff	Features will only be kept if the absolute value of their change_column is greater than or equal to this cutoff threshold
filtering_mode	Accepted values: "any" or "all" to include features that meet the criteria in *any* contrast or in *all* contrasts
include_estimates	Column names of estimates to include. Default: c("FC", "logFC", "tstat", "pval", "adjpval")
round_estimates	Whether to round estimates. Default: TRUE
rounding_decimal_for_percent_cells	Decimal place to use when rounding Percent cells
contrast_filter	Whether to filter contrasts in or our of analysis. If "keep", only the contrast names listed in contrasts will be included. If "remove, the contrast names listed by contrasts will be removed. If "none", all contrasts in the dataset are used. Options: "keep", "remove", or "none"
contrasts	Contrast names to filter by contrast_filter. If contrast_filter is "none", this parameter has no effect.
groups	Group names to filter by groups_filter. If groups_filter is "none", this parameter has no effect. Options: "keep", "remove", or "none"
groups_filter	Whether to filter groups in or out of analysis. If "keep", only the group names listed in groups will be included. If "remove", the group names listed by groups will be removed. If "none", all groups in the dataset are used.
label_font_size	Font size for labels in the plot (default: 6)
label_distance	Distance of labels from the bars (default: 1)
y_axis_expansion	Expansion of the y-axis (default: 0.08)
fill_colors	Fill colors for the bars (default: c("steelblue1", "whitesmoke"))

pie_chart_in_3d
: Whether to draw pie charts in 3D (default: TRUE)

bar_width
: Width of the bars (default: 0.4)

draw_bar_border
: Whether to draw borders around bars (default: TRUE)

plot_type
: "bar" or "pie"

plot_titles_fontsize
: Font size for plot titles (default: 12)

print_plots
: Whether to print plots during analysis (Defaults to FALSE, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to TRUE, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir
: subdirectory in where plots will be saved if save_plots is TRUE

See Also

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = as.data.frame(nidap_raw_counts),
    "clean" = as.data.frame(nidap_clean_raw_counts),
    "filt" = as.data.frame(nidap_filtered_counts)
  )
) |>
  diff_counts(
    count_type = "filt",
    sub_count_type = NULL,
    sample_id_colname = "Sample",
    feature_id_colname = "Gene",
    covariates_colnames = c("Group", "Batch"),
    contrast_colname = c("Group"),
    contrasts = c("B-A", "C-A", "B-C"),
    voom_normalization_method = "quantile",
  ) |>
  filter_diff()
head(moo@analyses$diff_filt)
```

gene_counts *RSEM expected gene counts*

Description

RSEM expected gene counts

Usage

```
gene_counts
```

Format

`gene_counts`:
A data frame with columns 'gene_id', 'GeneName', and a column for each sample's expected count.

Source

Generated by running RENEE v2.5.8 on the test dataset

get_colors_lst *Create named list of default colors for plotting*

Description

Create named list of default colors for plotting

Usage

```
get_colors_lst(sample_metadata, palette_fun = grDevices::palette.colors, ...)
```

Arguments

`sample_metadata`	sample metadata as a data frame or tibble. The first column is assumed to contain the sample IDs which must correspond to column names in the raw counts.
`palette_fun`	Function for selecting colors. Assumed to contain `n` for the number of colors. Default: `grDevices::palette.colors()`
`...`	additional arguments forwarded to `palette_fun`

Value

named list, with each column in `sample_metadata` containing entry with a named vector of colors

Examples

```
get_colors_lst(nidap_sample_metadata)
## Not run:
get_colors_lst(nidap_sample_metadata, palette_fun = RColorBrewer::brewer.pal, name = "Set3")

## End(Not run)
```

`get_colors_vctr` *Get vector of colors for observations in one column of a data frame*

Description

Get vector of colors for observations in one column of a data frame

Usage

```
get_colors_vctr(dat, colname, palette_fun = grDevices::palette.colors, ...)
```

Arguments

`dat`	data frame
`colname`	column name in `dat`
`palette_fun`	Function for selecting colors. Assumed to contain `n` for the number of colors. Default: `grDevices::palette.colors()`
`...`	additional arguments forwarded to `palette_fun`

Value

named vector of colors for each unique observation in `dat$colname`

`join_dfs_wide` *Join dataframes in named list to wide dataframe*

Description

The first column is assumed to be shared by all dataframes

Usage

```
join_dfs_wide(df_list, join_fn = dplyr::left_join)
```

Arguments

`df_list`	named list of dataframes
`join_fn`	join function to use (Default: `dplyr::left_join`)

Value

wide dataframe

Examples

```
dfs <- list(
  "a_vs_b" = data.frame(id = c("a1", "b2", "c3"), score = runif(3)),
  "b_vs_c" = data.frame(id = c("a1", "b2", "c3"), score = rnorm(3))
)
dfs |> join_dfs_wide()
```

---

multiOmicDataSet *multiOmicDataSet class*

Description

multiOmicDataSet class

Usage

```
multiOmicDataSet(sample_metadata, anno_dat, counts_lst, analyses_lst = list())
```

Arguments

sample_metadata	sample metadata as a data frame or tibble. The first column is assumed to contain the sample IDs which must correspond to column names in the raw counts.
anno_dat	data frame of feature annotations, such as gene symbols or any other information about the features in counts_lst.
counts_lst	named list of data frames containing counts, e.g. expected feature counts from RSEM. Each data frame is expected to contain a feature_id column as the first column, and all remaining columns are sample IDs in the sample_meta.
analyses_lst	named list of analysis results, e.g. DESeq results object

See Also

Other moo constructors: create_multiOmicDataSet_from_dataframes(), create_multiOmicDataSet_from_files()

`nidap_batch_corrected_counts`
Batch-corrected counts for the NIDAP test dataset.

Description

Batch-corrected counts for the NIDAP test dataset.

Usage

```
nidap_batch_corrected_counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7943 rows and 10 columns.

`nidap_batch_corrected_counts_2`
Batch-corrected counts for the NIDAP test dataset. The result of running `batch_correct_counts()` *on* `nidap_norm_counts`.

Description

Batch-corrected counts for the NIDAP test dataset. The result of running `batch_correct_counts()` on `nidap_norm_counts`.

Usage

```
nidap_batch_corrected_counts_2
```

Format

An object of class `data.frame` with 7943 rows and 10 columns.

nidap_clean_raw_counts *Clean raw counts for the NIDAP test dataset. The result of running* `clean_raw_counts()` *on* `nidap_raw_counts`.

Description

Clean raw counts for the NIDAP test dataset. The result of running `clean_raw_counts()` on `nidap_raw_counts`.

Usage

```
nidap_clean_raw_counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 43280 rows and 10 columns.

nidap_deg_analysis *Differential gene expression analysis for the NIDAP test dataset.*

Description

Differential gene expression analysis for the NIDAP test dataset.

Usage

```
nidap_deg_analysis
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7943 rows and 25 columns.

nidap_deg_analysis_2 *Differential gene expression analysis for the NIDAP test dataset. The result of running* `diff_counts()` *on* `nidap_filtered_counts`.

Description

Differential gene expression analysis for the NIDAP test dataset. The result of running `diff_counts()` on `nidap_filtered_counts`.

Usage

```
nidap_deg_analysis_2
```

Format

An object of class `list` of length 3.

nidap_deg_gene_list *List of differentially expressed genes from the NIDAP test dataset using default parameters with* `filter_diff()`.

Description

List of differentially expressed genes from the NIDAP test dataset using default parameters with `filter_diff()`.

Usage

```
nidap_deg_gene_list
```

Format

An object of class `data.frame` with 641 rows and 16 columns.

nidap_filtered_counts *Filtered counts for the NIDAP test dataset. The result of running* `filter_counts()` *on* `nidap_clean_raw_counts`.

Description

Filtered counts for the NIDAP test dataset. The result of running `filter_counts()` on `nidap_clean_raw_counts`.

Usage

```
nidap_filtered_counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7943 rows and 10 columns.

nidap_norm_counts *Normalized counts for the NIDAP test dataset. The result of running* `normalize_counts()` *on* `nidap_filtered_counts`.

Description

Normalized counts for the NIDAP test dataset. The result of running `normalize_counts()` on `nidap_filtered_counts`.

Usage

```
nidap_norm_counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7943 rows and 10 columns.

nidap_raw_counts *Raw counts for the NIDAP test dataset Pairs with* `nidap_sample_metadata`.

Description

Raw counts for the NIDAP test dataset Pairs with `nidap_sample_metadata`.

Usage

```
nidap_raw_counts
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 43280 rows and 10 columns.

nidap_sample_metadata *Sample metadata for the NIDAP test dataset*

Description

Sample metadata for the NIDAP test dataset

Usage

```
nidap_sample_metadata
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 9 rows and 5 columns.

`nidap_venn_diagram_dat` *Output data from venn diagram. The result of running* `plot_venn_diagram()` *on* `nidap_volcano_summary_dat`

Description

Output data from venn diagram. The result of running `plot_venn_diagram()` on `nidap_volcano_summary_dat`

Usage

```
nidap_venn_diagram_dat
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 3068 rows and 4 columns.

`nidap_volcano_summary_dat` *Summarized differential expression analysis for input to venn diagram*

Description

Summarized differential expression analysis for input to venn diagram

Usage

```
nidap_volcano_summary_dat
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 4929 rows and 7 columns.

normalize_counts *Normalize counts*

Description

Normalize counts

Usage

```
normalize_counts(
  moo,
  count_type = "filt",
  norm_type = "voom",
  feature_id_colname = NULL,
  samples_to_include = NULL,
  sample_id_colname = NULL,
  group_colname = "Group",
  label_colname = NULL,
  input_in_log_counts = FALSE,
  voom_normalization_method = "quantile",
  samples_to_rename = c(""),
  add_label_to_pca = TRUE,
  principal_component_on_x_axis = 1,
  principal_component_on_y_axis = 2,
  legend_position_for_pca = "top",
  label_offset_x_ = 2,
  label_offset_y_ = 2,
  label_font_size = 3,
  point_size_for_pca = 8,
  color_histogram_by_group = TRUE,
  set_min_max_for_x_axis_for_histogram = FALSE,
  minimum_for_x_axis_for_histogram = -1,
  maximum_for_x_axis_for_histogram = 1,
  legend_font_size_for_histogram = 10,
  legend_position_for_histogram = "top",
  number_of_histogram_legend_columns = 6,
  plot_corr_matrix_heatmap = TRUE,
  colors_for_plots = NULL,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  interactive_plots = FALSE,
  plots_subdir = "norm"
)
```

Arguments

moo	multiOmicDataSet object (see `create_multiOmicDataSet_from_dataframes()`)

count_type the type of counts to use – must be a name in the counts slot (moo@counts)

norm_type normalization type. Default: "voom" which uses limma::voom.

feature_id_colname
: The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)

samples_to_include
: Which samples would you like to include? Usually, you will choose all sample columns, or you could choose to remove certain samples. Samples excluded here will be removed in this step and from further analysis downstream of this step. (Default: NULL - all sample IDs in moo@sample_meta will be used.)

sample_id_colname
: The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: NULL - first column in the sample metadata will be used.)

group_colname The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).

label_colname The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: NULL – sample_id_colname will be used.)

input_in_log_counts
: set this to TRUE if counts are already log2-transformed

voom_normalization_method
: Normalization method to be applied to the logCPM values when using limma::voom

samples_to_rename
: If you do not have a Plot Labels Column in your sample metadata table, you can use this parameter to rename samples manually for display on the PCA plot. Use "Add item" to add each additional sample for renaming. Use the following format to describe which old name (in your sample metadata table) you want to rename to which new name: old_name: new_name

add_label_to_pca
: label points on the PCA plot

principal_component_on_x_axis
: The principal component to plot on the x-axis for the PCA plot. Choices include 1, 2, 3, ... (default: 1)

principal_component_on_y_axis
: The principal component to plot on the y-axis for the PCA plot. Choices include 1, 2, 3, ... (default: 2)

`legend_position_for_pca`
: legend position for the PCA plot

`label_offset_x_`
: label offset x for the PCA plot

`label_offset_y_`
: label offset y for the PCA plot

`label_font_size`
: label font size for the PCA plot

`point_size_for_pca`
: geom point size for the PCA plot

`color_histogram_by_group`
: Set to FALSE to label histogram by Sample Names, or set to TRUE to label histogram by the column you select in the "Group Column Used to Color Histogram" parameter (below). Default is FALSE.

`set_min_max_for_x_axis_for_histogram`
: whether to set min/max value for histogram x-axis

`minimum_for_x_axis_for_histogram`
: x-axis minimum for histogram plot

`maximum_for_x_axis_for_histogram`
: x-axis maximum for histogram plot

`legend_font_size_for_histogram`
: legend font size for the histogram plot

`legend_position_for_histogram`
: legend position for the histogram plot. consider setting to 'none' for a large number of samples.

`number_of_histogram_legend_columns`
: number of columns for the histogram legend

`plot_corr_matrix_heatmap`
: Datasets with a large number of samples may be too large to create a correlation matrix heatmap. If this function takes longer than 5 minutes to run, Set to `FALSE` and the correlation matrix will not be be created. Default is `TRUE`.

`colors_for_plots`
: Colors for the PCA and histogram will be picked, in order, from this list. Colors must either be names in `grDevices::colors()` or valid hex codes.

`print_plots`
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

`save_plots`
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

`interactive_plots`
: set to TRUE to make PCA and Histogram plots interactive with `plotly`, allowing you to hover your mouse over a point or line to view sample information. The similarity heat map will not display if this toggle is set to `TRUE`. Default is `FALSE`.

`plots_subdir`
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Value

multiOmicDataSet with normalized counts

See Also

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = as.data.frame(nidap_raw_counts),
    "clean" = as.data.frame(nidap_clean_raw_counts),
    "filt" = as.data.frame(nidap_filtered_counts)
  )
) |>
  normalize_counts(
    group_colname = "Group",
    label_colname = "Label"
  )
head(moo@counts[["norm"]][["voom"]])
```

---

options *MOSuite Options*

Description

Internally used, package-specific options. All options will prioritize R options() values, and fall back to environment variables if undefined. If neither the option nor the environment variable is set, a default value is used.

Checking Option Values

Option values specific to MOSuite can be accessed by passing the package name to env.

```
options::opts(env = "MOSuite")

options::opt(x, default, env = "MOSuite")
```

Options

print_plots Whether to print plots during analysis

default: `FALSE`

option: moo_print_plots

envvar: MOO_PRINT_PLOTS (evaluated if possible, raw string otherwise)

save_plots Whether to save plots to files during analysis

default: `TRUE`

option: moo_save_plots

envvar: MOO_SAVE_PLOTS (evaluated if possible, raw string otherwise)

plots_dir Path where plots are saved when `moo_save_plots` is `TRUE`

default: `"figures/"`

option: moo_plots_dir

envvar: MOO_PLOTS_DIR (evaluated if possible, raw string otherwise)

See Also

options getOption Sys.setenv Sys.getenv

`plot_corr_heatmap` *Plot correlation heatmap*

Description

Plot correlation heatmap

Usage

```
plot_corr_heatmap(moo_counts, ...)
```

Arguments

`moo_counts`	counts dataframe or `multiOmicDataSet` containing `count_type` & `sub_count_type` in the counts slot
`...`	arguments forwarded to method plot_corr_heatmap_dat

Value

heatmap from `ComplexHeatmap::Heatmap()`

Methods

link to docs	class
plot_corr_heatmap_moo	multiOmicDataSet
plot_corr_heatmap_dat	data.frame

Method Usage:

```
# multiOmicDataSet
plot_corr_heatmap(moo_counts,
  count_type,
  sub_count_type = NULL,
  ...)

# dataframe
plot_corr_heatmap(moo_counts,
  sample_metadata,
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  group_colname = "Group",
  label_colname = "Label",
  color_values = c(
    "#5954d6", "#e1562c", "#b80058", "#00c6f8", "#d163e6", "#00a76c",
    "#ff9287", "#008cf9", "#006e00", "#796880", "#FFA500", "#878500"
  ))
```

See Also

Other plotters: plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), print_or_save_plot()

Other heatmaps: plot_expr_heatmap()

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
# plot correlation heatmap for a counts slot in a multiOmicDataset Object
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list("raw" = as.data.frame(nidap_raw_counts))
)
p <- plot_corr_heatmap(moo, count_type = "raw")

# plot correlation heatmap for a counts dataframe
plot_corr_heatmap(
  moo@counts$raw,
```

```
  sample_metadata = moo@sample_meta,
  sample_id_colname = "Sample",
  feature_id_colname = "Gene",
  group_colname = "Group",
  label_colname = "Label"
)
```

`plot_corr_heatmap_dat` *Plot correlation heatmap for counts dataframe*

Description

Plot correlation heatmap for counts dataframe

Arguments

`moo_counts`	counts dataframe (**Required**)
`sample_metadata`	sample metadata as a data frame or tibble (**Required**)
`sample_id_colname`	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)
`feature_id_colname`	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)
`group_colname`	The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).
`label_colname`	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: `NULL` – `sample_id_colname` will be used.)
`color_values`	vector of colors as hex values or names recognized by R

See Also

plot_corr_heatmap generic

Other plotters for counts dataframes: `plot_histogram_dat`, `plot_pca_dat`, `plot_read_depth_dat`

plot_corr_heatmap_moo *Plot correlation heatmap for multiOmicDataSet*

Description

Plot correlation heatmap for multiOmicDataSet

Arguments

moo_counts	multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	the type of counts to use. Must be a name in the counts slot (names(moo@counts)).
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).
...	arguments forwarded to method plot_corr_heatmap_dat

See Also

plot_corr_heatmap generic

Other plotters for multiOmicDataSets: plot_histogram_moo, plot_pca_moo, plot_read_depth_moo

plot_expr_heatmap *Plot expression heatmap*

Description

The samples (i.e. the columns) are clustered in an unsupervised fashion based on how similar their expression profiles are across the included genes. This can help identify samples that are non clustering with their group as you might expect based on the experimental design.

Usage

```
plot_expr_heatmap(
  moo_counts,
  count_type,
  sub_count_type = NULL,
  sample_metadata = NULL,
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  group_colname = "Group",
  label_colname = NULL,
  samples_to_include = NULL,
 color_values = c("#5954d6", "#e1562c", "#b80058", "#00c6f8", "#d163e6", "#00a76c",
    "#ff9287", "#008cf9", "#006e00", "#796880", "#FFA500", "#878500"),
```

```
  include_all_genes = FALSE,
  filter_top_genes_by_variance = TRUE,
  top_genes_by_variance_to_include = 500,
  specific_genes_to_include_in_heatmap = "None",
  cluster_genes = TRUE,
  gene_distance_metric = "correlation",
  gene_clustering_method = "average",
  display_gene_dendrograms = TRUE,
  display_gene_names = FALSE,
  center_and_rescale_expression = TRUE,
  cluster_samples = FALSE,
  arrange_sample_columns = TRUE,
  order_by_gene_expression = FALSE,
  gene_to_order_columns = " ",
  gene_expression_order = "low_to_high",
  smpl_distance_metric = "correlation",
  smpl_clustering_method = "average",
  display_smpl_dendrograms = TRUE,
  reorder_dendrogram = FALSE,
  reorder_dendrogram_order = c(),
  display_sample_names = TRUE,
  group_columns = c("Group", "Replicate", "Batch"),
  assign_group_colors = FALSE,
  assign_color_to_sample_groups = c(),
 group_colors = c("#5954d6", "#e1562c", "#b80058", "#00c6f8", "#d163e6", "#00a76c",
    "#ff9287", "#008cf9", "#006e00", "#796880", "#FFA500", "#878500"),
  heatmap_color_scheme = "Default",
  autoscale_heatmap_color = TRUE,
  set_min_heatmap_color = -2,
  set_max_heatmap_color = 2,
  aspect_ratio = "Auto",
  legend_font_size = 10,
  gene_name_font_size = 4,
  sample_name_font_size = 8,
  display_numbers = FALSE,
  plot_filename = "expr_heatmap.png",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "heatmap"
)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	the type of counts to use. Must be a name in the counts slot (names(moo@counts)).
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).

`sample_metadata`
: sample metadata as a data frame or tibble (only required if `moo_counts` is a dataframe)

`sample_id_colname`
: The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)

`feature_id_colname`
: The column from the counts dataa containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)

`group_colname`
: The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).

`label_colname`
: The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: `NULL` – `sample_id_colname` will be used.)

`samples_to_include`
: Which samples would you like to include? Usually, you will choose all sample columns, or you could choose to remove certain samples. Samples excluded here will be removed in this step and from further analysis downstream of this step. (Default: `NULL` - all sample IDs in `moo@sample_meta` will be used.)

`color_values`
: vector of colors as hex values or names recognized by R

`include_all_genes`
: Set to TRUE if all genes are to be included. Set to FALSE if you want to filter genes by variance and/or provide a list of specific genes that will appear in the heatmap.

`filter_top_genes_by_variance`
: Set to TRUE if you want to only include the top genes by variance. Set to FALSE if you do not want to filter genes by variance.

`top_genes_by_variance_to_include`
: The number of genes to include if filtering genes by variance. This parameter is ignored if "Filter top genes by variance" is set to FALSE.

`specific_genes_to_include_in_heatmap`
: Enter the gene symbols to be included in the heatmap, with each gene symbol separated with a space from the others. Alternatively, paste in a column of gene names from any spreadsheet application. This parameter is ignored if "Include all genes" is set to TRUE.

`cluster_genes`
: Choose whether to cluster the rows (genes). If TRUE, rows will have clustering applied. If FALSE, clustering will not be applied to rows.

gene_distance_metric
: Distance metric to be used in clustering genes. (TODO document options)

gene_clustering_method
: Clustering method metric to be used in clustering samples. (TODO document options)

display_gene_dendrograms
: Set to TRUE to show gene dendrograms. Set to FALSE to hide dendrograms.

display_gene_names
: Set to TRUE to display gene names on the right side of the heatmap. Set to FALSE to hide gene names.

center_and_rescale_expression
: Center and rescale expression for each gene across all included samples.

cluster_samples
: Choose whether to cluster the columns (samples). If TRUE, columns will have clustering applied. If FALSE, clustering will not be applied to columns.

arrange_sample_columns
: If TRUE, arranges columns by annotation groups. If FALSE, and "Cluster Samples" is FALSE, samples will appear in the order of input (samples to include)

order_by_gene_expression
: If TRUE, set gene name below and direction for ordering

gene_to_order_columns
: Gene to order columns by expression levels

gene_expression_order
: Choose direction for gene order

smpl_distance_metric
: Distance metric to be used in clustering samples. (TODO document options)

smpl_clustering_method
: Clustering method to be used in clustering samples. (TODO document options)

display_smpl_dendrograms
: Set to TRUE to show sample dendrograms. Set to FALSE to hide dendrogram.

reorder_dendrogram
: If TRUE, set the order of the dendrogram (below)

reorder_dendrogram_order
: Reorder the samples (columns) of the dendrogram by name, e.g. “sample2”,“sample3",“sample1".

display_sample_names
: Set to TRUE if you want sample names to be displayed on the plot. Set to FALSE to hide sample names.

group_columns
: Columns containing the sample groups for annotation tracks

assign_group_colors
: If TRUE, set the groups assigned colors (below)

assign_color_to_sample_groups
: Enter each sample to color in the format: group_name: color This parameter is ignored if "Assign Colors" is set to FALSE.

group_colors
: Set group annotation colors.

`heatmap_color_scheme`
: color scheme (TODO document options)

`autoscale_heatmap_color`
: Set to TRUE to autoscale the heatmap colors between the maximum and minimum heatmap color parameters. If FALSE, set the heatmap colors between "Set max heatmap color" and "Set min heatmap color" (below).

`set_min_heatmap_color`
: If Autoscale heatmap color is set to FALSE, set the minimum heatmap z-score value

`set_max_heatmap_color`
: If Autoscale heatmap color is set to FALSE, set the maximum heatmap z-score value.

`aspect_ratio`
: Set figure Aspect Ratio. Ratio refers to entire figure including legend. If set to Auto figure size is based on number of rows and columns form counts matrix. default - Auto

`legend_font_size`
: Set Font size for figure legend. Default is 10.

`gene_name_font_size`
: Font size for gene names. If you don't want gene labels to show, toggle "Display Gene Names" below to FALSE

`sample_name_font_size`
: Font size for sample names. If you don't want to display samples names, toggle "Display sample names" (below) to FALSE

`display_numbers`
: Setting to FALSE (default) will not display numerical value of heat on heatmap. Set to TRUE if you want to see these numbers on the plot.

`plot_filename`
: plot output filename - only used if save_plots is TRUE

`print_plots`
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

`save_plots`
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

`plots_subdir`
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Details

By default, the top 500 genes by variance are used, as these are generally going to include those genes that most distinguish your samples from one another. You can change this as well as many other parameters about this heatmap if you explore the advanced options.

Value

heatmap from `ComplexHeatmap::Heatmap()`

See Also

Other plotters: `plot_corr_heatmap()`, `plot_histogram()`, `plot_pca()`, `plot_read_depth()`, `print_or_save_plot()`

Other heatmaps: plot_corr_heatmap()

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
# plot expression heatmap for a counts slot in a multiOmicDataset Object
moo <- multiOmicDataSet(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = nidap_raw_counts,
    "norm" = list(
      "voom" = as.data.frame(nidap_norm_counts)
    )
  )
)
p <- plot_expr_heatmap(moo, count_type = "norm", sub_count_type = "voom")

# customize the plot
plot_expr_heatmap(moo,
  count_type = "norm", sub_count_type = "voom",
  top_genes_by_variance_to_include = 100
)

# plot expression heatmap for a counts dataframe
counts_dat <- moo@counts$norm$voom
plot_expr_heatmap(
  counts_dat,
  sample_metadata = nidap_sample_metadata,
  sample_id_colname = "Sample",
  feature_id_colname = "Gene",
  group_colname = "Group",
  label_colname = "Label",
  top_genes_by_variance_to_include = 100
)
```

---

plot_histogram *Plot histogram*

Description

Plot histogram

Usage

```
plot_histogram(moo_counts, ...)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
...	arguments forwarded to method

Value

ggplot object

Methods

link to docs	class
plot_histogram_moo	multiOmicDataSet
plot_histogram_dat	data.frame

See Also

Other plotters: plot_corr_heatmap(), plot_expr_heatmap(), plot_pca(), plot_read_depth(), print_or_save_plot()

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_pca(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
# plot histogram for a counts slot in a multiOmicDataset Object
moo <- multiOmicDataSet(
  sample_metadata = nidap_sample_metadata,
  anno_dat = data.frame(),
  counts_lst = list("raw" = nidap_raw_counts)
)
p <- plot_histogram(moo, count_type = "raw")

# customize the plot
plot_histogram(moo,
  count_type = "raw",
  group_colname = "Group", color_by_group = TRUE
)

# plot histogram for a counts dataframe directly
counts_dat <- moo@counts$raw
plot_histogram(
  counts_dat,
  sample_metadata = nidap_sample_metadata,
  sample_id_colname = "Sample",
  feature_id_colname = "GeneName",
```

```
  label_colname = "Label"
)
```

`plot_histogram_dat` *Plot histogram for counts dataframe*

Description

Plot histogram for counts dataframe

Arguments

`moo_counts`	counts dataframe (**required**)
`sample_metadata`	sample metadata as a data frame or tibble (**required**)
`sample_id_colname`	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: `NULL` - first column in the sample metadata will be used.)
`feature_id_colname`	The column from the counts dataa containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)
`group_colname`	The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).
`label_colname`	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: `NULL` – `sample_id_colname` will be used.)
`color_values`	vector of colors as hex values or names recognized by R
`color_by_group`	Set to FALSE to label histogram by Sample Names, or set to TRUE to label histogram by the column you select in the "Group Column Used to Color Histogram" parameter (below). Default is FALSE.
`set_min_max_for_x_axis`	whether to override the default for `ggplot2::xlim()` (default: `FALSE`)
`minimum_for_x_axis`	value to override default `min` for `ggplot2::xlim()`

maximum_for_x_axis
: value to override default max for ggplot2::xlim()

x_axis_label
: text label for the x axis ggplot2::xlab()

y_axis_label
: text label for the y axis ggplot2::ylab()

legend_position
: passed to in legend.position ggplot2::theme()

legend_font_size
: passed to ggplot2::element_text() via ggplot2::theme()

number_of_legend_columns
: passed to ncol in ggplot2::guide_legend()

interactive_plots
: set to TRUE to make the plot interactive with plotly, allowing you to hover your mouse over a point or line to view sample information. The similarity heat map will not display if this toggle is set to TRUE. Default is FALSE.

See Also

plot_histogram generic

Other plotters for counts dataframes: plot_corr_heatmap_dat, plot_pca_dat, plot_read_depth_dat

Examples

```
# plot histogram for a counts dataframe directly
plot_histogram(
  nidap_clean_raw_counts,
  sample_metadata = nidap_sample_metadata,
  sample_id_colname = "Sample",
  feature_id_colname = "Gene",
  label_colname = "Label"
)

# customize the plot
plot_histogram(
  nidap_clean_raw_counts,
  sample_metadata = nidap_sample_metadata,
  sample_id_colname = "Sample",
  feature_id_colname = "Gene",
  group_colname = "Group",
  color_by_group = TRUE
)
```

---

plot_histogram_moo *Plot histogram for multiOmicDataSet*

Description

Plot histogram for multiOmicDataSet

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	Required if moo_counts is a multiOmicDataSet: the type of counts to use – must be a name in the counts slot (moo@counts).
sub_count_type	Used if moo_counts is a multiOmicDataSet AND if count_type is a list, specify the sub count type within the list
...	arguments forwarded to method: plot_histogram_dat

See Also

plot_histogram generic

Other plotters for multiOmicDataSets: plot_corr_heatmap_moo, plot_pca_moo, plot_read_depth_moo

Examples

```
# plot histogram for a counts slot in a multiOmicDataset Object
moo <- multiOmicDataSet(
  sample_metadata = nidap_sample_metadata,
  anno_dat = data.frame(),
  counts_lst = list("raw" = nidap_raw_counts)
)
p <- plot_histogram(moo, count_type = "raw")

# customize the plot
plot_histogram(moo,
  count_type = "raw",
  group_colname = "Group", color_by_group = TRUE
)
```

---

plot_pca *Perform and plot a Principal Components Analysis*

Description

Perform and plot a Principal Components Analysis

Usage

```
plot_pca(moo_counts, principal_components = c(1, 2), ...)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
principal_components	vector with numbered principal components to plot. Use 2 for a 2D pca with ggplot, or 3 for a 3D pca with plotly. (Default: c(1,2))
...	additional arguments forwarded to method (see Details below)

Details

See the low-level function docs for additional arguments depending on whether you're plotting 2 or 3 PCs:

- plot_pca_2d - used when there are **2** principal components
- plot_pca_3d - used when there are **3** principal components

Value

PCA plot (2D or 3D depending on the number of principal_components)

Methods

link to docs	class
plot_pca_moo	multiOmicDataSet
plot_pca_dat	data.frame

See Also

Other plotters: plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_read_depth(), print_or_save_plot()

Other PCA functions: calc_pca(), plot_pca_2d(), plot_pca_3d()

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_read_depth(), run_deseq2(), set_color_pal()

Examples

```
# multiOmicDataSet
moo <- multiOmicDataSet(
  sample_metadata = nidap_sample_metadata,
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = nidap_raw_counts,
    "clean" = nidap_clean_raw_counts
  )
```

```
)
plot_pca(moo, count_type = "clean", principal_components = c(1, 2))

# 3D
plot_pca(moo, count_type = "clean", principal_components = c(1, 2, 3))

# dataframe
plot_pca(nidap_clean_raw_counts,
  sample_metadata = nidap_sample_metadata,
  principal_components = c(1, 2)
)
```

---

plot_pca_2d *Perform and plot a 2D Principal Components Analysis*

Description

Perform and plot a 2D Principal Components Analysis

Usage

```
plot_pca_2d(
  moo_counts,
  count_type = NULL,
  sub_count_type = NULL,
  sample_metadata = NULL,
  sample_id_colname = NULL,
  feature_id_colname = NULL,
  group_colname = "Group",
  label_colname = "Label",
  samples_to_rename = NULL,
 color_values = c("#5954d6", "#e1562c", "#b80058", "#00c6f8", "#d163e6", "#00a76c",
    "#ff9287", "#008cf9", "#006e00", "#796880", "#FFA500", "#878500"),
  principal_components = c(1, 2),
  legend_position = "top",
  point_size = 1,
  add_label = TRUE,
  label_font_size = 3,
  label_offset_x_ = 2,
  label_offset_y_ = 2,
  interactive_plots = FALSE,
  plots_subdir = "pca",
  plot_filename = "pca_2D.png",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots")
)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	type to assign the values of counts_dat to in the counts slot
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).
sample_metadata	sample metadata as a data frame or tibble.
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: NULL - first column in the sample metadata will be used.)
feature_id_colname	The column from the counts dataa containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)
group_colname	The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).
label_colname	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: NULL – sample_id_colname will be used.)
samples_to_rename	If you do not have a Plot Labels Column in your sample metadata table, you can use this parameter to rename samples manually for display on the PCA plot. Use "Add item" to add each additional sample for renaming. Use the following format to describe which old name (in your sample metadata table) you want to rename to which new name: old_name: new_name
color_values	vector of colors as hex values or names recognized by R
principal_components	vector with numbered principal components to plot
legend_position	passed to in legend.position ggplot2::theme()
point_size	size for ggplot2::geom_point()
add_label	whether to add text labels for the points
label_font_size	label font size for the PCA plot

label_offset_x_
: label offset x for the PCA plot

label_offset_y_
: label offset y for the PCA plot

interactive_plots
: set to TRUE to make PCA and Histogram plots interactive with `plotly`, allowing you to hover your mouse over a point or line to view sample information. The similarity heat map will not display if this toggle is set to `TRUE`. Default is `FALSE`.

plots_subdir
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

plot_filename
: plot output filename - only used if save_plots is TRUE

print_plots
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

Value

ggplot object

See Also

plot_pca generic

Other PCA functions: `calc_pca()`, `plot_pca()`, `plot_pca_3d()`

---

plot_pca_3d *3D PCA for counts dataframe*

Description

3D PCA for counts dataframe

Usage

```
plot_pca_3d(
  moo_counts,
  count_type = NULL,
  sub_count_type = NULL,
  sample_metadata = NULL,
  feature_id_colname = NULL,
  sample_id_colname = NULL,
  samples_to_rename = NULL,
  group_colname = "Group",
  label_colname = "Label",
  principal_components = c(1, 2, 3),
  point_size = 8,
```

```
  label_font_size = 24,
 color_values = c("#5954d6", "#e1562c", "#b80058", "#00c6f8", "#d163e6", "#00a76c",
    "#ff9287", "#008cf9", "#006e00", "#796880", "#FFA500", "#878500"),
  plot_title = "PCA 3D",
  plot_filename = "pca_3D.html",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "pca"
)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	type to assign the values of counts_dat to in the counts slot
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).
sample_metadata	sample metadata as a data frame or tibble.
feature_id_colname	The column from the counts dataa containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)
sample_id_colname	The column from the sample metadata containing the sample names. The names in this column must exactly match the names used as the sample column names of your input Counts Matrix. (Default: NULL - first column in the sample metadata will be used.)
samples_to_rename	If you do not have a Plot Labels Column in your sample metadata table, you can use this parameter to rename samples manually for display on the PCA plot. Use "Add item" to add each additional sample for renaming. Use the following format to describe which old name (in your sample metadata table) you want to rename to which new name: old_name: new_name
group_colname	The column from the sample metadata containing the sample group information. This is usually a column showing to which experimental treatments each sample belongs (e.g. WildType, Knockout, Tumor, Normal, Before, After, etc.).
label_colname	The column from the sample metadata containing the sample labels as you wish them to appear in the plots produced by this template. This can be the same Sample Names Column. However, you may desire different labels to display on your figure (e.g. shorter labels are sometimes preferred on plots). In that case, select the column with your preferred Labels here. The selected column should contain unique names for each sample. (Default: NULL – sample_id_colname will be used.)

principal_components
: vector with numbered principal components to plot

point_size
: size for ggplot2::geom_point()

label_font_size
: label font size for the PCA plot

color_values
: vector of colors as hex values or names recognized by R

plot_title
: title for the plot

plot_filename
: plot output filename - only used if save_plots is TRUE

print_plots
: Whether to print plots during analysis (Defaults to FALSE, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to TRUE, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir
: subdirectory in figures/ where plots will be saved if save_plots is TRUE

Value

plotly::plot_ly figure

See Also

Other PCA functions: calc_pca(), plot_pca(), plot_pca_2d()

---

plot_pca_dat *Plot 2D or 3D PCA for counts dataframe*

Description

Plot 2D or 3D PCA for counts dataframe

Arguments

moo_counts
: counts dataframe

sample_metadata
: **Required** if moo_counts is a data.frame: sample metadata as a data frame or tibble.

principal_components
: vector with numbered principal components to plot. Use 2 for a 2D pca with ggplot, or 3 for a 3D pca with plotly. (Default: c(1,2))

...
: additional arguments forwarded to plot_pca_2d() (if 2 PCs) or plot_pca_3d() (if 3 PCs).

See Also

plot_pca generic

Other plotters for counts dataframes: plot_corr_heatmap_dat, plot_histogram_dat, plot_read_depth_dat

plot_pca_moo *Plot 2D or 3D PCA for multiOmicDataset*

Description

Plot 2D or 3D PCA for multiOmicDataset

Arguments

moo_counts	multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	the type of counts to use. Must be a name in the counts slot (names(moo@counts)).
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).
principal_components	vector with numbered principal components to plot. Use 2 for a 2D pca with ggplot, or 3 for a 3D pca with plotly. (Default: c(1,2))
...	additional arguments forwarded to plot_pca_2d() (if 2 PCs) or plot_pca_3d() (if 3 PCs).

Value

PCA plot

See Also

plot_pca generic

Other plotters for multiOmicDataSets: plot_corr_heatmap_moo, plot_histogram_moo, plot_read_depth_moo

plot_read_depth *Plot read depth as a bar plot*

Description

The first argument can be a multiOmicDataset object (moo) or a data.frame containing counts. For a moo, choose which counts slot to use with count_type & (optionally) sub_count_type.

Usage

```
plot_read_depth(moo_counts, ...)
```

Arguments

moo_counts	counts dataframe or multiOmicDataSet containing count_type & sub_count_type in the counts slot
...	arguments forwarded to method

Value

ggplot barplot

Methods

link to docs	class
plot_read_depth_moo	multiOmicDataSet
plot_read_depth_dat	data.frame

See Also

Other plotters: plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), print_or_save_plot()

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), run_deseq2(), set_color_pal()

Examples

```
# multiOmicDataSet
moo <- multiOmicDataSet(
  sample_metadata = nidap_sample_metadata,
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = nidap_raw_counts,
    "clean" = nidap_clean_raw_counts
  )
)

plot_read_depth(moo, count_type = "clean")

# dataframe
plot_read_depth(nidap_clean_raw_counts)
```

---

plot_read_depth_dat *Plot read depth for* data.frame

Description

Plot read depth for data.frame

Arguments

moo_counts	counts dataframe

Value

ggplot barplot

See Also

plot_read_depth generic

Other plotters for counts dataframes: plot_corr_heatmap_dat, plot_histogram_dat, plot_pca_dat

Examples

```
# dataframe
plot_read_depth(nidap_clean_raw_counts)
```

---

plot_read_depth_moo *Plot read depth for multiOmicDataSet*

Description

Plot read depth for multiOmicDataSet

Arguments

moo_counts	multiOmicDataSet containing count_type & sub_count_type in the counts slot
count_type	the type of counts to use. Must be a name in the counts slot (names(moo@counts)).
sub_count_type	used if count_type is a list in the counts slot: specify the sub count type within the list. Must be a name in names(moo@counts[[count_type]]).

Value

ggplot barplot

See Also

plot_read_depth generic

Other plotters for multiOmicDataSets: plot_corr_heatmap_moo, plot_histogram_moo, plot_pca_moo

Examples

```
# multiOmicDataSet
moo <- multiOmicDataSet(
  sample_metadata = nidap_sample_metadata,
  anno_dat = data.frame(),
  counts_lst = list(
    "raw" = nidap_raw_counts,
    "clean" = nidap_clean_raw_counts
```

```
  )
)

plot_read_depth(moo, count_type = "clean")
```

`plot_venn_diagram` *Plot a venn diagram, UpSet plot, or table of intersections*

Description

generates Venn diagram of intersections across a series of sets (e.g., intersections of significant genes across tested contrasts). This Venn diagram is available for up to five sets; Intersection plot is available for any number of sets. Specific sets can be selected for the visualizations and the returned dataset may include all (default) or specified intersections.

Usage

```
plot_venn_diagram(
  moo_diff_summary_dat,
  feature_id_colname = NULL,
  contrasts_colname = "Contrast",
  select_contrasts = c(),
  plot_type = "Venn diagram",
  intersection_ids = c(),
  venn_force_unique = TRUE,
  venn_numbers_format = "raw",
  venn_significant_digits = 2,
  venn_fill_colors = c("darkgoldenrod2", "darkolivegreen2", "mediumpurple3",
    "darkorange2", "lightgreen"),
  venn_fill_transparency = 0.2,
  venn_border_colors = "fill colors",
  venn_font_size_for_category_names = 3,
  venn_category_names_distance = c(),
  venn_category_names_position = c(),
  venn_font_size_for_counts = 6,
  venn_outer_margin = 0,
  intersections_order = "degree",
  display_empty_intersections = FALSE,
  intersection_bar_color = "steelblue4",
  intersection_point_size = 2.2,
  intersection_line_width = 0.7,
  table_font_size = 0.7,
  table_content = "all intersections",
  graphics_device = grDevices::png,
  dpi = 300,
  image_width = 4000,
```

```
  image_height = 3000,
  plot_filename = "venn_diagram.png",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "diff"
)
```

Arguments

`moo_diff_summary_dat`
: Summarized differential expression analysis

`feature_id_colname`
: The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)

`contrasts_colname`
: Name of the column in `moo_diff_summary_dat` that contains the contrast names (default: "Contrast")

`select_contrasts`
: A vector of contrast names to select for the plot. If empty, all contrasts are used.

`plot_type`
: Type of plot to generate: "Venn diagram" or "Intersection plot". Default: "Venn diagram"

`intersection_ids`
: A vector of intersection IDs to select for the plot. If empty, all intersections are used.

`venn_force_unique`
: If TRUE, forces unique elements in the Venn diagram. Default: TRUE

`venn_numbers_format`
: Format for the numbers in the Venn diagram. Options: "raw", "percent", "raw-percent", "percent-raw". Default: "raw"

`venn_significant_digits`
: Number of significant digits for the Venn diagram numbers. Default: 2

`venn_fill_colors`
: A vector of colors to fill the Venn diagram categories. Default: c("darkgoldenrod2", "darkolivegreen2", "mediumpurple3", "darkorange2", "lightgreen")

`venn_fill_transparency`
: Transparency level for the Venn diagram fill colors. Default: 0.2

`venn_border_colors`
: Colors for the borders of the Venn diagram categories. Default: "fill colors" (uses the same colors as `venn_fill_colors`)

`venn_font_size_for_category_names`
: Font size for the category names in the Venn diagram. Default: 3

`venn_category_names_distance`
: Distance of the category names from the Venn diagram circles. Default: c()

venn_category_names_position
: Position of the category names in the Venn diagram. Default: c()

venn_font_size_for_counts
: Font size for the counts in the Venn diagram. Default: 6

venn_outer_margin
: Outer margin for the Venn diagram. Default: 0

intersections_order
: Order of the intersections in the plot. Default: "by size"

display_empty_intersections
: If TRUE, displays empty intersections in the plot. Default: FALSE

intersection_bar_color
: Color for the intersection bars in the plot. Default: "lightgray"

intersection_point_size
: Size of the points in the intersection plot. Default: 2

intersection_line_width
: Width of the lines in the intersection plot. Default: 0.5

table_font_size
: Font size for the table in the plot. Default: 3

table_content
: Content of the table in the plot. Default: NULL

graphics_device
: passed to `ggsave(device)`. Default: `grDevices::png`

dpi
: dots-per-inch of the output image (see `ggsave()`) - only used if save_plots is TRUE

image_width
: output image width in pixels - only used if save_plots is TRUE

image_height
: output image height in pixels - only used if save_plots is TRUE

plot_filename
: plot output filename - only used if save_plots is TRUE

print_plots
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Examples

```
plot_venn_diagram(nidap_volcano_summary_dat, print_plots = TRUE)
```

plot_volcano_enhanced *Enhanced Volcano Plot*

Description

Uses Bioconductor's Enhanced Volcano Plot.

Usage

```
plot_volcano_enhanced(
  moo_diff,
  feature_id_colname = NULL,
  signif_colname = c("B-A_adjpval", "B-C_adjpval"),
  signif_threshold = 0.05,
  change_colname = c("B-A_logFC", "B-C_logFC"),
  change_threshold = 1,
  value_to_sort_the_output_dataset = "p-value",
  num_features_to_label = 30,
  use_only_addition_labels = FALSE,
  additional_labels = "",
  is_red = TRUE,
  lab_size = 4,
  change_sig_name = "p-value",
  change_lfc_name = "log2FC",
  title = "Volcano Plots",
  use_custom_lab = FALSE,
  ylim = 0,
  custom_xlim = "",
  xlim_additional = 0,
  ylim_additional = 0,
  axis_lab_size = 24,
  point_size = 2,
  image_width = 3000,
  image_height = 3000,
  dpi = 300,
  interactive_plots = FALSE,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "diff",
  plot_filename = "volcano_enhanced.png"
)
```

Arguments

moo_diff	Differential expression analysis result from one or more contrasts. This must be a dataframe.

feature_id_colname
: The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: NULL - first column in the counts matrix will be used.)

signif_colname
: column name of significance values (e.g., adjusted p-values or FDR). This column will be used to determine which points are considered significant in the volcano plot.

signif_threshold
: Numeric value specifying the significance cutoff for p-values (i.e. filters on signif_colname)

change_colname
: column name of fold change values.

change_threshold
: Numeric value specifying the fold change cutoff for significance (i.e. filters on change_colname)

value_to_sort_the_output_dataset
: How to sort the output dataset. Options are "fold-change" or "p-value".

num_features_to_label
: Number of top features/genes to label in the volcano plot. Default is 30.

use_only_addition_labels
: If TRUE, only the additional labels specified in additional_labels will be used for labeling in the volcano plot, ignoring the top features.

additional_labels
: comma-separated string of feature names or IDs to include in the volcano plot.

is_red
: Logical. If TRUE, highlights points in red.

lab_size
: Size of the labels in the volcano plot.

change_sig_name
: Name for the significance column in the plot. Default is "p-value".

change_lfc_name
: Name for the fold change column in the plot. Default is "log2FC".

title
: Title of the plot. Default is "Volcano Plots".

use_custom_lab
: If TRUE, uses custom labels for the plot (set by change_sig_name and change_lfc_name)

ylim
: Y-axis limits for the plot.

custom_xlim
: Custom X-axis limits for the plot.

xlim_additional
: Additional space to add to the X-axis limits.

ylim_additional
: Additional space to add to the Y-axis limits.

axis_lab_size
: Size of the axis labels.

point_size
: Size of the points in the plot.

image_width
: output image width in pixels - only used if save_plots is TRUE

image_height
: output image height in pixels - only used if save_plots is TRUE

dpi	dots-per-inch of the output image (see `ggsave()`) - only used if save_plots is TRUE
interactive_plots	set to TRUE to make PCA and Histogram plots interactive with `plotly`, allowing you to hover your mouse over a point or line to view sample information. The similarity heat map will not display if this toggle is set to `TRUE`. Default is `FALSE`.
print_plots	Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')
save_plots	Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')
plots_subdir	subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`
plot_filename	plot output filename - only used if save_plots is TRUE

Examples

```
plot_volcano_enhanced(nidap_deg_analysis, print_plots = TRUE)
```

`plot_volcano_summary` *Volcano Plot - Summary*

Description

Produces one volcano plot for each tested contrast in the input DEG table. It can be sorted by either fold change, t-statistic, or p-value. The returned dataset includes one row for each significant gene in each contrast, and contains columns from the DEG analysis of that contrast as well as columns useful to the Venn diagram template downstream.

Usage

```
plot_volcano_summary(
  moo_diff,
  feature_id_colname = NULL,
  signif_colname = "pval",
  signif_threshold = 0.05,
  change_threshold = 1,
  value_to_sort_the_output_dataset = "t-statistic",
  num_features_to_label = 30,
  add_features = FALSE,
  label_features = FALSE,
  custom_gene_list = "",
  default_label_color = "black",
  custom_label_color = "green3",
  label_x_adj = 0.2,
  label_y_adj = 0.2,
```

```
  line_thickness = 0.5,
  label_font_size = 4,
  label_font_type = 1,
  displace_feature_labels = FALSE,
  custom_gene_list_special_label_displacement = "",
  special_label_displacement_x_axis = 2,
  special_label_displacement_y_axis = 2,
  color_of_signif_threshold_line = "blue",
  color_of_non_significant_features = "black",
  color_of_logfold_change_threshold_line = "red",
  color_of_features_meeting_only_signif_threshold = "lightgoldenrod2",
  color_for_features_meeting_pvalue_and_foldchange_thresholds = "red",
  flip_vplot = FALSE,
  use_default_x_axis_limit = TRUE,
  x_axis_limit = 5,
  use_default_y_axis_limit = TRUE,
  y_axis_limit = 10,
  point_size = 2,
  add_deg_columns = c("FC", "logFC", "tstat", "pval", "adjpval"),
  graphics_device = grDevices::png,
  image_width = 15,
  image_height = 15,
  dpi = 300,
  use_default_grid_layout = TRUE,
  number_of_rows_in_grid_layout = 1,
  aspect_ratio = 0,
  plot_filename = "volcano_summary.png",
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_subdir = "diff"
)
```

Arguments

`moo_diff`	Differential expression analysis result from one or more contrasts. This must be a dataframe.
`feature_id_colname`	The column from the counts data containing the Feature IDs (Usually Gene or Protein ID). This is usually the first column of your input Counts Matrix. Only columns of Text type from your input Counts Matrix will be available to select for this parameter. (Default: `NULL` - first column in the counts matrix will be used.)
`signif_colname`	column name of significance values (e.g., adjusted p-values or FDR). This column will be used to determine which points are considered significant in the volcano plot.
`signif_threshold`	Numeric value specifying the significance cutoff for p-values (i.e. filters on `signif_colname`)

change_threshold
: Numeric value specifying the fold change cutoff for significance (i.e. filters on change_colname)

value_to_sort_the_output_dataset
: How to sort the output dataset. Options are "fold-change" or "p-value".

num_features_to_label
: Number of top features/genes to label in the volcano plot. Default is 30.

add_features
: Add custom_gene_list To Labels. Set TRUE when you want to label a specific set of features (features) in the "custom_gene_list" parameter" IN ADDITION to the number of features you set in the "Number of Features to Label" parameter.

label_features
: Select TRUE when you want to label ONLY a specific list of features(features) given in the "custom_gene_list" parameter.

custom_gene_list
: Provide a list of features (comma separated) to be labeled on the volcano plot. You must toggle one of the following ON to see these labels: "Add features" or "Label Only My Feature List".

default_label_color
: Set the color for the text used to add feature (gene) name labels to points.

custom_label_color
: Set the color for the specific list of features (features) provided in the "Feature List" parameter.

label_x_adj
: adjust position of the labels on the x-axis. Default: 0.2

label_y_adj
: adjust position of the labels on the y-axis. Default: 0.2

line_thickness
: Set the thickness of the lines in the plot. Default: 0.5

label_font_size
: Set the font size of the labels. Default: 4

label_font_type
: Set the font type of the labels. Default: 1

displace_feature_labels
: Set to TRUE to displace gene labels. Default: FALSE. Set TRUE if you want to displace the feature (gene) label for a specific set of features. Make sure to use custom x- and y- limits and give sufficient space for displacement; otherwise other labels than the desired ones will appear displaced.

custom_gene_list_special_label_displacement
: Provide a list of features (comma separated) for which you want special displacement of the feature label.

special_label_displacement_x_axis
: Displacement of the feature label on the x-axis. Default: 2

special_label_displacement_y_axis
: Displacement of the feature label on the y-axis. Default: 2

color_of_signif_threshold_line
: Color of the significance threshold line. Default: "blue"

color_of_non_significant_features
: Color of the non-significant features. Default: "black"

color_of_logfold_change_threshold_line
: Color of the log fold change threshold line. Default: "red"

color_of_features_meeting_only_signif_threshold
: Color of the features that meet only the significance threshold. Default: "lightgoldenrod2"

color_for_features_meeting_pvalue_and_foldchange_thresholds
: Color of the features that meet both the p-value and fold change thresholds. Default: "red"

flip_vplot
: Set to TRUE to flip the fold change values so that the volcano plot looks like a comparison was B-A. Default: FALSE

use_default_x_axis_limit
: Set to TRUE to use the default x-axis limit. Default: TRUE

x_axis_limit
: Custom x-axis limit. Default: c(-5, 5)

use_default_y_axis_limit
: Set to TRUE to use the default y-axis limit. Default: TRUE

y_axis_limit
: Custom y-axis limit. Default: c(0, 10)

point_size
: Size of the points in the plot. Default: 1

add_deg_columns
: Add additional columns from the DEG analysis to the output dataset. Default: `"FC", "logFC", "tstat", "pval", "adjpval"`

graphics_device
: passed to `ggsave(device)`. Default: `grDevices::png`

image_width
: output image width in pixels - only used if save_plots is TRUE

image_height
: output image height in pixels - only used if save_plots is TRUE

dpi
: dots-per-inch of the output image (see `ggsave()`) - only used if save_plots is TRUE

use_default_grid_layout
: Set to TRUE to use the default grid layout. Default: TRUE

number_of_rows_in_grid_layout
: Number of rows in the grid layout. Default: 1

aspect_ratio
: Aspect ratio of the output image. Default: 4/3

plot_filename
: Filename for the output plot. Default: "volcano_plot.png"

print_plots
: Whether to print plots during analysis (Defaults to `FALSE`, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')

save_plots
: Whether to save plots to files during analysis (Defaults to `TRUE`, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')

plots_subdir
: subdirectory in `figures/` where plots will be saved if `save_plots` is `TRUE`

Examples

```
plot_volcano_summary(nidap_deg_analysis, print_plots = TRUE)
```

print_or_save_plot *Print and/or save a ggplot*

Description

If save_plots is TRUE, the plot will be saved as an image to the path at file.path(plots_dir, filename). If plot_obj is a ggplot, ggplot2::ggsave() is used to save the image. Otherwise, graphics_device is used (grDevice::png() by default).

Usage

```
print_or_save_plot(
  plot_obj,
  filename,
  print_plots = options::opt("print_plots"),
  save_plots = options::opt("save_plots"),
  plots_dir = options::opt("plots_dir"),
  graphics_device = grDevices::png,
  ...
)
```

Arguments

plot_obj	plot object (e.g. ggplot, ComplexHeatmap...)
filename	name of the output file. will be joined with the plots_dir option.
print_plots	Whether to print plots during analysis (Defaults to FALSE, overwritable using option 'moo_print_plots' or environment variable 'MOO_PRINT_PLOTS')
save_plots	Whether to save plots to files during analysis (Defaults to TRUE, overwritable using option 'moo_save_plots' or environment variable 'MOO_SAVE_PLOTS')
plots_dir	Path where plots are saved when moo_save_plots is TRUE (Defaults to "figures/", overwritable using option 'moo_plots_dir' or environment variable 'MOO_PLOTS_DIR')
graphics_device	Default: grDevice::png(). Only used if the plot is not a ggplot.
...	arguments forwarded to ggplot2::ggsave()

Value

invisibly returns the path where the plot image was saved to the disk

See Also

Other plotters: plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth()

read_multiOmicDataSet *Read a multiOmicDataSet from disk*

Description

Read a multiOmicDataSet from disk

Usage

```
read_multiOmicDataSet(filepath)
```

Arguments

filepath	Path to an RDS file produced by write_multiOmicDataSet()

Value

multiOmicDataSet

set_color_pal *Set color palette for a single group/column*

Description

This allows you to set custom palettes individually for groups in the dataset

Usage

```
set_color_pal(moo, colname, palette_fun = grDevices::palette.colors, ...)
```

Arguments

moo	multiOmicDataSet object (see create_multiOmicDataSet_from_dataframes())
colname	group column name to set the palette for
palette_fun	Function for selecting colors. Assumed to contain n for the number of colors. Default: grDevices::palette.colors()
...	additional arguments forwarded to palette_fun

Value

moo with colors updated at moo@analyses$colors$colname

See Also

Other moo methods: batch_correct_counts(), clean_raw_counts(), diff_counts(), filter_counts(), filter_diff(), normalize_counts(), plot_corr_heatmap(), plot_expr_heatmap(), plot_histogram(), plot_pca(), plot_read_depth(), run_deseq2()

Examples

```
moo <- create_multiOmicDataSet_from_dataframes(
  sample_metadata = as.data.frame(nidap_sample_metadata),
  counts_dat = as.data.frame(nidap_raw_counts)
)
moo@analyses$colors$Group
moo <- moo |> set_color_pal("Group", palette_fun = RColorBrewer::brewer.pal, name = "Set2")
moo@analyses$colors$Group
```

---

write_multiOmicDataSet

Write a multiOmicDataSet to disk as an RDS file

Description

Write a multiOmicDataSet to disk as an RDS file

Usage

```
write_multiOmicDataSet(moo, filepath = "moo.rds")
```

Arguments

moo	multiOmicDataSet object to serialize
filepath	Path to the RDS file to write (default: "moo.rds")

Value

Invisibly returns filepath

`write_multiOmicDataSet_properties` *Write multiOmicDataSet properties to disk as CSV files*

Description

Writes the properties of a multiOmicDataSet object to disk as separate files in output_dir. Properties that are data frames are saved as CSV files, while all other objects are saved as RDS files.

Usage

```
write_multiOmicDataSet_properties(moo, output_dir = "moo")
```

Arguments

`moo`	`multiOmicDataSet` object to write properties from
`output_dir`	Directory where the properties will be saved (default: "moo")

Value

Invisibly returns the `output_dir` where the files were saved

Index