Package 'OlinkAnalyze'

June 24, 2025

Type Package

Title Facilitate Analysis of Proteomic Data from Olink

Version 4.3.0

Description A collection of functions to facilitate analysis of proteomic data from Olink, primarily NPX data that has been exported from Olink Software. The functions also work on QUANT data from Olink by log- transforming the QUANT data. The functions are focused on reading data, facilitating data wrangling and quality control analysis, performing statistical analysis and generating figures to visualize the results of the statistical analysis. The goal of this package is to help users extract biological insights from proteomic data run on the Olink platform.

License AGPL (>= 3)

Contact biostattools@olink.com

Depends R (>= 4.1.0)

Imports broom, car, cli (>= 3.6.2), dplyr (>= 1.1.1), data.table, emmeans, forcats, generics, ggplot2, ggpubr, ggrepel, grDevices, grid, magrittr, methods, readxl, rlang, rstatix, stats, stringr, tibble, tidyr, tidyselect, tools, utils

Suggests arrow, clusterProfiler, extrafont, FSA, ggplotify, kableExtra, knitr, lme4, lmerTest, markdown, msigdbr (>= 9.0.0), openssl, ordinal, pheatmap, rmarkdown, scales, systemfonts, testthat (>= 3.0.0), umap, vdiffr, withr, zip

VignetteBuilder kableExtra, knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://olink.com/
https://github.com/Olink-Proteomics/OlinkRPackage

NeedsCompilation no

Author Kathleen Nevola [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5183-6444>, kathy-nevola),
Marianne Sandin [aut] (ORCID: <https://orcid.org/0000-0001-6186-963X>,
marisand),
Jamey Guess [aut] (ORCID: <https://orcid.org/0000-0002-4017-0923>,
jrguess),
Simon Forsberg [aut] (ORCID: <https://orcid.org/0000-0002-7451-9222>,
simfor),
Christoffer Cambronero [aut] (Orbmac),
Pascal Pucholt [aut] (ORCID: <https://orcid.org/0000-0003-3342-1373>,
AskPascal),
Boxi Zhang [aut] (ORCID: <https://orcid.org/0000-0001-7758-6204>,
boxizhang),
Masoumeh Sheikhi [aut] (MasoumehSheikhi),
Klev Diamanti [aut] (ORCID: <https://orcid.org/0000-0002-4922-8415>,
klevdiamanti),
Amrita Kar [aut] (amrita-kar),
Lei Conze [aut] (leiliuC),
Kristyn Chin [aut] (kristynchin-olink),
Danai Topouza [aut] (dtopouza, ORCID:
<https://orcid.org/0000-0002-6897-9281>),
Stephen Pollo [aut] (spollo-olprot, ORCID:
<https://orcid.org/0000-0001-9252-4976>),
Kang Dong [aut] (KangD-dev, ORCID:
<https://orcid.org/0000-0002-4567-5007>),
Kristian Hodén [ctb] (ORCID: <https://orcid.org/0000-0003-0354-0662>,
kristianHoden),
Per Eriksson [ctb] (ORCID: <https://orcid.org/0000-0001-7633-403X>,
b_watcher),
Nicola Moloney [ctb] (ORCID: <https://orcid.org/0000-0003-4967-3284>),
Britta Lötstedt [ctb] (ORCID: <https://orcid.org/0000-0003-3545-5489>),
Emmett Sprecher [ctb] (ORCID: <https://orcid.org/0000-0002-7710-695X>),
Jessica Barbagallo [ctb] (jbarbagallo),
Olof Mansson [ctr] (olofmansson),
Ola Caster [ctb] (OlaCaster),
Olink [cph, fnd]

Maintainer Kathleen Nevola <biostattools@olink.com>

Repository CRAN

Date/Publication 2025-06-24 16:10:02 UTC

Contents

`check_data_completeness`

Check data completeness

Description

Throw informative warnings if a dataset appears to have problems

Usage

```
check_data_completeness(df)
```

Arguments

`df`	a NPX dataframe, e.g. from read_NPX()

Value

None. Used for side effects (warnings)

Examples

```
npx_data1 %>%
    dplyr::mutate(NPX = dplyr::if_else(
                          SampleID == "A1" & Panel == "Olink Cardiometabolic",
                          NA_real_,
                          NPX)) %>%
    OlinkAnalyze:::check_data_completeness()
```

`manifest` *Example Sample Manifest*

Description

Sample manifest is generated randomly to demonstrate use of functions in this package.

Usage

```
manifest
```

Format

This dataset contains columns:

SubjectID Subject Identifier, A-Z

Visit Visit Number, 1-6

SampleID 138 unique sample IDs

Site Site1 or Site2

Details

A tibble with 138 rows and 4 columns. This manifest contains 26 example subjects, with 6 visits and 2 sites.

`mapping_file_id` *Identifying which mapping file to use*

Description

Identifying which mapping file to use

Usage

```
mapping_file_id(ref_product)
```

Arguments

ref_product	one of "HT" or "Reveal" depending on reference product

Value

dataframe of mapping file to use for OlinkID mapping (eHT_e3072_mapping or reveal_e3072_mapping)

---

norm_internal_adjust *Combine reference and non-reference datasets*

Description

The function is used by norm_internal_subset and norm_internal_bridge to combine the reference dataset that has Adj_factor = 0 and the non-reference dataset that used the adjustment factors provided in adj_fct_df.

Usage

```
norm_internal_adjust(
  ref_df,
  ref_name,
  ref_cols,
  not_ref_df,
  not_ref_name,
  not_ref_cols,
  adj_fct_df
)
```

Arguments

ref_df	The reference dataset to be used in normalization (required).
ref_name	Project name of the reference dataset (required).
ref_cols	Named list of column names in the reference dataset (required).
not_ref_df	The non-reference dataset to be used in normalization (required).
not_ref_name	Project name of the non-reference dataset (required).
not_ref_cols	Named list of column names in the non-reference dataset (required).
adj_fct_df	Dataset containing the adjustment factors to be applied to the non-reference dataset for (required).

Details

The function calls norm_internal_adjust_ref and norm_internal_adjust_not_ref and combines their outputs.

Value

Tibble or ArrowObject with the normalized dataset.

Author(s)

Klev Diamanti

`norm_internal_adjust_not_ref`
Add adjustment factors to a dataset

Description

Add adjustment factors to a dataset

Usage

```
norm_internal_adjust_not_ref(df, name, cols, adj_fct_df, adj_fct_cols)
```

Arguments

`df`	The dataset to be normalized (required).
`name`	Project name of the dataset (required).
`cols`	Named list of column names in the dataset (required).
`adj_fct_df`	Dataset containing the adjustment factors to be applied to the dataset `not_ref_df` (required).
`adj_fct_cols`	Named list of column names in the dataset containing adjustment factors (required).

Value

Tibble or ArrowObject with the normalized dataset with additional columns "Project" and "Adj_factor".

Author(s)

Klev Diamanti

`norm_internal_adjust_ref` *Modify the reference dataset to be combined with the non-reference normalized dataset*

Description

Modify the reference dataset to be combined with the non-reference normalized dataset

Usage

```
norm_internal_adjust_ref(ref_df, ref_name)
```

Arguments

`ref_df`	The reference dataset to be used in normalization (required).
`ref_name`	Project name of the reference dataset (required).

Value

Tibble or ArrowObject with the reference dataset with additional columns "Project" and "Adj_factor".

Author(s)

Klev Diamanti

`norm_internal_assay_median` *Compute median value of the quantification method for each Olink assay*

Description

The function computes the median value of the the quantification method for each Olink assay in the set of samples `samples`, and it adds the column `Project`.

Usage

```
norm_internal_assay_median(df, samples, name, cols)
```

Arguments

`df`	The dataset to calculate medians from (required).
`samples`	Character vector of sample identifiers to be used for adjustment factor calculation in the dataset `df` (required).
`name`	Project name of the dataset that will be added in the column `Project` (required).
`cols`	Named list of column names identified in the dataset `df` (required).

Details

This function is typically used by internal functions `norm_internal_subset` and `norm_internal_reference_median` that compute median quantification value for each assay across multiple samples specified by `samples`.

Value

Tibble or ArrowObject with one row per Olink assay and the columns OlinkID, Project, and assay_med

Author(s)

Klev Diamanti

`norm_internal_bridge` *Internal bridge normalization function*

Description

Internal bridge normalization function

Usage

```
norm_internal_bridge(
  ref_df,
  ref_samples,
  ref_name,
  ref_cols,
  not_ref_df,
  not_ref_name,
  not_ref_cols
)
```

Arguments

`ref_df`	The reference dataset to be used in normalization (required).
`ref_samples`	Character vector of sample identifiers to be used for adjustment factor calculation in the reference dataset (required).
`ref_name`	Project name of the reference dataset (required).
`ref_cols`	Named list of column names in the reference dataset (required).
`not_ref_df`	The non-reference dataset to be used in normalization (required).
`not_ref_name`	Project name of the non-reference dataset (required).
`not_ref_cols`	Named list of column names in the non-reference dataset (required).

Value

Tibble or ArrowObject with the normalized dataset.

Author(s)

Klev Diamanti

`norm_internal_cross_product` *Internal function normalizing Olink Explore 3k to Olink Explore 3072*

Description

Internal function normalizing Olink Explore 3k to Olink Explore 3072

Usage

```
norm_internal_cross_product(
  ref_df,
  ref_samples,
  ref_name,
  ref_cols,
  ref_product,
  not_ref_df,
  not_ref_name,
  not_ref_cols
)
```

Arguments

`ref_df`	The reference dataset to be used in normalization (required).
`ref_samples`	Character vector of sample identifiers to be used for adjustment factor calculation in the reference dataset (required).
`ref_name`	Project name of the reference dataset (required).
`ref_cols`	Named list of column names in the reference dataset (required).
`ref_product`	Name of reference product (required).
`not_ref_df`	The non-reference dataset to be used in normalization (required).
`not_ref_name`	Project name of the non-reference dataset (required).
`not_ref_cols`	Named list of column names in the non-reference dataset (required).

Value

Tibble or ArrowObject with a dataset with the following additional columns:

- OlinkID_E3072: Corresponding assay identifier from Olink Explore 3072.
- Project: Project of origin.
- BridgingRecommendation: Recommendation of whether the assay is bridgeable or not. One of "NotBridgeable", "MedianCentering", or "QuantileSmoothing".
- MedianCenteredNPX: NPX values adjusted based on the median of the pair-wise differences of NPX values between bridge samples.
- QSNormalizedNPX: NPX values adjusted based on the quantile smoothing normalization among bridge samples.

Author(s)

Klev Diamanti

---

norm_internal_reference_median

Internal reference median normalization function

Description

Internal reference median normalization function

Usage

```
norm_internal_reference_median(
  ref_df,
  ref_samples,
  ref_name,
  ref_cols,
  reference_medians
)
```

Arguments

ref_df	The reference dataset to be used in normalization (required).
ref_samples	Character vector of sample identifiers to be used for adjustment factor calculation in the reference dataset (required).
ref_name	Project name of the reference dataset (required).
ref_cols	Named list of column names in the reference dataset (required).
reference_medians	Dataset with columns "OlinkID" and "Reference_NPX" (required). Used for reference median normalization.

Value

Tibble or ArrowObject with the normalized dataset.

Author(s)

Klev Diamanti

`norm_internal_rename_cols` *Update column names of non-reference dataset based on those of reference dataset*

Description

This function handles cases when specific columns referring to the same thing are named differently in `df1` and `df2` normalization datasets. It only renames columns panel_version, qc_warn, and assay_warn based on their names in the reference dataset.#'

Usage

```
norm_internal_rename_cols(ref_cols, not_ref_cols, not_ref_df)
```

Arguments

`ref_cols`	Named list of column names identified in the reference dataset.
`not_ref_cols`	Named list of column names identified in the non-reference dataset.
`not_ref_df`	Non-reference dataset to be used in normalization.

Value

`not_ref_df` with updated column names.

Author(s)

Klev Diamanti

`norm_internal_subset` *Internal subset normalization function*

Description

This function performs subset normalization using a subset of the samples from either or both reference and non-reference datasets. When all samples from each dataset are used, the function performs intensity normalization.

Usage

```
norm_internal_subset(
  ref_df,
  ref_samples,
  ref_name,
  ref_cols,
  not_ref_df,
  not_ref_samples,
  not_ref_name,
  not_ref_cols
)
```

Arguments

`ref_df`	The reference dataset to be used in normalization (required).
`ref_samples`	Character vector of sample identifiers to be used for adjustment factor calculation in the reference dataset (required).
`ref_name`	Project name of the reference dataset (required).
`ref_cols`	Named list of column names in the reference dataset (required).
`not_ref_df`	The non-reference dataset to be used in normalization (required).
`not_ref_samples`	Character vector of sample identifiers to be used for adjustment factor calculation in the non-reference dataset (required).
`not_ref_name`	Project name of the non-reference dataset (required).
`not_ref_cols`	Named list of column names in the non-reference dataset (required).

Value

Tibble or ArrowObject with the normalized dataset.

Author(s)

Klev Diamanti

`norm_internal_update_maxlod` *Update MaxLOD to the maximum MaxLOD across normalized datasets.*

Description

Update MaxLOD to the maximum MaxLOD across normalized datasets.

Usage

```
norm_internal_update_maxlod(df, cols)
```

Arguments

`df`	Normalized Olink dataset (required).
`cols`	Named list of column names in the dataset (required).

Value

The same dataset as the input *df* with the column reflecting MaxLOD updated.

`npx_data1` *NPX Data in Long format*

Description

Data is generated randomly to demonstrate use of functions in this package.

Usage

```
npx_data1
```

Format

In addition to standard read_NPX() columns, this dataset also contains columns:

Subject Subject Identifier

Treatment Treated or Untreated

Site Site indicator, 5 unique values

Time Baseline, Week.6 and Week.12

Project Project ID number

Details

A tibble with 29,440 rows and 17 columns. Dataset npx_data1 is an Olink NPX data file (tibble) in long format with 158 unique Sample ID's (including 2 repeats each of control samples: CONTROL_SAMPLE_AS 1 CONTROL_SAMPLE_AS 2). The data also contains 1104 assays (uniquely identified using OlinkID) over 2 Panels.

---

npx_data2 *NPX Data in Long format, Follow-up*

Description

Data is generated randomly to demonstrate use of functions in this package. The format is very similar to data(npx_data1). Both datasets can be used together to demonstrate the use of normalization functionality.

Usage

```
npx_data2
```

Format

In addition to standard read_NPX() columns, this dataset also contains columns:

Subject Subject Identifier

Treatment Treated or Untreated

Site Site indicator, 5 unique values

Time Baseline, Week.6 and Week.12

Project Project ID number

Details

A tibble with 32,384 rows and 17 columns. npx_data2 is an Olink NPX data file (tibble) in long format with 174 unique Sample ID's (including 2 repeats each of control samples: CONTROL_SAMPLE_AS 1 CONTROL_SAMPLE_AS 2). The data also contains 1104 assays (uniquely identified using OlinkID) over 2 Panels. This dataset also contain 16 bridge samples with SampleID's that are also present in data(npx_data1). These sample ID's are: A13, A29, A30, A36, A45, A46, A52, A63, A71, A73, B3, B4, B37, B45, B63, B75

`olink_anova` *Function which performs an ANOVA per protein*

Description

Performs an ANOVA F-test for each assay (by OlinkID) in every panel using car::Anova and Type III sum of squares. The function handles both factor and numerical variables and/or covariates.

Samples that have no variable information or missing factor levels are automatically removed from the analysis (specified in a message if verbose = TRUE). Character columns in the input dataframe are automatically converted to factors (specified in a message if verbose = TRUE). Numerical variables are not converted to factors. Control samples should be removed before using this function. Control assays (AssayType is not "assay", or Assay contains "control" or "ctrl") should be removed before using this function. If a numerical variable is to be used as a factor, this conversion needs to be done on the dataframe before the function call.

Crossed analysis, i.e. A*B formula notation, is inferred from the variable argument in the following cases:

- c('A','B')
- c('A: B')
- c('A: B', 'B') or c('A: B', 'A')

Inference is specified in a message if verbose = TRUE.
For covariates, crossed analyses need to be specified explicitly, i.e. two main effects will not be expanded with a c('A','B') notation. Main effects present in the variable takes precedence. The formula notation of the final model is specified in a message if verbose = TRUE.

Adjusted p-values are calculated by stats::p.adjust according to the Benjamini & Hochberg (1995) method ("fdr"). The threshold is determined by logic evaluation of Adjusted_pval < 0.05. Covariates are not included in the p-value adjustment.

Usage

```
olink_anova(
  df,
  variable,
  outcome = "NPX",
  covariates = NULL,
  model_formula,
  return.covariates = FALSE,
  verbose = TRUE
)
```

Arguments

df	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' or '*' notation.
outcome	Character. The dependent variable. Default: NPX.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':' or '*' notation. Crossed analysis will not be inferred from main effects.
model_formula	(optional) Symbolic description of the model to be fitted in standard formula notation (e.g. "NPX~A*B"). If provided, this will override the `outcome`, `variable` and `covariates` arguments. Can be a string or of class `stats::formula()`.
return.covariates	Boolean. Default: False. Returns F-test results for the covariates. Note: Adjusted p-values will be NA for the covariates.
verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A "tibble" containing the ANOVA results for every protein. The tibble is arranged by ascending p-values. Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- df: "numeric" degrees of freedom
- sumsq: "numeric" sum of square
- meansq: "numeric" mean of square
- statistic: "numeric" value of the statistic
- p.value: "numeric" nominal p-value
- Adjusted_pval: "numeric" adjusted p-value for the test (Benjamini&Hochberg)
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

npx_df <- npx_data1 |> filter(!grepl('control|ctrl',SampleID, ignore.case = TRUE))

#One-way ANOVA, no covariates.
```

```
#Results in a model NPX~Time
anova_results <- olink_anova(df = npx_df, variable = "Time")

#Two-way ANOVA, one main effect covariate.
#Results in model NPX~Treatment*Time+Site.
anova_results <- olink_anova(df = npx_df,
                             variable=c("Treatment:Time"),
                             covariates="Site")

#One-way ANOVA, interaction effect covariate.
#Results in model NPX~Treatment+Site:Time+Site+Time.
anova_results <- olink_anova(df = npx_df,
                             variable="Treatment",
                             covariates="Site:Time")
```

`olink_anova_posthoc` *Function which performs an ANOVA posthoc test per protein.*

Description

Performs a post hoc ANOVA test using emmeans::emmeans with Tukey p-value adjustment per assay (by OlinkID) for each panel at confidence level 0.95. See `olink_anova` for details of input notation.

The function handles both factor and numerical variables and/or covariates. Control samples should be removed before using this function. Control assays (AssayType is not "assay", or Assay contains "control" or "ctrl") should be removed before using this function. The posthoc test for a numerical variable compares the difference in means of the outcome variable (default: NPX) for 1 standard deviation difference in the numerical variable, e.g. mean NPX at mean(numerical variable) versus mean NPX at mean(numerical variable) + 1*SD(numerical variable).

Usage

```
olink_anova_posthoc(
  df,
  olinkid_list = NULL,
  variable,
  covariates = NULL,
  outcome = "NPX",
  model_formula,
  effect,
  effect_formula,
  mean_return = FALSE,
  post_hoc_padjust_method = "tukey",
  verbose = TRUE
)
```

Arguments

df	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
olinkid_list	Character vector of OlinkID's on which to perform post hoc analysis. If not specified, all assays in df are used.
variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' notation.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':' or '*' notation. Crossed analysis will not be inferred from main effects.
outcome	Character. The dependent variable. Default: NPX.
model_formula	(optional) Symbolic description of the model to be fitted in standard formula notation (e.g. "NPX~A*B"). If provided, this will override the `outcome`, `variable` and `covariates` arguments. Can be a string or of class `stats::formula()`.
effect	Term on which to perform post-hoc. Character vector. Must be subset of or identical to variable.
effect_formula	(optional) A character vector specifying the names of the predictors over which estimated marginal means are desired as defined in the `emmeans` package. May also be a formula. If provided, this will override the `effect` argument. See `?emmeans::emmeans()` for more information.
mean_return	Boolean. If true, returns the mean of each factor level rather than the difference in means (default). Note that no p-value is returned for mean_return = TRUE and no adjustment is performed.
post_hoc_padjust_method	P-value adjustment method to use for post-hoc comparisons within an assay. Options include `tukey`, `sidak`, `bonferroni` and `none`.
verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A "tibble" of posthoc tests for specified effect, arranged by ascending adjusted p-values. Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- contrast: "character" the groups that were compared
- estimate: "numeric" difference in mean NPX between groups
- conf.low: "numeric" confidence interval for the mean (lower end)
- conf.high: "numeric" confidence interval for the mean (upper end)
- Adjusted_pval: "numeric" adjusted p-value for the test
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

npx_df <- npx_data1 |> filter(!grepl('control|ctrl',SampleID, ignore.case = TRUE))

#Two-way ANOVA, one main effect (Site) covariate.
#Results in model NPX~Treatment*Time+Site.
anova_results <- olink_anova(df = npx_df,
                             variable=c("Treatment:Time"),
                             covariates="Site")

#Posthoc test for the model NPX~Treatment*Time+Site,
#on the interaction effect Treatment:Time with covariate Site.

#Filtering out significant and relevant results.
significant_assays <- anova_results |>
filter(Threshold == 'Significant' & term == 'Treatment:Time') |>
select(OlinkID) |>
distinct() |>
pull()

#Posthoc, all pairwise comparisons
anova_posthoc_results <- olink_anova_posthoc(npx_df,
variable=c("Treatment:Time"),
covariates="Site",
olinkid_list = significant_assays,
effect = "Treatment:Time")


#Posthoc, treated vs untreated at each timepoint, adjusted for Site effect
anova_posthoc_results <- olink_anova_posthoc(npx_df,
model_formula = "NPX~Treatment*Time+Site",
olinkid_list = significant_assays,
effect_formula = "pairwise~Treatment|Time")
```

---

olink_boxplot *Function which plots boxplots of selected variables*

Description

Generates faceted boxplots of NPX vs. grouping variable(s) for a given list of proteins (OlinkIDs) using ggplot and ggplot2::geom_boxplot.

Usage

```
olink_boxplot(
  df,
  variable,
  olinkid_list,
  verbose = FALSE,
  number_of_proteins_per_plot = 6,
  posthoc_results = NULL,
  ttest_results = NULL,
  ...
)
```

Arguments

df	NPX data frame in long format with at least protein name (Assay), OlinkID (unique), UniProt and at least one grouping variable.
variable	A character vector or character value indicating which column to use as the x-axis and fill grouping variable. The first or single value is used as x-axis, the second as fill. Further values in a vector are not plotted.
olinkid_list	Character vector indicating which proteins (OlinkIDs) to plot.
verbose	Boolean. If the plots are shown as well as returned in the list (default is false).
number_of_proteins_per_plot	Number of boxplots to include in the facet plot (default 6).
posthoc_results	Data frame from ANOVA posthoc analysis using olink_anova_posthoc() function.
ttest_results	Data frame from ttest analysis using olink_ttest() function.
...	coloroption passed to specify color order

Value

A list of objects of class "ggplot" (the actual ggplot object is entry 1 in the list). Box and whisker plot of NPX (y-axis) by variable (x-axis) for each Assay

Examples

```
library(dplyr)
npx_df <- npx_data1 |> filter(!grepl('control|ctrl',SampleID, ignore.case = TRUE))
anova_results <- olink_anova(npx_df, variable = "Site")
significant_assays <- anova_results |>
    filter(Threshold == 'Significant') |>
    pull(OlinkID)
olink_boxplot(npx_df,
              variable = "Site",
              olinkid_list = significant_assays,
              verbose = TRUE,
              number_of_proteins_per_plot = 3)
```

`olink_bridgeability_plot` *Plots for each bridgeable assays between two products.*

Description

Plots for each bridgeable assays between two products.

Usage

```
olink_bridgeability_plot(
  data,
  olink_id = NULL,
  median_counts_threshold = 150L,
  min_count = 10L
)
```

Arguments

`data`	A tibble containing the cross-product bridge normalized dataset generated by `olink_normalization`.
`olink_id`	Character vector of Olink assay identifiers *OlinkID* for which bridgeability plots will be created. If null, plots for all assays in *data* will be created. (default = NULL)
`median_counts_threshold`	Threshold indicating the minimum median counts for each product (default = 150).
`min_count`	Threshold indicating the minimum number of counts per data point (default = 10). Data below *min_count* are excluded.

Value

An object of class "ggplot" containing 4 plots for each assay.

Author(s)

Amrita Kar Klev Diamanti

Generates a combined plot per assay containing a violin and boxplot for IQR ranges; correlation plot of NPX values; a median count bar plot and KS plots from the 2 products.

Examples

```
npx_ht <- OlinkAnalyze:::data_ht_small |>
  dplyr::filter(
    .data[["SampleType"]] == "SAMPLE"
  )

npx_3072 <- OlinkAnalyze:::data_3k_small |>
  dplyr::filter(
    .data[["SampleType"]] == "SAMPLE"
  )

overlapping_samples <- intersect(
  x = npx_ht$SampleID,
  y = npx_3072$SampleID
)

data_norm <- OlinkAnalyze::olink_normalization(
  df1 = npx_ht,
  df2 = npx_3072,
  overlapping_samples_df1 = overlapping_samples,
  df1_project_nr = "Explore HT",
  df2_project_nr = "Explore 3072",
  reference_project = "Explore HT"
)

data_norm_bridge_p <- OlinkAnalyze::olink_bridgeability_plot(
  data = data_norm,
  olink_id = c("OID40770", "OID40835"),
  median_counts_threshold = 150L,
  min_count = 10L
)
```

---

olink_bridgeselector *Bridge selection function*

Description

The bridge selection function will select a number of bridge samples based on the input data. It selects samples with good detection, which passes QC and cover a good range of the data. If possible, Olink recommends 8-16 bridge samples. When running the selector, Olink recommends starting at sampleMissingFreq = 0.10 which represents a maximum of 10\ data below LOD per sample. If there are not enough samples output, increase to 20\ The function accepts NPX Excel files with data < LOD replaced.

Usage

```
olink_bridgeselector(df, sampleMissingFreq, n)
```

Arguments

df	Tibble/data frame in long format such as produced by the Olink Analyze read_NPX function.
sampleMissingFreq	The threshold for sample wise missingness.
n	Number of bridge samples to be selected.

Value

A "tibble" with sample IDs and mean NPX for a defined number of bridging samples. Columns include:

- SampleID: Sample ID
- PercAssaysBelowLOD: Percent of Assays that are below LOD for the sample
- MeanNPX: Mean NPX for the sample

Examples

```
bridge_samples <- olink_bridgeselector(npx_data1, sampleMissingFreq = 0.1, n = 20)
```

---

olink_color_discrete *Olink color scale for discrete ggplots*

Description

Olink color scale for discrete ggplots

Usage

```
olink_color_discrete(..., alpha = 1, coloroption = NULL)
```

Arguments

...	Optional. Additional arguments to pass to ggplot2::discrete_scale()
alpha	transparency
coloroption	string, one or more of the following: c('red', 'orange', 'yellow', 'green', 'teal', 'turqoise', 'lightblue', 'darkblue', 'purple', 'pink')

Value

No return value, called for side effects

Examples

```
library(ggplot2)

ggplot(mtcars, aes(x=wt, y=mpg, color=as.factor(cyl))) +
geom_point(size = 4) +
olink_color_discrete() +
theme_bw()

ggplot(mtcars, aes(x=wt, y=mpg, color=as.factor(cyl))) +
geom_point(size = 4) +
olink_color_discrete(coloroption = c('lightblue', 'red', 'green')) +
theme_bw()
```

---

olink_color_gradient *Olink color scale for continuous ggplots*

Description

Olink color scale for continuous ggplots

Usage

```
olink_color_gradient(..., alpha = 1, coloroption = NULL)
```

Arguments

...	Optional. Additional arguments to pass to scale_color_gradientn()
alpha	transparency (optional)
coloroption	string, one or more of the following: c('red', 'orange', 'yellow', 'green', 'teal', 'turqoise', 'lightblue', 'darkblue', 'purple', 'pink')

Value

No return value, called for side effects

Examples

```
library(ggplot2)

dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
dsub$diff <- with(dsub, sqrt(abs(x-y))* sign(x-y))

ggplot(dsub, aes(x, y, colour=diff)) +
geom_point() +
 theme_bw() +
 olink_color_gradient()
```

olink_displayPlateDistributions

Plot distributions of a given variable for all plates

Description

Displays a bar chart for each plate representing the distribution of the given grouping variable on each plate using ggplot2::ggplot and ggplot2::geom_bar.

Usage

```
olink_displayPlateDistributions(data, fill.color)
```

Arguments

`data`	tibble/data frame in long format returned from the olink_plate_randomizer function.
`fill.color`	Column name to be used as coloring variable for wells.

Value

An object of class "ggplot" showing the percent distribution of fill.color in each plate (x-axis)

See Also

- `olink_plate_randomizer()` for generating a plating scheme
- `olink_displayPlateLayout()` for visualizing the generated plate layouts

Examples

```
randomized.manifest <- olink_plate_randomizer(manifest)
olink_displayPlateDistributions(data=randomized.manifest,fill.color="Site")
```

olink_displayPlateLayout

Plot all plates colored by a variable

Description

Displays each plate in a facet with cells colored by the given variable using ggplot and ggplot2::geom_tile.

Usage

```
olink_displayPlateLayout(
  data,
  fill.color,
  PlateSize = 96,
  num_ctrl = 8,
  rand_ctrl = FALSE,
  Product,
  include.label = FALSE
)
```

Arguments

`data`	tibble/data frame in long format returned from the olink_plate_randomizer function.
`fill.color`	Column name to be used as coloring variable for wells.
`PlateSize`	Integer. Either 96 or 48. 96 is default.
`num_ctrl`	Numeric. Number of controls on each plate (default = 8)
`rand_ctrl`	Logical. Whether controls are added to be randomized across the plate (default = FALSE)
`Product`	String. Name of Olink product used to set PlateSize if not provided. Optional.
`include.label`	Should the variable group be shown in the plot.

Value

An object of class "ggplot" showing each plate in a facet with the cells colored by values in column fill.color in input `data`.

See Also

- `olink_plate_randomizer()` for generating a plating scheme
- `olink_displayPlateDistributions()` for validating that sites are properly randomized

Examples

```
randomized.manifest <- olink_plate_randomizer(manifest)
olink_displayPlateLayout(data = randomized.manifest, fill.color="Site")
```

olink_dist_plot *Function to plot the NPX distribution by panel*

Description

Generates boxplots of NPX vs. SampleID colored by QC_Warning (default) or any other grouping variable and faceted by Panel using ggplot and ggplot2::geom_boxplot.

Usage

```
olink_dist_plot(df, color_g = "QC_Warning", ...)
```

Arguments

df	NPX data frame in long format. Must have columns SampleID, NPX and Panel
color_g	Character value indicating which column to use as fill color (default: QC_Warning)
...	Color option passed to specify color order.

Value

An object of class "ggplot" which displays NPX distribution for each sample per panel

Examples

```
olink_dist_plot(npx_data1, color_g = "QC_Warning")
```

olink_fill_discrete *Olink fill scale for discrete ggplots*

Description

Olink fill scale for discrete ggplots

Usage

```
olink_fill_discrete(..., alpha = 1, coloroption = NULL)
```

Arguments

...	Optional. Additional arguments to pass to ggplot2::discrete_scale()
alpha	transparency (optional)
coloroption	string, one or more of the following: c('red', 'orange', 'yellow', 'green', 'teal', 'turqoise', 'lightblue', 'darkblue', 'purple', 'pink')

Value

No return value, called for side effects

Examples

```
library(ggplot2)

dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
dsub$diff <- with(dsub, sqrt(abs(x-y))* sign(x-y))

ggplot(dsub, aes(x, y, colour=diff)) +
geom_point() +
 theme_bw() +
 olink_fill_discrete()
```

---

olink_fill_gradient *Olink fill scale for continuous ggplots*

Description

Olink fill scale for continuous ggplots

Usage

```
olink_fill_gradient(..., alpha = 1, coloroption = NULL)
```

Arguments

...	Optional. Additional arguments to pass to ggplot2::scale_fill_gradientn()
alpha	transparency (optional)
coloroption	string, one or more of the following: c('red', 'orange', 'yellow', 'green', 'teal', 'turqoise', 'lightblue', 'darkblue', 'purple', 'pink')

Value

No return value, called for side effects

Examples

```
library(ggplot2)

dsub <- subset(diamonds, x > 5 & x < 6 & y > 5 & y < 6)
dsub$diff <- with(dsub, sqrt(abs(x-y))* sign(x-y))
ggplot(dsub, aes(x, y, colour=diff)) +
geom_point() +
 theme_bw() +
 olink_fill_gradient()
```

`olink_heatmap_plot` *Function to plot a heatmap of the NPX data*

Description

Generates a heatmap using `pheatmap::pheatmap` of all samples from NPX data.

Usage

```
olink_heatmap_plot(
  df,
  variable_row_list = NULL,
  variable_col_list = NULL,
  center_scale = TRUE,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = TRUE,
  show_colnames = TRUE,
  colnames = "both",
  annotation_legend = TRUE,
  fontsize = 10,
  na_col = "black",
  ...
)
```

Arguments

`df`	Data frame in long format with SampleID, NPX, OlinkID, Assay and columns of choice for annotations.
`variable_row_list`	Columns in `df` to be annotated for rows in the heatmap.
`variable_col_list`	Columns in `df` to be annotated for columns in the heatmap.
`center_scale`	Logical. If data should be centered and scaled across assays (default `TRUE`).
`cluster_rows`	Logical. Determining if rows should be clustered (default `TRUE`).
`cluster_cols`	Logical. Determining if columns should be clustered (default `TRUE`).
`show_rownames`	Logical. Determining if row names are shown (default `TRUE`).
`show_colnames`	Logical. Determining if column names are shown (default `TRUE`).
`colnames`	Character. Determines how to label the columns. Must be 'assay', 'oid', or 'both' (default 'both').
`annotation_legend`	Logical. Determining if legend for annotations should be shown (default `TRUE`).
`fontsize`	Fontsize (default 10)
`na_col`	Color of cells with `NA` (default black)
`...`	Additional arguments used in `pheatmap::pheatmap`

Details

The values are by default scaled across and centered in the heatmap. Columns and rows are by default sorted by by dendrogram. Unique sample names are required.

Value

An object of class `ggplot`, generated from the `gtable` returned by `pheatmap::pheatmap`.

Examples

```
library(dplyr)
npx_data <- npx_data1 %>%
      filter(!stringr::str_detect(SampleID,'CONT'))
try({ # This will fail if ggplotify is not installed
#Heatmap
  olink_heatmap_plot(df=npx_data)

#Heatmap with annotation
  olink_heatmap_plot(df=npx_data, variable_row_list = c('Time','Site'))

#Heatmap with calls from pheatmap
  olink_heatmap_plot(df=npx_data, cutree_rows = 3)
})
```

---

olink_iqr *Compute inter-quartile range (IQR) of multiplied by a fixed value*

Description

Compute inter-quartile range (IQR) of multiplied by a fixed value

Usage

```
olink_iqr(df, quant_col, iqr_group, iqr_sd)
```

Arguments

`df`	Olink dataset
`quant_col`	Character vector of name of quantification column
`iqr_group`	Grouping for which to compute IQR for
`iqr_sd`	Fixed value to multiply IQR with

Value

Input dataset with two additional columns, iqr and iqr_sd

`olink_lmer` *Function which performs a linear mixed model per protein*

Description

Fits a linear mixed effects model for every protein (by OlinkID) in every panel, using lmerTest::lmer and stats::anova. The function handles both factor and numerical variables and/or covariates.

Samples that have no variable information or missing factor levels are automatically removed from the analysis (specified in a message if verbose = TRUE). Character columns in the input dataframe are automatically converted to factors (specified in a message if verbose = TRUE). Numerical variables are not converted to factors. If a numerical variable is to be used as a factor, this conversion needs to be done on the dataframe before the function call.

Crossed analysis, i.e. A*B formula notation, is inferred from the variable argument in the following cases:

- c('A','B')
- c('A:B')
- c('A:B', 'B') or c('A:B', 'A')

Inference is specified in a message if verbose = TRUE.
For covariates, crossed analyses need to be specified explicitly, i.e. two main effects will not be expanded with a c('A','B') notation. Main effects present in the variable takes precedence.
The random variable only takes main effect(s).
The formula notation of the final model is specified in a message if verbose = TRUE.

Output p-values are adjusted by stats::p.adjust according to the Benjamini-Hochberg method ("fdr"). Adjusted p-values are logically evaluated towards adjusted p-value<0.05.

Usage

```
olink_lmer(
  df,
  variable,
  outcome = "NPX",
  random,
  covariates = NULL,
  model_formula,
  return.covariates = FALSE,
  verbose = TRUE
)
```

Arguments

`df`	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, 1-2 variables with at least 2 levels.

variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' or '*' notation.
outcome	Character. The dependent variable. Default: NPX.
random	Single character value or character array.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':' or '*' notation. Crossed analysis will not be inferred from main effects.
model_formula	(optional) Symbolic description of the model to be fitted in standard formula notation (e.g. "NPX~A*B + (1|ID)"). If provided, this will override the `outcome`, `variable` and `covariates` arguments. Can be a string or of class `stats::formula()`.
return.covariates	Boolean. Default: False. Returns results for the covariates. Note: Adjusted p-values will be NA for the covariates.
verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A "tibble" containing the results of fitting the linear mixed effects model to every protein by OlinkID, ordered by ascending p-value. Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- sumsq: "numeric" sum of square
- meansq: "numeric" mean of square
- NumDF: "integer" numerator of degrees of freedom
- DenDF: "numeric" denominator of decrees of freedom
- statistic: "numeric" value of the statistic
- p.value: "numeric" nominal p-value
- Adjusted_pval: "numeric" adjusted p-value for the test (Benjamini&Hochberg)
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
if (requireNamespace("lme4", quietly = TRUE) & requireNamespace("lmerTest", quietly = TRUE)){
  # Results in model NPX~Time*Treatment+(1|Subject)+(1|Site)
  lmer_results <- olink_lmer(df = npx_data1,
  variable=c("Time", 'Treatment'),
  random = c('Subject', 'Site'))
}
```

`olink_lmer_plot` *Function which performs a point-range plot per protein on a linear mixed model*

Description

Generates a point-range plot faceted by Assay using ggplot and ggplot2::geom_pointrange based on a linear mixed effects model using lmerTest:lmer and emmeans::emmeans. See `olink_lmer` for details of input notation.

Usage

```
olink_lmer_plot(
  df,
  variable,
  outcome = "NPX",
  random,
  olinkid_list = NULL,
  covariates = NULL,
  x_axis_variable,
  col_variable = NULL,
  number_of_proteins_per_plot = 6,
  verbose = FALSE,
  ...
)
```

Arguments

`df`	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, 1-2 variables with at least 2 levels.
`variable`	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' or '*' notation.
`outcome`	Character. The dependent variable. Default: NPX.
`random`	Single character value or character array.
`olinkid_list`	Character vector indicating which proteins (by OlinkID) for which to create figures.
`covariates`	Single character value or character array. Default: NULL. Covariates to include. Takes ':' or '*' notation. Crossed analysis will not be inferred from main effects.
`x_axis_variable`	Character. Which main effect to use as x-axis in the plot.
`col_variable`	Character. If provided, the interaction effect col_variable:x_axis_variable will be plotted with x_axis_variable on the x-axis and col_variable as color.
`number_of_proteins_per_plot`	Number plots to include in the list of point-range plots. Defaults to 6 plots per figure

verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.
...	coloroption for color ordering

Value

A list of objects of class "ggplot" showing point-range plot of NPX (y-axis) over x_axis_variable for each assay (facet), colored by col_variable if provided.

Examples

```
library(dplyr)
if (requireNamespace("lme4", quietly = TRUE) & requireNamespace("lmerTest", quietly = TRUE)){
lmer_results <- olink_lmer(df = npx_data1,
                           variable=c("Time", 'Treatment'),
                           random = c('Subject'))

assay_list <- lmer_results %>%
    filter(Threshold == 'Significant' & term == 'Time:Treatment') %>%
    select(OlinkID) %>%
    distinct() %>%
    pull()

list_of_pointrange_plots <- olink_lmer_plot(df = npx_data1,
                                            variable=c("Time", 'Treatment'),
                                            random = c('Subject'),
                                            x_axis_variable = 'Time',
                                            col_variable = 'Treatment',
                                            verbose=TRUE,
                                            olinkid_list = assay_list,
                                            number_of_proteins_per_plot = 10)
}
```

---

olink_lmer_posthoc *Function which performs a linear mixed model posthoc per protein.*

Description

Similar to olink_lmer but performs a post hoc analysis based on a linear mixed model effects model using lmerTest::lmer and emmeans::emmeans on proteins. See `olink_lmer` for details of input notation.

The function handles both factor and numerical variables and/or covariates. Differences in estimated marginal means are calculated for all pairwise levels of a given variable. Degrees of freedom are estimated using Satterthwaite's approximation. The posthoc test for a numerical variable compares the difference in means of the outcome variable (default: NPX) for 1 standard deviation difference in the numerical variable, e.g. mean NPX at mean(numerical variable) versus mean NPX at

mean(numerical variable) + 1*SD(numerical variable). The output tibble is arranged by ascending Tukey adjusted p-values.

Usage

```
olink_lmer_posthoc(
  df,
  olinkid_list = NULL,
  variable,
  outcome = "NPX",
  random,
  model_formula,
  effect,
  effect_formula,
  covariates = NULL,
  mean_return = FALSE,
  post_hoc_padjust_method = "tukey",
  verbose = TRUE
)
```

Arguments

df	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, 1-2 variables with at least 2 levels and subject ID.
olinkid_list	Character vector of OlinkID's on which to perform post hoc analysis. If not specified, all assays in df are used.
variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' or '*' notation.
outcome	Character. The dependent variable. Default: NPX.
random	Single character value or character array.
model_formula	(optional) Symbolic description of the model to be fitted in standard formula notation (e.g. "NPX~A*B + (1|ID)"). If provided, this will override the `outcome`, `variable` and `covariates` arguments. Can be a string or of class `stats::formula()`.
effect	Term on which to perform post-hoc. Character vector. Must be subset of or identical to variable.
effect_formula	(optional) A character vector specifying the names of the predictors over which estimated marginal means are desired as defined in the `emmeans` package. May also be a formula. If provided, this will override the `effect` argument. See `?emmeans::emmeans()` for more information.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':' or '*' notation. Crossed analysis will not be inferred from main effects.
mean_return	Boolean. If true, returns the mean of each factor level rather than the difference in means (default). Note that no p-value is returned for mean_return = TRUE and no adjustment is performed.

post_hoc_padjust_method
: P-value adjustment method to use for post-hoc comparisons within an assay. Options include `tukey`, `sidak`, `bonferroni` and `none`.

verbose
: Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A "tibble" containing the results of the pairwise comparisons between given variable levels for proteins specified in olinkid_list (or full df). Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- contrast: "character" the groups that were compared
- estimate: "numeric" difference in mean NPX between groups
- conf.low: "numeric" confidence interval for the mean (lower end)
- conf.high: "numeric" confidence interval for the mean (upper end)
- Adjusted_pval: "numeric" adjusted p-value for the test
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)
if (requireNamespace("lme4", quietly = TRUE) & requireNamespace("lmerTest", quietly = TRUE)){

lmer_results <- olink_lmer(df = npx_data1,
                           variable=c("Time", 'Treatment'),
                           random = c('Subject'))

assay_list <- lmer_results %>%
    filter(Threshold == 'Significant' & term == 'Time:Treatment') %>%
    select(OlinkID) %>%
    distinct() %>%
    pull()

results_lmer_posthoc <- olink_lmer_posthoc(df = npx_data1,
                                           olinkid_list = assay_list,
                                           variable=c("Time", 'Treatment'),
                                           effect = 'Time:Treatment',
                                           random = 'Subject',
                                           verbose = TRUE)

#Estimate treated vs untreated at each timepoint
```

```
results_lmer_posthoc <- olink_lmer_posthoc(df = npx_data1,
                                           olinkid_list = assay_list,
                                          model_formula = "NPX~Time*Treatment+(1|Subject)",
                                           effect_formula = "pairwise~Treatment|Time",
                                           verbose = TRUE)
}
```

`olink_lod` *Calculate LOD using Negative Controls or Fixed LOD*

Description

Calculate LOD using Negative Controls or Fixed LOD

Usage

```
olink_lod(data, lod_file_path = NULL, lod_method = "NCLOD")
```

Arguments

`data`	npx data file
`lod_file_path`	location of lod file from Olink. Only needed if lod_method = "FixedLOD" or "Both". Default `NULL`.
`lod_method`	method for calculating LOD using either "FixedLOD" or negative controls ("NCLOD"), or both ("Both"). Default `NCLOD`.

Value

A dataframe with 2 additional columns, LOD and PCNormalizedLOD if `lod_method` is FixedLOD or NCLOD. When `Normalization = "Plate Control"`, LOD and PCNormalizedLOD are identical.

If `lod_method` is "Both", 4 additional columns will be added:

- NCLOD - LOD calculated from negative controls and normalized based on normalization column
- NCPCNormalizedLOD - PC Normalized LOD calculated from negative controls
- FixedLOD - LOD calculated from fixed LOD file and normalized based on normalization column
- FixedPCNormalizedLOD - PC Normalized LOD calculated from fixed LOD file

Examples

```
## Not run:
   \donttest{
  try({ # This will fail if the files do not exist.

  # Import NPX data
    npx_data <- read_NPX("path/to/npx_file")

  # Estimate LOD from negative controls
    npx_data_lod_NC <- olink_lod(data = npx_data,  lod_method = "NCLOD")

  # Estimate LOD from fixed LOD
  ## Locate the fixed LOD file
    lod_file_path <- "path/to/lod_file"

    npx_data_lod_Fixed <- olink_lod(data = npx_data,
                                    lod_file_path = lod_file_path,
                                    lod_method = "FixedLOD")
     })
  }

## End(Not run)
```

`olink_median` *Compute median of quantified value*

Description

Compute median of quantified value

Usage

```
olink_median(df, quant_col, median_group)
```

Arguments

`df`	Olink dataset
`quant_col`	Character vector of name of quantification column
`median_group`	Grouping for which to compute median for

Value

Input dataset with one additional columns, median

olink_median_iqr_outlier *Compute outliers based on median +/- iqr_sd * IQR*

Description

Compute outliers based on median +/- iqr_sd * IQR

Usage

```
olink_median_iqr_outlier(df, quant_col, group, iqr_sd)
```

Arguments

df	Olink dataset
quant_col	Character vector of name of quantification column
group	Grouping for which to compute median for
iqr_sd	Fixed value to multiply IQR with

Value

Boolean vector with length equal to the number of input rows indicating outlier.

olink_normalization *Normalize two Olink datasets*

Description

Normalizes two Olink datasets to each other, or one Olink dataset to a reference set of medians values.

Usage

```
olink_normalization(
  df1,
  df2 = NULL,
  overlapping_samples_df1,
  overlapping_samples_df2 = NULL,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P1",
  reference_medians = NULL,
  format = FALSE
)
```

Arguments

df1	First dataset to be used for normalization (required).
df2	Second dataset to be used for normalization. Required for bridge and subset normalization.
overlapping_samples_df1	Character vector of samples to be used for the calculation of adjustment factors in df1 (required).
overlapping_samples_df2	Character vector of samples to be used for the calculation of adjustment factors in df2. Required for subset normalization.
df1_project_nr	Project name of first dataset (required).
df2_project_nr	Project name of second dataset. Required for bridge and subset normalization.
reference_project	Project to be used as reference project. Should be one of df1_project_nr and df2_project_nr. Required for bridge and subset normalization.
reference_medians	Dataset with columns "OlinkID" and "Reference_NPX". Required for reference median normalization.
format	Boolean that controls whether the normalized dataset will be formatted for input to downstream analysis. Only applicable for cross-product bridge normalization.

Details

The function handles four different types of normalization:

- **Bridge normalization**: One of the datasets is adjusted to another using overlapping samples (bridge samples). Overlapping samples need to have the same identifiers in both datasets. Normalization is performed using the median of the pair-wise differences between the bridge samples in the two datasets. The two datasets are provided as df1 and df2, and the one being adjusted to is specified in the input reference_project; overlapping samples are specified in overlapping_samples_df1. Only overlapping_samples_df1 should be provided regardless of the dataset used as reference_project.
- **Subset normalization**: One of the datasets is adjusted to another using a subset of samples from each. Normalization is performed using the differences of the medians between the subsets from the two datasets. Both overlapping_samples_df1 and overlapping_samples_df2 need to be provided, and sample identifiers do not need to be the same.
  - A special case of subset normalization occurs when all samples (except control samples and samples with QC warnings) from each dataset are used for normalization; this special case is called intensity normalization. In intensity normalization all unique sample identifiers from df1 are provided as input in overlapping_samples_df1 and all unique sample identifiers from df2 are provided as input in overlapping_samples_df2.
- **Reference median normalization**: One of the datasets (df1) is adjusted to a predefined set of adjustment factors. This is effectively subset normalization, but using differences of medians to pre-recorded median values. df1, overlapping_samples_df1, df1_project_nr and reference_medians need to be specified. Dataset df1 is normalized using the differences in median between the overlapping samples and the reference medians.

- **Cross-product normalization**: One of the datasets is adjusted to another using the median of pair-wise differences of overlapping samples (bridge samples) or quantile smoothing using overlapping samples as reference to adjust the distributions. Overlapping samples need to have the same identifiers in both datasets. The two datasets are provided as `df1` and `df2`, and the one being adjusted to is specified in the input `reference_project`; **Note that** in cross-product normalization the reference project is predefined, and in case the argument `reference_project` does not match the expected reference project an error will be returned. Overlapping samples are specified in `overlapping_samples_df1`. Only `overlapping_samples_df1` should be provided regardless of the dataset used as `reference_project`. This functionality **does not** modify the column with original quantification values (e.g. NPX), instead it normalizes it with 2 different approaches in columns "MedianCenteredNPX" and "QSNormalizedNPX", and provides a recommendation in "BridgingRecommendation" about which of the two columns is to be used.

The output dataset is `df1` if reference median normalization, or `df2` appended to `df1` if bridge, subset or cross-product normalization. The output dataset contains all original columns from the original dataset(s), and the columns:

- "Project" and "Adj_factor" in case of reference median, bridge and subset normalization. The former marks the project of origin based on `df1_project_nr` and `df2_project_nr`, and the latter the adjustment factor that was applied to the non-reference dataset.
- "Project", "OlinkID_E3072", "MedianCenteredNPX", "QSNormalizedNPX", "BridgingRecommendation" in case of cross-product normalization. The columns correspond to the project of origin based on `df1_project_nr` and `df2_project_nr`, the assay identifier in the non-reference project, the bridge-normalized quantification value, the quantile smoothing-normalized quantification value, and the recommendation about which of the two normalized values is more suitable for downstream analysis.

Value

Tibble or ArrowObject with the normalized dataset.

Examples

```
# prepare datasets
npx_df1 <- npx_data1 |>
  dplyr::mutate(
    Normalization = "Intensity"
  )
npx_df2 <- npx_data2 |>
  dplyr::mutate(
    Normalization = "Intensity"
  )

# bridge normalization

# overlapping samples - exclude control samples
overlap_samples <- intersect(x = npx_df1$SampleID,
                             y = npx_df2$SampleID) |>
  (\(x) x[!grepl("^CONTROL_SAMPLE", x)])()
```

```
# normalize
olink_normalization(
  df1 = npx_df1,
  df2 = npx_df2,
  overlapping_samples_df1 = overlap_samples,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P1"
)

# subset normalization

# find a suitable subset of samples from each dataset:
# exclude control samples
# exclude samples that do not pass QC
df1_samples <- npx_df1 |>
  dplyr::group_by(
    dplyr::pick(
      dplyr::all_of("SampleID")
    )
  )|>
  dplyr::filter(
    all(.data[["QC_Warning"]] == 'Pass')
  ) |>
  dplyr::ungroup() |>
  dplyr::filter(
    !grepl(pattern = "^CONTROL_SAMPLE", x = .data[["SampleID"]])
  ) |>
  dplyr::pull(
    .data[["SampleID"]]
  ) |>
  unique()
df2_samples <- npx_df2 |>
  dplyr::group_by(
    dplyr::pick(
      dplyr::all_of("SampleID")
    )
  )|>
  dplyr::filter(
    all(.data[["QC_Warning"]] == 'Pass')
  ) |>
  dplyr::ungroup() |>
  dplyr::filter(
    !grepl(pattern = "^CONTROL_SAMPLE", x = .data[["SampleID"]])
  ) |>
  dplyr::pull(
    .data[["SampleID"]]
  ) |>
  unique()

# select a subset of samples from each set from above
df1_subset <- sample(x = df1_samples, size = 16L)
```

```
df2_subset <- sample(x = df2_samples, size = 20L)

# normalize
olink_normalization(
  df1 = npx_df1,
  df2 = npx_df2,
  overlapping_samples_df1 = df1_subset,
  overlapping_samples_df2 = df2_subset,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P1"
)

# special case of subset normalization using all samples
olink_normalization(
  df1 = npx_df1,
  df2 = npx_df2,
  overlapping_samples_df1 = df1_samples,
  overlapping_samples_df2 = df2_samples,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P1"
)

# reference median normalization

# For the sake of this example, set the reference median to 1
ref_med_df <- npx_data1 |>
  dplyr::select(
    dplyr::all_of(
      c("OlinkID")
    )
  ) |>
  dplyr::distinct() |>
  dplyr::mutate(
    Reference_NPX = runif(n = dplyr::n(),
                          min = -1,
                          max = 1)
  )

# normalize
olink_normalization(
  df1 = npx_df1,
  overlapping_samples_df1 = df1_subset,
  reference_medians = ref_med_df
)

# cross-product normalization

# get reference samples
overlap_samples_product <- intersect(
  x = unique(OlinkAnalyze:::data_ht_small$SampleID),
  y = unique(OlinkAnalyze:::data_3k_small$SampleID)
```

```
) |>
  (\(.) .[!grepl("CONTROL", .)])()

# normalize
olink_normalization(
  df1 = OlinkAnalyze:::data_ht_small,
  df2 = OlinkAnalyze:::data_3k_small,
  overlapping_samples_df1 = overlap_samples_product,
  df1_project_nr = "proj_ht",
  df2_project_nr = "proj_3k",
  reference_project = "proj_ht",
  format = FALSE
)
```

`olink_normalization_bridge` *Bridge normalization of all proteins between two NPX projects.*

Description

Normalizes two NPX projects (data frames) using shared samples.

Usage

```
olink_normalization_bridge(
  project_1_df,
  project_2_df,
  bridge_samples,
  project_1_name = "P1",
  project_2_name = "P2",
  project_ref_name = "P1"
)
```

Arguments

`project_1_df`	Data frame of the first project (required).
`project_2_df`	Data frame of the second project (required).
`bridge_samples`	Named list of 2 arrays containing SampleID of shared samples to be used for the calculation of adjustment factor. The names of the two arrays should be DF1 and DF2 corresponding to projects 1 and 2, respectively. Arrays should be of equal length and index of each entry should correspond to the same sample. (required)
`project_1_name`	Name of the first project (default: P1).

`project_2_name` Name of the second project (default: P2).

`project_ref_name`
: Name of the project to be used as reference set. Needs to be one of the project_1_name or project_2_name. It marks the project to which the other project will be adjusted to (default: P1).

Details

This function is a wrapper of olink_normalization.

In bridging normalization one of the projects is adjusted to another using shared samples (bridge samples). It is not necessary for the shared samples to be named the same in each project. Adjustment between the two projects is made using the median of the paired differences between the shared samples. The two data frames are inputs project_1_df and project_2_df, the one being adjusted to is specified in the input project_ref_name and the shared samples are specified in bridge_samples.

Value

A "tibble" of NPX data in long format containing normalized NPX values, including adjustment factors and name of project.

Examples

```
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find overlapping samples, but exclude Olink control
overlap_samples <- dplyr::intersect(unique(npx_df1$SampleID),
                                    unique(npx_df2$SampleID))
overlap_samples_list <- list("DF1" = overlap_samples,
                             "DF2" = overlap_samples)

# Normalize
olink_normalization_bridge(project_1_df = npx_df1,
                           project_2_df = npx_df2,
                           bridge_samples = overlap_samples_list,
                           project_1_name = "P1",
                           project_2_name = "P2",
                           project_ref_name = "P1")
```

`olink_normalization_bridgeable` *Identify if assays shared between Olink Explore 3072 and Olink Explore HT can be bridged*

Description

The function uses a dataset from Olink Explore 3072 and a dataset from Olink Explore HT, and examines if the matched assays between the two products can be normalized to each other. The input datasets should be exported from Olink software and should not be altered prior to importing them to this function.

Usage

```
olink_normalization_bridgeable(lst_df, ref_cols, not_ref_cols, seed = 1)
```

Arguments

`lst_df`	A named list of the 2 input datasets. First element should be the reference dataset from Olink Explore HT and the second element should originate from Olink Explore 3072.
`ref_cols`	A named list with the column names to use. Exported from olink_norm_input_check.
`not_ref_cols`	A named list with the column names from the non-reference dataset. Exported from olink_norm_input_check.
`seed`	Integer random seed (Default: seek = 1).

Details

All processes below assume that the first element from *lst_df* is the reference dataset (e.g. Olink Explore HT), and the other element of the list is the non-reference dataset (e.g. Olink Explore 3072). The input datasets **have to be pre-processed** by `olink_norm_input_check` which will take care of mapping of assay identifiers and various checks. Also, the input datasets should exclusively contain datapoints from bridge samples. When this function is called from the function `olink_normalization`, then the list is created seamlessly in the background, and the datasets have been already processed by `olink_norm_input_check`.

The input *ref_cols* is a named list masking column names of the reference dataset. This list is generated automatically from `olink_norm_input_check` when it is called from `olink_normalization`. In addition, `olink_normalization` has also utilized `norm_internal_rename_cols` to rename the columns of the non-reference dataset according to the ones of the reference dataset, hence all column names should match.

Value

A "tibble" in long format with the following columns:

- OlinkID: Underscore-separated Olink identifiers of matching assays between Olink Explore HT and Olink Explore 3072.

- BridgingRecommendation: A character vector indicating whether the matching assays are considered as bridgeable or not, and the recommended type of normalization to perform.

Author(s)

Amrita Kar Marianne Sandin Danai G. Topouza Klev Diamanti

Examples

```
# check input datasets
data_explore_check <- OlinkAnalyze:::olink_norm_input_check(
  df1 = OlinkAnalyze:::data_3k_small,
  df2 = OlinkAnalyze:::data_ht_small,
  overlapping_samples_df1 = intersect(
    x = unique(OlinkAnalyze:::data_3k_small$SampleID),
    y = unique(OlinkAnalyze:::data_ht_small$SampleID)
  ) |>
    (\(x) x[!grepl("CONTROL", x)])() |>
    head(20L),
  overlapping_samples_df2 = NULL,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P2",
  reference_medians = NULL
)

# create lst_df
lst_df <- list(
  data_explore_check$ref_df,
  data_explore_check$not_ref_df
)
names(lst_df) <- c(data_explore_check$ref_name,
                   data_explore_check$not_ref_name)

# create ref_cols
ref_cols <- data_explore_check$ref_cols
not_ref_cols <- data_explore_check$not_ref_cols

# run olink_normalization_bridgeable
is_bridgeable_result <- OlinkAnalyze:::olink_normalization_bridgeable(
  lst_df = lst_df,
  ref_cols = ref_cols,
  not_ref_cols = not_ref_cols,
  seed = 1
)
```

olink_normalization_n *Bridge and/or subset normalization of all proteins among multiple NPX projects.*

Description

This function normalizes pairs of NPX projects (data frames) using shared samples or subsets of samples.

Usage

```
olink_normalization_n(norm_schema)
```

Arguments

norm_schema	A tibble with more than 1 rows and (strictly) the following columns: "order", "name", "data", "samples", "normalization_type", "normalize_to". See "Details" for the structure of the data frame (required)

Details

This function is a wrapper of olink_normalization_bridge and olink_normalization_subset.

The input of this function is a tibble that contains all the necessary information to normalize multiple NPX projects. This tibble is called the normalization schema. The basic idea is that every row of the data frame is a separate project to be normalized. We assume that there is always one baseline project that does not normalize to any other. All other project normalize to one or more projects. The function handles projects that are normalized in a chain, for example:

- 1. project 2 normalizes to project 1, and project 3 normalizes to project 2.
- 2. project 2 normalizes to project 1, and project 3 normalizes to the combined data frame of projects 1 and 2 (that is already normalized).

The function can also handle a mixed schema of bridge and subset normalization.

Specifications of the normalization schema data frame:

- order: should strictly be a numeric or integer array with unique identifiers for each project. It is necessary that this array starts from 1 and that it contains no NAs.
- name: should strictly be a character array with unique identifiers for each project. Each entry should represent the name of the project located in the same row. No NAs are allowed.
- data: a named list of NPX data frames representing the projects to be normalized. Names of the items of the list should be identical to "names". No NAs are allowed.

- samples: a two-level nested named list of sample identifiers from each NPX project from "data". Names of the first level of the nested list should be identical to "names" and to the names of the list from "data". Projects that will be used only as reference should have their corresponding element in the list as NA, while all other projects should contain a named list of 2 arrays containing identifiers of samples to be used for the calculation of adjustment factor. The names of the two arrays should be DF1 and DF2 corresponding to the reference project and the project in the current row, respectively. For bridge normalization arrays should be of equal length and the index of each entry should correspond to the same sample. For subset normalization arrays do not need to be of equal length and the order the samples appear in does not matter. DF1 might contain sample identifiers from more than one project as long as the project in the current row is to be normalized to multiple other projects.
- normalization_type: a character array containing the flags "Bridge" or "Subset". Projects that will be used only as reference should have their corresponding element in the array as NA, while all other projects should contain a flag. For the time being the flag "Median" is not supported.
- normalize_to: a character array pointing to the project this project is to be normalized to. Elements of the array should be exclusively from the "order" column. Elements of the array may be comma-separated if the project is to be normalized to multiple projects.

Value

A "tibble" of NPX data in long format containing normalized NPX values, including adjustment factors and name of project.

Examples

```
#### Bridge normalization of two projects

# prepare datasets
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find overlapping samples, but exclude Olink control
overlap_samples <- dplyr::intersect(unique(npx_df1$SampleID),
                                    unique(npx_df2$SampleID))
overlap_samples_list <- list("DF1" = overlap_samples,
                             "DF2" = overlap_samples)

# create tibble for input
norm_schema_bridge <- dplyr::tibble(
  order              = c(1, 2),
  name               = c("NPX_DF1", "NPX_DF2"),
  data               = list("NPX_DF1" = npx_df1,
                            "NPX_DF2" = npx_df2),
```

```
  samples            = list("NPX_DF1" = NA_character_,
                            "NPX_DF2" = overlap_samples_list),
  normalization_type = c(NA_character_, "Bridge"),
  normalize_to       = c(NA_character_, "1")
)

# normalize
olink_normalization_n(norm_schema = norm_schema_bridge)

#### Subset normalization of two projects

# datasets
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find a suitable subset of samples from both projects, but exclude Olink
# controls and samples that fail QC.
df1_samples <- npx_df1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique() |>
  sample(size = 16, replace = FALSE)
df2_samples <- npx_df2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique() |>
  sample(size = 16, replace = FALSE)

# create named list
subset_samples_list <- list("DF1" = df1_samples,
                            "DF2" = df2_samples)

# create tibble for input
norm_schema_subset <- dplyr::tibble(
  order              = c(1, 2),
  name               = c("NPX_DF1", "NPX_DF2"),
  data               = list("NPX_DF1" = npx_df1,
                            "NPX_DF2" = npx_df2),
  samples            = list("NPX_DF1" = NA_character_,
                            "NPX_DF2" = subset_samples_list),
  normalization_type = c(NA_character_, "Subset"),
  normalize_to       = c(NA_character_, "1")
)
```

```
# Normalize
olink_normalization_n(norm_schema = norm_schema_subset)

#### Subset normalization  of two projects using all samples

# datasets
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find a suitable subset of samples from both projects, but exclude Olink
# controls and samples that fail QC.
df1_samples_all <- npx_df1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique()
df2_samples_all <- npx_df2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique()

# create named list
subset_samples_all_list <- list("DF1" = df1_samples_all,
                                "DF2" = df2_samples_all)

# create tibble for input
norm_schema_subset_all <- dplyr::tibble(
  order              = c(1, 2),
  name               = c("NPX_DF1", "NPX_DF2"),
  data               = list("NPX_DF1" = npx_df1,
                            "NPX_DF2" = npx_df2),
  samples            = list("NPX_DF1" = NA_character_,
                            "NPX_DF2" = subset_samples_all_list),
 normalization_type = c(NA_character_, "Subset"),
 normalize_to       = c(NA_character_, "1")
)

# Normalize
olink_normalization_n(norm_schema = norm_schema_subset_all)

#### Multi-project normalization using bridge and subset samples

## NPX data frames to bridge
```

```
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# manipulating the sample NPX datasets to create another two random ones
npx_df3 <- npx_data2 |>
  dplyr::mutate(SampleID = paste(SampleID, "_mod", sep = ""),
                PlateID = paste(PlateID, "_mod", sep = ""),
                NPX = sample(x = NPX, size = dplyr::n(), replace = FALSE)) |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

npx_df4 <- npx_data1 |>
  dplyr::mutate(SampleID = paste(SampleID, "_mod2", sep = ""),
                PlateID = paste(PlateID, "_mod2", sep = ""),
                NPX = sample(x = NPX, size = dplyr::n(), replace = FALSE)) |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

## samples to use for normalization
# Bridge samples with same identifiers between npx_df1 and npx_df2
overlap_samples <- dplyr::intersect(unique(npx_df1$SampleID),
                                    unique(npx_df2$SampleID))
overlap_samples_df1_df2 <- list("DF1" = overlap_samples,
                                "DF2" = overlap_samples)
rm(overlap_samples)

# Bridge samples with different identifiers between npx_df2 and npx_df3
overlap_samples_df2_df3 <- list("DF1" = sample(x = unique(npx_df2$SampleID),
                                               size = 10,
                                               replace = FALSE),
                                "DF2" = sample(x = unique(npx_df3$SampleID),
                                               size = 10,
                                               replace = FALSE))

# Samples to use for intensity normalization between npx_df4 and the
# normalized dataset of npx_df1 and npx_df2
overlap_samples_df12_df4 <- list("DF1" = sample(x = c(unique(npx_df1$SampleID),
                                                      unique(npx_df2$SampleID)),
                                                size = 100,
                                                replace = FALSE) |>
                                   unique(),
                                 "DF2" = sample(x = unique(npx_df4$SampleID),
                                                size = 40,
                                                replace = FALSE))
```

```
# create tibble for input
norm_schema_n <- dplyr::tibble(
  order              = c(1, 2, 3, 4),
  name               = c("NPX_DF1", "NPX_DF2", "NPX_DF3", "NPX_DF4"),
  data               = list("NPX_DF1" = npx_df1,
                            "NPX_DF2" = npx_df2,
                            "NPX_DF3" = npx_df3,
                            "NPX_DF4" = npx_df4),
  samples            = list("NPX_DF1" = NA_character_,
                            "NPX_DF2" = overlap_samples_df1_df2,
                            "NPX_DF3" = overlap_samples_df2_df3,
                            "NPX_DF4" = overlap_samples_df12_df4),
  normalization_type = c(NA_character_, "Bridge", "Bridge", "Subset"),
  normalize_to       = c(NA_character_, "1", "2", "1,2")
)

olink_normalization_n(norm_schema = norm_schema_n)
```

`olink_normalization_n_check` *An internal function to perform checks on the input of the function olink_normalization_n.*

Description

An internal function to perform checks on the input of the function olink_normalization_n.

Usage

```
olink_normalization_n_check(norm_schema)
```

Arguments

`norm_schema`	A tibble with more than 1 rows and (strictly) the following columns: "order", "name", "data", "samples", "normalization_type", "normalize_to". See above for details of the structure of the data frame. See details in help for olink_normalization_n. (required)

Value

a character message. If the message is "TRUE" then all checks passed, otherwise an error message will be printed.

`olink_normalization_product_format` *Formatting the output of olink_normalization_product for seamless use with downstream analysis functions.*

Description

Replaces the NPX values of the non-reference project by the Median Centered or QS Normalized NPX, according to the Bridging Recommendation. Edits the BridgingRecommendation column to indicate whether an assay is NotBridgeable, NotOverlapping, MedianCentering, or QuantileSmoothing bridged. Replaces OlinkID by the concatenation of the Explore HT and Explore 3072 OlinkIDs to record the OlinkIDs from both projects for bridgeable assays. Assays that are NotBridgeable or NotOverlapping retain their original non-reference OlinkIDs and NPX values. Replaces SampleID with the concatenation of SampleID and Project to make unique sample IDs for downstream analysis. Removes internal and external controls. Removes MedianCenteredNPX, QSNormalizedNPX, OlinkID_E3072 columns.

Usage

```
olink_normalization_product_format(
  df_norm,
  df1,
  df1_project_nr,
  df2,
  df2_project_nr,
  reference_project,
  ref_product
)
```

Arguments

`df_norm`	A "tibble" of Olink data in long format resulting from the olink_normalization_product function.
`df1`	First dataset to be used for normalization, pre-normalization. Must match df1 used in olink_normalization product bridging.
`df1_project_nr`	Project name of first dataset. Must match name used in olink_normalization product bridging.
`df2`	Second dataset to be used for normalization, pre-normalization. Must match df2 used in olink_normalization product bridging.
`df2_project_nr`	Project name of second dataset. Must match name used in olink_normalization product bridging.
`reference_project`	Project name of reference project. Must match name used in olink_normalization product bridging and be one of df1_project_nr or df2_project_nr.
`ref_product`	Name of reference product. Must be one of "HT" or "Reveal".

Value

A "tibble" of Olink data in long format containing both input datasets with the bridged NPX quantifications, with the above modifications.

Author(s)

Danai G. Topouza Klev Diamanti

Examples

```
# bridge samples
bridge_samples <- intersect(
  x = unique(OlinkAnalyze:::data_ht_small$SampleID),
  y = unique(OlinkAnalyze:::data_3k_small$SampleID)
) |>
  (\(x) x[!grepl("CONTROL", x)])()

# run olink_normalization
df_norm <- olink_normalization(
  df1 = OlinkAnalyze:::data_ht_small,
  df2 = OlinkAnalyze:::data_3k_small,
  overlapping_samples_df1 = bridge_samples,
  df1_project_nr = "Explore HT",
  df2_project_nr = "Explore 3072",
  reference_project = "Explore HT"
)

# format output
OlinkAnalyze:::olink_normalization_product_format(
  df_norm = df_norm,
  df1 = OlinkAnalyze:::data_ht_small,
  df2 = OlinkAnalyze:::data_3k_small,
  df1_project_nr = "Explore HT",
  df2_project_nr = "Explore 3072",
  reference_project = "Explore HT",
  ref_product = "HT"
)
```

`olink_normalization_project_name_check`

An internal function to perform checks on the input project names in the functions olink_normalization_bridge and olink_normalization_subset. The function is expected to run all checks on project names to make sure that normalization can be performed smoothly. It should work independently of the function calling it.

Description

An internal function to perform checks on the input project names in the functions olink_normalization_bridge and olink_normalization_subset. The function is expected to run all checks on project names to make sure that normalization can be performed smoothly. It should work independently of the function calling it.

Usage

```
olink_normalization_project_name_check(
  project_1_name,
  project_2_name,
  project_ref_name
)
```

Arguments

project_1_name	Name of project 1 (required)
project_2_name	Name of project 2 (required)
project_ref_name	Name of reference project (required)

Value

a character message. If the message is "TRUE" then all checks passed, otherwise an error message will be printed.

---

olink_normalization_qs

Quantile smoothing normalization of all proteins between two NPX projects.

Description

This function uses bridge samples to map quantiles of the non-reference dataset to the ones of the reference dataset. Mapped quantiles are used to transform the quantifications of the the non-reference dataset to the reference.

Usage

```
olink_normalization_qs(
  lst_df,
  ref_cols,
  not_ref_cols,
  bridge_samples,
  ref_product
)
```

Arguments

lst_df	A named list of the 2 input datasets. First element should be the reference dataset from Olink Explore HT and the second element should originate from Olink Explore 3072. (required)
ref_cols	A named list with the column names to use. Exported from olink_norm_input_check. (required)
not_ref_cols	A named list with the column names from the non-reference dataset. Exported from olink_norm_input_check. (required)
bridge_samples	Character vector of samples to be used for the quantile mapping. (required)
ref_product	Name of reference product. Must be one of "HT" or "Reveal".

Details

In the case when a study is separated into multiple projects, an additional normalization step is needed to allow the data to be comparable across projects. Across different Olink products, some of the assays exist in corresponding but distinct NPX spaces. For those assays, the median of paired differences is insufficient for bridging as it only considers one anchor point (the median/50% quantile). Instead, quantile smoothing (QS) using multiple anchor points (5%, 10%, 25%, 50%, 75%, 90% and 95% quantiles) is favored to map the Explore 3072 data to the Explore HT distribution. The `olink_normalization_qs()` performs quantile smoothing bridging normalization between datasets from two Olink products (for example Olink Explore 3072 and Olink Explore HT) by performing the following steps:

- An empirical cumulative distribution function is used to map datapoints for the bridging samples from one product to the equivalent space in the other product.
- A spline regression model is constructed using unmapped and mapped data from one product, using anchor points from the quantiles defined above.
- The spline regression model is used to predict the normalized NPX values for all datapoints

More information on quantile smoothing and between product normalization can be found in the Bridging Olink Explore 3072 to Olink Explore HT tutorial.

Value

A "tibble" of Olink data in long format containing both input datasets with the quantile normalized quantifications.

Author(s)

Amrita Kar Marianne Sandin Masoumeh Sheikhi Klev Diamanti

Examples

```
# Bridge samples
bridge_samples <- intersect(
  x = unique(OlinkAnalyze:::data_ht_small$SampleID),
  y = unique(OlinkAnalyze:::data_3k_small$SampleID)
```

```
) |>
  (\(x) x[!grepl("CONTROL", x)])()

# Run the internal function olink_norm_input_check
check_norm <- OlinkAnalyze:::olink_norm_input_check(
  df1 = OlinkAnalyze:::data_ht_small,
  df2 = OlinkAnalyze:::data_3k_small,
  overlapping_samples_df1 = bridge_samples,
  overlapping_samples_df2 = NULL,
  df1_project_nr = "P1",
  df2_project_nr = "P2",
  reference_project = "P1",
  reference_medians = NULL
)

# Named list of input datasets
lst_df <- list(
  check_norm$ref_df,
  check_norm$not_ref_df
)
names(lst_df) <- c(check_norm$ref_name, check_norm$not_ref_name)

ref_cols <- check_norm$ref_cols
not_ref_cols <- check_norm$not_ref_cols

qs_result <- OlinkAnalyze:::olink_normalization_qs(
 lst_df = lst_df,
 ref_cols = ref_cols,
 not_ref_cols = not_ref_cols,
 bridge_samples = bridge_samples,
 ref_product = "HT"
)
```

`olink_normalization_sample_check`

An internal function to perform checks on the input samples in the functions olink_normalization_bridge and olink_normalization_subset. The function is expected to run all checks on SampleID to make sure that normalization can be performed smoothly. It should work independently of the function calling it.

Description

An internal function to perform checks on the input samples in the functions olink_normalization_bridge and olink_normalization_subset. The function is expected to run all checks on SampleID to make sure that normalization can be performed smoothly. It should work independently of the function calling it.

Usage

```
olink_normalization_sample_check(
  list_samples,
  check_mode,
  project_1_all_samples,
  project_2_all_samples
)
```

Arguments

list_samples	Named list of 2 arrays containing SampleID of the subset or bridge samples to be used for normalization. The names of the two arrays should be DF1 and DF2 corresponding to projects 1 and 2, respectively. (required)
check_mode	Flag "bridge" or "subset" indicating the type of normalization the check should be tailored to (required)
project_1_all_samples	Array of all samples from project 1 (required)
project_2_all_samples	Array of all samples from project 2 (required)

Value

a character message. If the message is "TRUE" then all checks passed, otherwise an error message will be printed.

---

olink_normalization_subset

Subset normalization of all proteins between two NPX projects.

Description

Normalizes two NPX projects (data frames) using all or a subset of samples.

Usage

```
olink_normalization_subset(
  project_1_df,
  project_2_df,
  reference_samples,
  project_1_name = "P1",
  project_2_name = "P2",
  project_ref_name = "P1"
)
```

Arguments

project_1_df	Data frame of the first project (required).
project_2_df	Data frame of the second project (required).
reference_samples	Named list of 2 arrays containing SampleID of the subset of samples to be used for the calculation of median NPX within each project. The names of the two arrays should be DF1 and DF2 corresponding to projects 1 and 2, respectively. Arrays do not need to be of equal length and the order the samples appear in does not play any role. (required)
project_1_name	Name of the first project (default: P1).
project_2_name	Name of the second project (default: P2).
project_ref_name	Name of the project to be used as reference set. Needs to be one of the project_1_name or project_2_name. It marks the project to which the other project will be adjusted to (default: P1).

Details

This function is a wrapper of olink_normalization.

In subset normalization one of the projects is adjusted to another using a subset of all samples from each. Please note that the subsets of samples are not expected to be replicates of each other or to have the SampleID. Adjustment between the two projects is made using the assay-specific differences in median between the subsets of samples from the two projects. The two data frames are inputs project_1_df and project_2_df, the one being adjusted to is specified in the input project_ref_name and the shared samples are specified in reference_samples.

A special case of subset normalization is to use all samples (except control samples) from each project as a subset.

Value

A "tibble" of NPX data in long format containing normalized NPX values, including adjustment factors and name of project.

Examples

```
#### Subset normalization

# datasets
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
```

```
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find a suitable subset of samples from both projects, but exclude Olink
# controls and samples that fail QC.
df1_samples <- npx_df1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique() |>
  sample(size = 16, replace = FALSE)
df2_samples <- npx_df2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique() |>
  sample(size = 16, replace = FALSE)

# create named list
subset_samples_list <- list("DF1" = df1_samples,
                            "DF2" = df2_samples)

# Normalize
olink_normalization_subset(project_1_df = npx_df1,
                           project_2_df = npx_df2,
                           reference_samples = subset_samples_list,
                           project_1_name = "P1",
                           project_2_name = "P2",
                           project_ref_name = "P1")


#### Special case of subset normalization using all samples

# datasets
npx_df1 <- npx_data1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")
npx_df2 <- npx_data2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::select(-Project) |>
  dplyr::mutate(Normalization = "Intensity")

# Find a suitable subset of samples from both projects, but exclude Olink
# controls and samples that fail QC.
df1_samples_all <- npx_df1 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
```

```
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique()
df2_samples_all <- npx_df2 |>
  dplyr::filter(!stringr::str_detect(SampleID, "CONTROL_")) |>
  dplyr::group_by(SampleID) |>
  dplyr::filter(all(QC_Warning == 'Pass')) |>
  dplyr::pull(SampleID) |>
  unique()

# create named list
subset_samples_all_list <- list("DF1" = df1_samples_all,
                            "DF2" = df2_samples_all)

# Normalize
olink_normalization_subset(project_1_df = npx_df1,
                           project_2_df = npx_df2,
                           reference_samples = subset_samples_all_list,
                           project_1_name = "P1",
                           project_2_name = "P2",
                           project_ref_name = "P1")
```

`olink_norm_input_assay_overlap` *Check datasets and reference_medians for Olink identifiers not shared across datasets.*

Description

Check *datasets* and *reference_medians* for Olink identifiers not shared across datasets.

Usage

```
olink_norm_input_assay_overlap(
  lst_df,
  reference_medians,
  lst_cols,
  norm_mode = norm_mode
)
```

Arguments

`lst_df`	Named list of datasets to be normalized.
`reference_medians`	Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.
`lst_cols`	Named list of vectors with the required column names for each dataset in *lst_df*.

norm_mode — Character string indicating the type of normalization to be performed. Expecting one of bridge, subset, ref_median or norm_cross_product. # nolint line_length_linter

Value

A named list containing *lst_df* and *reference_medians* will assays shared across all datasets.

Author(s)

Klev Diamanti

`olink_norm_input_check` *Check inputs of* `olink_normalization` *function.*

Description

This function is a wrapper of multiple help functions which check the inputs of the `olink_normalization` function.

Usage

```
olink_norm_input_check(
  df1,
  df2,
  overlapping_samples_df1,
  overlapping_samples_df2,
  df1_project_nr,
  df2_project_nr,
  reference_project,
  reference_medians
)
```

Arguments

df1	First dataset to be used in normalization (required).
df2	Second dataset to be used in normalization.
overlapping_samples_df1	Samples to be used for adjustment factor calculation in df1 (required).
overlapping_samples_df2	Samples to be used for adjustment factor calculation in df2.
df1_project_nr	Project name of first dataset (df1).
df2_project_nr	Project name of first dataset (df2).
reference_project	Project name of reference_project. Should be one of *df1_project_nr* or *df2_project_nr*. Indicates the project to which the other project is adjusted to.

`reference_medians`
: Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.

Details

The following checks are performed:

- olink_norm_input_validate:
  - Determines the normalization to be performed by intersecting inputs with internal global variable `olink_norm_mode_combos`.
  - Returns the type of normalization to be performed from `olink_norm_modes`.
  - Message with the normalization type.
  - Error message if input is invalid.
- olink_norm_input_class:
  - Checks if all inputs are of the expected class:
    - `df1`, `df2` and `reference_medians`: tibble or R6 ArrowObject
    - `overlapping_samples_df1`, `overlapping_samples_df2`, `df1_project_nr`, `df2_project_nr` and `reference_project`: Character vector
  - Also checks the validity of names of project and reference project.
  - Error if invalid input classes are detected.
- olink_norm_input_check_df_cols:
  - Detects the column names of input datasets `df1` and `df2` to allow for alternative names.
  - Returns named list of column names to use downstream.
  - Warning if `Normalization` column missing from all datasets.
  - Warning if `LOD` is missing or if there are multiple `LOD` columns.
  - Error if required columns are missing.
  - Error if not all input datasets have or lack `Normalization` column.
  - Error if input datasets have been quantified with different methods.
- olink_norm_input_ref_medians:
  - Checks validity of dataset containing `reference_medians`.
  - Error if required columns are missing based on `olink_norm_ref_median_cols`.
  - Error if columns are not of the correct class bases on `olink_norm_ref_median_cols`.
  - Error if there duplicate assay identifiers.
- olink_norm_input_check_samples:
  - Check character vectors of reference sample identifiers for:
    - Being present in `df1` and/or `df2`.
    - Duplicate identifiers.
- olink_norm_input_clean_assays:
  - Returns a named list with the updated `df1`, `df2` and/or `reference_medians`.
  - Removes assays that are not of the format OID followed by 5 digits.
  - Removes assays that are marked with `Normalization = EXCLUDED`.

- olink_norm_input_assay_overlap:
  - Returns a named list with the updated df1, df2 and/or reference_medians.
  - Remove assays not shared between df1 and df2, or between df1 and reference_medians.
- olink_norm_input_norm_method:
  - Check if all assays in df1 and df2 have been originally normalized with the same method "Intensity" or "Plate control".
  - Warning is thrown if not.

Value

Named list of updated inputs to use for normalization:

- df1: dataset df1.
- df2: NULL if reference median normalization, or dataset df2.
- overlapping_samples_df1: character vector of reference samples from df1.
- overlapping_samples_df2: NULL if reference median normalization, or character vector of reference samples from df1.
- df1_project_nr: name of df1 project.
- df2_project_nr: NULL if reference median normalization, or name of df2 project.
- reference_project: NULL if reference median normalization, or name of reference project.
- reference_medians: NULL if bridge or subset normalization, or dataset with reference_medians.
- df1_cols: column names of df1 to use downstream.
- df2_cols: NULL if reference median normalization, or column names of df2 to use downstream.
- norm_mode: one of bridge, subset, ref_median, and norm_cross_product indicating the normalization to be performed.

Author(s)

Klev Diamanti

---

olink_norm_input_check_df_cols

Check columns of a list of datasets to be normalized.

Description

This function takes as input a named list of datasets and checks if their columns allow the normalization to be performed. The input may contain "tibble", "ArrowTable" or a mixture of them.

Usage

```
olink_norm_input_check_df_cols(lst_df)
```

Arguments

lst_df	Named list of datasets to be normalized.

Value

Named list of vectors with the required column names for each dataset in *lst_df* if no error.

Author(s)

Klev Diamanti

Examples

```
# One dataset
OlinkAnalyze:::olink_norm_input_check_df_cols(
  lst_df = list(
    "p1" = npx_data1
  ) |>
    lapply(function(l_df) {
      l_df |>
        dplyr::select(
          -dplyr::any_of(c("Normalization"))
        )
     })
  )

# Two datasets
OlinkAnalyze:::olink_norm_input_check_df_cols(
  lst_df = list(
    "p1" = npx_data1,
    "p2" = npx_data2
  ) |>
    lapply(function(l_df) {
      l_df |>
        dplyr::select(
          -dplyr::any_of(c("Normalization"))
        )
     })
  )

# Multiple datasets
OlinkAnalyze:::olink_norm_input_check_df_cols(
  lst_df = list(
    "p1" = npx_data1,
    "p2" = npx_data2,
    "p3" = npx_data1,
    "p4" = npx_data2
  ) |>
    lapply(function(l_df) {
      l_df |>
        dplyr::select(
          -dplyr::any_of(c("Normalization"))
```

```
      )
    })
)
```

`olink_norm_input_check_samples` *Check reference samples to be used for normalization*

Description

This function takes as input a two named lists of character vectors with matching names and checks the validity of the reference samples. In case of 1 set of df samples, then all checks are skipped as reference median normalization is to be performed.

Usage

```
olink_norm_input_check_samples(lst_df_samples, lst_ref_samples, norm_mode)
```

Arguments

`lst_df_samples`	Named list of all sample identifiers from datasets to be normalized.
`lst_ref_samples`	Named list of reference sample identifiers to be used for normalization.
`norm_mode`	Character string indicating the type of normalization to be performed. Expecting one of bridge, subset, ref_median or norm_cross_product. # nolint line_length_linter

Value

`NULL` if no warning or error.

Author(s)

Klev Diamanti

Examples

```
# Reference median normalization
OlinkAnalyze:::olink_norm_input_check_samples(
  lst_df_samples = list(
    "p1" = unique(npx_data1$SampleID)
  ),
  lst_ref_samples = list(
    "p1" = npx_data1 |>
      dplyr::filter(
        !grepl(pattern = "CONTROL_SAMPLE",
```

```
        x = .data[["SampleID"]],
        fixed = TRUE)
      ) |>
      dplyr::pull(.data[["SampleID"]]) |>
      unique() |>
      sort() |>
      head(n = 6L)
  ),
  norm_mode = "ref_median"
)

# Bridge normalization
ref_samples_bridge <- intersect(x = npx_data1$SampleID,
                                y = npx_data2$SampleID) |>
  (\(x) x[!grepl(pattern = "CONTROL_SAMPLE", x = x, fixed = TRUE)])()

OlinkAnalyze:::olink_norm_input_check_samples(
  lst_df_samples = list(
    "p1" = unique(npx_data1$SampleID),
    "p2" = unique(npx_data2$SampleID)
  ),
  lst_ref_samples = list(
    "p1" = ref_samples_bridge,
    "p2" = ref_samples_bridge
  ),
  norm_mode = "bridge"
)

# Subset normalization
ref_samples_subset_1 <- npx_data1 |>
  dplyr::filter(
    !grepl(pattern = "CONTROL_SAMPLE",
           x = .data[["SampleID"]],
           fixed = TRUE)
    & .data[["QC_Warning"]] == "Pass"
  ) |>
  dplyr::pull(
    .data[["SampleID"]]
  ) |>
  unique()
ref_samples_subset_2 <- npx_data2 |>
  dplyr::filter(
    !grepl(pattern = "CONTROL_SAMPLE",
           x = .data[["SampleID"]],
           fixed = TRUE)
    & .data[["QC_Warning"]] == "Pass"
  ) |>
  dplyr::pull(
    .data[["SampleID"]]
  ) |>
  unique()

OlinkAnalyze:::olink_norm_input_check_samples(
```

```
  lst_df_samples = list(
    "p1" = unique(npx_data1$SampleID),
    "p2" = unique(npx_data2$SampleID)
  ),
  lst_ref_samples = list(
    "p1" = ref_samples_subset_1,
    "p2" = ref_samples_subset_2
  ),
  norm_mode = "subset"
)
```

`olink_norm_input_class` *Check classes of input in olink_normalization function*

Description

Check if *df1* , *df2* and/or *reference_medians* are tibble or ArrowDataset datasets; if *overlapping_samples_df1* and/or *overlapping_samples_df2* are character vectors; and if *df1_project_nr*, *df2_project_nr* and/or *reference_project* are scalar character vectors.

Usage

```
olink_norm_input_class(
  df1,
  df2,
  overlapping_samples_df1,
  overlapping_samples_df2,
  df1_project_nr,
  df2_project_nr,
  reference_project,
  reference_medians,
  norm_mode
)
```

Arguments

`df1`	First dataset to be used in normalization (required).
`df2`	Second dataset to be used in normalization.
`overlapping_samples_df1`	Samples to be used for adjustment factor calculation in df1 (required).
`overlapping_samples_df2`	Samples to be used for adjustment factor calculation in df2.
`df1_project_nr`	Project name of first dataset (df1).
`df2_project_nr`	Project name of first dataset (df2).

`reference_project`
: Project name of reference_project. Should be one of *df1_project_nr* or *df2_project_nr*. Indicates the project to which the other project is adjusted to.

`reference_medians`
: Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.

`norm_mode`
: Scalar character from *olink_norm_modes* with the normalization to be performed. Output from `olink_norm_input_validate`.

Value

`NULL` unless there is an error

Author(s)

Klev Diamanti

`olink_norm_input_clean_assays`
: *Check datasets and reference_medians for unexpected Olink identifiers or excluded assays*

Description

Check *datasets* and *reference_medians* for unexpected Olink identifiers or excluded assays

Usage

```
olink_norm_input_clean_assays(lst_df, reference_medians, lst_cols, norm_mode)
```

Arguments

`lst_df`
: Named list of datasets to be normalized.

`reference_medians`
: Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.

`lst_cols`
: Named list of vectors with the required column names for each dataset in *lst_df*.

`norm_mode`
: Character string indicating the type of normalization to be performed. Expecting one of bridge, subset, ref_median or norm_cross_product. # nolint line_length_linter

Value

A named list containing *lst_df* and *reference_medians* stripped from unexpected Olink identifiers or excluded assays

Author(s)

Klev Diamanti

`olink_norm_input_cross_product` *Check if bridge or cross-platform normalization*

Description

A function to check whether we are to perform simple bridge normalization, or cross-platform (Olink Explore 3072 - Olink Explore HT/Olink Reveal) normalization.

The function uses the internal dataset *eHT_e3072_mapping* to determine the product source of each dataset. If both datasets originate from the same Olink product, then it will return bridge. If the datasets to be normalized originate from Olink Explore HT and Olink Explore 3072 or Olink Reveal and Olink Explore 3072, it will return norm_cross_product. In any other case an error is thrown.

Usage

```
olink_norm_input_cross_product(
  lst_df,
  lst_cols,
  reference_project,
  product_ids,
  ref_ids
)
```

Arguments

`lst_df`	Named list of datasets to be normalized.
`lst_cols`	Named list of vectors with the required column names for each dataset in *lst_df*.
`reference_project`	Project name of reference_project. Should be one of *df1_project_nr* or *df2_project_nr*. Indicates the project to which the other project is adjusted to.
`product_ids`	Named character vector with the Olink product name that each input dataset matches to.
`ref_ids`	Named character vector with *df1_project_nr* and *df2_project_nr* marked as "ref" and "not_ref".

Value

Character string indicating the type of normalization to be performed. One of bridge, subset, ref_median or norm_cross_product. # nolint line_length_linter And the updated list of datasets in case of cross-platform normalization.

Author(s)

Klev Diamanti

olink_norm_input_norm_method *Check datasets and reference_medians for Olink identifiers not shared across datasets.*

Description

Check *datasets* and *reference_medians* for Olink identifiers not shared across datasets.

Usage

```
olink_norm_input_norm_method(lst_df, lst_cols)
```

Arguments

lst_df	Named list of datasets to be normalized.
lst_cols	Named list of vectors with the required column names for each dataset in *lst_df*.

Value

NULL if all assays are normalized with the same approach.

Author(s)

Klev Diamanti; Kathleen Nevola

olink_norm_input_ref_medians *Check datasets of reference_medians*

Description

Check datasets of *reference_medians*

Usage

```
olink_norm_input_ref_medians(reference_medians)
```

Arguments

reference_medians	Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.

Value

NULL otherwise error.

Author(s)

Klev Diamanti

`olink_norm_input_validate` *Validate inputs of normalization function*

Description

This function takes as input some of the inputs of the Olink normalization function and checks the validity of the input.

Usage

```
olink_norm_input_validate(
  df1,
  df2,
  overlapping_samples_df1,
  overlapping_samples_df2,
  reference_medians
)
```

Arguments

`df1`	First dataset to be used in normalization (required).
`df2`	Second dataset to be used in normalization.
`overlapping_samples_df1`	Samples to be used for adjustment factor calculation in df1 (required).
`overlapping_samples_df2`	Samples to be used for adjustment factor calculation in df2.
`reference_medians`	Dataset with columns "OlinkID" and "Reference_NPX". Used for reference median normalization.

Details

Depending on the input the function will return:

- **Error**: if the required components are lacking from the input or if the normalization cannot be performed.
- **Warning**: if the normalization can be determined but extra inputs are provided. This will be followed by a message and the type of normalization to be performed.
- **Message**: Information about the type of normalization to be performed.

Note that input are passed directly from the main `olink_normalization` function.

Value

Scalar character from *olink_norm_modes* if normalization can be determined from the input, otherwise see details.

Author(s)

Klev Diamanti

`olink_norm_product_id` *Identify names of product for each project*

Description

Identify names of product for each project

Usage

```
olink_norm_product_id(lst_df, lst_cols)
```

Arguments

`lst_df`	Named list of datasets to be normalized.
`lst_cols`	Named list of vectors with the required column names for each dataset in *lst_df*.

Value

Named character vector with the Olink product name that each input dataset matches to.

Author(s)

Kathy Nevola Klev Diamanti

`olink_norm_reference_id`
Identify reference project.

Description

Identify reference project.

Usage

```
olink_norm_reference_id(lst_product, reference_project)
```

Arguments

`lst_product`	Named character vector with the Olink product name that each input dataset matches to.
`reference_project`	Project name of reference_project. Should be one of *df1_project_nr* or *df2_project_nr*. Indicates the project to which the other project is adjusted to.

Value

Named character vector with *df1_project_nr* and *df2_project_nr* marked as "ref" and "not_ref".

Author(s)

Kathy Nevola Klev Diamanti

`olink_one_non_parametric` *Function which performs a Kruskal-Wallis Test or Friedman Test per protein*

Description

Performs an Kruskal-Wallis Test for each assay (by OlinkID) in every panel using stats::kruskal.test. Performs an Friedman Test for each assay (by OlinkID) in every panel using rstatix::friedman_test. The function handles factor variable.

Samples that have no variable information or missing factor levels are automatically removed from the analysis (specified in a message if verbose = TRUE). Character columns in the input dataframe are automatically converted to factors (specified in a message if verbose = T). Numerical variables are not converted to factors. If a numerical variable is to be used as a factor, this conversion needs to be done on the dataframe before the function call.

Inference is specified in a message if verbose = TRUE.
The formula notation of the final model is specified in a message if verbose = TRUE.

Adjusted p-values are calculated by stats::p.adjust according to the Benjamini & Hochberg (1995) method (“fdr”). The threshold is determined by logic evaluation of Adjusted_pval < 0.05.

Usage

```
olink_one_non_parametric(
  df,
  variable,
  dependence = FALSE,
  subject = NULL,
  verbose = TRUE
)
```

Arguments

df	NPX or Quantified_value data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
variable	Single character value.
dependence	Boolean. Default: FALSE. When the groups are independent, the kruskal-Wallis will run, when the groups are dependent, the Friedman test will run.
subject	Group information for the repeated measurement. If (dependence = TRUE), this parameter need to be specified.
verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A tibble containing the Kruskal-Wallis Test or Friedman Test results for every protein.

Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- df: "numeric" degrees of freedom
- method: "character" which method was used
- statistic: "named numeric" the value of the test statistic with a name describing it
- p.value: "numeric" p-value for the test
- Adjusted_pval: "numeric" adjusted p-value for the test (Benjamini&Hochberg)
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

# One-way Kruskal-Wallis Test
try({ # May fail if dependencies are not installed
kruskal_results <- olink_one_non_parametric(df = npx_data1,
                                            variable = "Site")
})

#Friedman Test
friedman_results <- olink_one_non_parametric(df = npx_data1,
                                             variable = "Time",
                                             subject = "Subject",
                                             dependence = TRUE)
```

`olink_one_non_parametric_posthoc` *Function which performs posthoc test per protein for the results from Friedman or Kruskal-Wallis Test.*

Description

Performs a posthoc test using rstatix::wilcox_test or FSA::dunnTest with Benjamini-Hochberg p-value adjustment per assay (by OlinkID) for each panel at confidence level 0.95. See `olink_one_non_parametric` for details of input notation.

The function handles both factor and numerical variables.

Usage

```
olink_one_non_parametric_posthoc(
  df,
  olinkid_list = NULL,
  variable,
  test = "kruskal",
  subject = "Subject",
  verbose = TRUE
)
```

Arguments

`df`	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
`olinkid_list`	Character vector of OlinkID's on which to perform post hoc analysis. If not specified, all assays in df are used.
`variable`	Single character value or character array.
`test`	Single character value indicates running the post hoc test for friedman or kruskal.
`subject`	Group information for the repeated measurement. If (dependence = TRUE), this parameter need to be specified.
`verbose`	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

Tibble of posthoc tests for specified effect, arranged by ascending adjusted p-values.

Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID

- Panel: "character" Name of Olink Panel
- term: "character" term in model
- contrast: "character" the groups that were compared
- estimate: "numeric" the value of the test statistic with a name describing it
- Adjusted_pval: "numeric" adjusted p-value for the test
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

try({ # May fail if dependencies are not installed
# One-way Kruskal-Wallis Test
kruskal_results <- olink_one_non_parametric(df = npx_data1,
                                            variable = "Site")
})

#Friedman Test
friedman_results <- olink_one_non_parametric(df = npx_data1,
                                            variable = "Time",
                                            subject = "Subject",
                                            dependence = TRUE)

#Posthoc test for the results from Friedman Test
friedman_posthoc_results <- olink_one_non_parametric_posthoc(npx_data1,
                                                             variable = "Time",
                                                             test = "friedman",
                                                    olinkid_list = {friedman_results %>%
                                                   filter(Threshold == 'Significant') %>%
                                                              dplyr::select(OlinkID) %>%
                                                              distinct() %>%
                                                              pull()})
```

`olink_ordinalRegression`

Function which A two-way ordinal analysis of variance can address an experimental design with two independent variables, each of which is a factor variable. The main effect of each independent variable can be tested, as well as the effect of the interaction of the two factors.

Description

Performs an ANOVA F-test for each assay (by OlinkID) in every panel using stats::Anova and Type III sum of squares. Dependent variable will be treated as ordered factor. The function handles only factor and/or covariates.

Samples that have no variable information or missing factor levels are automatically removed from the analysis (specified in a message if verbose = T). Character columns in the input dataframe are automatically converted to factors (specified in a message if verbose = T). Crossed analysis, i.e. A*B formula notation, is inferred from the variable argument in the following cases:

- c('A','B')
- c('A: B')
- c('A: B', 'B') or c('A: B', 'A')

Inference is specified in a message if verbose = T.
The formula notation of the final model is specified in a message if verbose = T.

Adjusted p-values are calculated by stats::p.adjust according to the Benjamini & Hochberg (1995) method ("fdr"). The threshold is determined by logic evaluation of Adjusted_pval < 0.05. Covariates are not included in the p-value adjustment.

Usage

```
olink_ordinalRegression(
  df,
  variable,
  covariates = NULL,
  return.covariates = F,
  verbose = T
)
```

Arguments

df	NPX or Quantified_value data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':'/'*' notation.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':'/'*' notation. Crossed analysis will not be inferred from main effects.
return.covariates	Logical. Default: False. Returns F-test results for the covariates. Note: Adjusted p-values will be NA for the covariates.
verbose	Logical. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

A tibble containing the ANOVA results for every protein. The tibble is arranged by ascending p-values.

Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- statistic: "numeric" value of the statistic
- p.value: "numeric" nominal p-value
- Adjusted_pval: "numeric" adjusted p-value for the test
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

try({ # May fail if dependencies are not installed.
#Two-way Ordinal Regression with CLM.
#Results in model NPX~Treatment+Time+Treatment:Time.
   ordinalRegression_results <- olink_ordinalRegression(df = npx_data1,
                                                        variable="Treatment:Time")
})
```

---

olink_ordinalRegression_posthoc

Function which performs an posthoc test per protein.

Description

Performs a post hoc ANOVA test using emmeans::emmeans with Tukey p-value adjustment per assay (by OlinkID) for each panel at confidence level 0.95. See `olink_anova` for details of input notation.

The function handles both factor and numerical variables and/or covariates. The posthoc test for a numerical variable compares the difference in means of the ordinal outcome variable (default: NPX) for 1 standard deviation difference in the numerical variable, e.g. mean ordinal NPX at mean(numerical variable) versus mean NPX at mean(numerical variable) + 1*SD(numerical variable).

Usage

```
olink_ordinalRegression_posthoc(
  df,
  olinkid_list = NULL,
  variable,
```

```
  covariates = NULL,
  effect,
  effect_formula,
  mean_return = FALSE,
  post_hoc_padjust_method = "tukey",
  verbose = T
)
```

Arguments

df	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt, Panel and a factor with at least 3 levels.
olinkid_list	Character vector of OlinkID's on which to perform post hoc analysis. If not specified, all assays in df are used.
variable	Single character value or character array. Variable(s) to test. If length > 1, the included variable names will be used in crossed analyses . Also takes ':' notation.
covariates	Single character value or character array. Default: NULL. Covariates to include. Takes ':'/'*' notation. Crossed analysis will not be inferred from main effects.
effect	Term on which to perform post-hoc. Character vector. Must be subset of or identical to variable.
effect_formula	(optional) A character vector specifying the names of the predictors over which estimated marginal means are desired as defined in the emmeans package. May also be a formula. If provided, this will override the effect argument. See ?emmeans::emmeans() for more information.
mean_return	Boolean. If true, returns the mean of each factor level rather than the difference in means (default). Note that no p-value is returned for mean_return = TRUE and no adjustment is performed.
post_hoc_padjust_method	P-value adjustment method to use for post-hoc comparisons within an assay. Options include tukey, sidak, bonferroni and none.
verbose	Boolean. Default: True. If information about removed samples, factor conversion and final model formula is to be printed to the console.

Value

Tibble of posthoc tests for specified effect, arranged by ascending adjusted p-values.

#' Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- term: "character" term in model
- contrast: "character" the groups that were compared

- estimate: "numeric" difference in mean of the ordinal NPX between groups
- Adjusted_pval: "numeric" adjusted p-value for the test
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)
#Two-way Ordinal Regression.
#Results in model NPX~Treatment*Time.
try({ # May not work if dependencies are not installed.
ordinalRegression_results <- olink_ordinalRegression(df = npx_data1,
                              variable="Treatment:Time")

#Filtering out significant and relevant results.
significant_assays <- ordinalRegression_results %>%
  filter(Threshold == 'Significant' & term == 'Time') %>%
  select(OlinkID) %>%
  distinct() %>%
  pull()

#Posthoc test for the model NPX~Treatment*Time,
#on the effect Time.

#Posthoc
ordinalRegression_results_posthoc_results <- olink_ordinalRegression_posthoc(npx_data1,
                                                    variable=c("Treatment:Time"),
                                                    olinkid_list = significant_assays,
                                                    effect = "Time")
                                                    })
```

`olink_pal` *Olink color panel for plotting*

Description

Olink color panel for plotting

Usage

```
olink_pal(alpha = 1, coloroption = NULL)
```

Arguments

`alpha`	transparency (optional)
`coloroption`	string, one or more of the following: c('red', 'orange', 'yellow', 'green', 'teal', 'turqoise', 'lightblue', 'darkblue', 'purple', 'pink')

Value

A character vector of palette hex codes for colors

Examples

```
library(scales)

#Color matrices
show_col(olink_pal()(10), labels = FALSE)
show_col(olink_pal(coloroption = c('lightblue', 'green'))(2), labels = FALSE)

#Contour plot
filled.contour(volcano, color.palette = olink_pal(), asp = 1)
filled.contour(volcano, color.palette = hue_pal(), asp = 1)
```

---

olink_pathway_enrichment

Performs pathway enrichment using over-representation analysis (ORA) or gene set enrichment analysis (GSEA)

Description

This function performs enrichment analysis based on statistical test results and full data using clusterProfiler's gsea and enrich functions for MSigDB.

Usage

```
olink_pathway_enrichment(
  data,
  test_results,
  method = "GSEA",
  ontology = "MSigDb",
  organism = "human",
  pvalue_cutoff = 0.05,
  estimate_cutoff = 0
)
```

Arguments

data	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt,SampleID, QC_Warning, NPX, and LOD
test_results	a dataframe of statistical test results including Adjusted_pval and estimate columns.
method	Either "GSEA" (default) or "ORA"

ontology	Supports "MSigDb" (default), "KEGG", "GO", and "Reactome" as arguments. MSigDb contains C2 and C5 genesets. C2 and C5 encompass KEGG, GO, and Reactome.
organism	Either "human" (default) or "mouse"
pvalue_cutoff	(numeric) maximum Adjusted p-value cutoff for ORA filtering of foreground set (default = 0.05). This argument is not used for GSEA.
estimate_cutoff	(numeric) minimum estimate cutoff for ORA filtering of foreground set (default = 0) This argument is not used for GSEA.

Details

MSigDB is subset if the ontology argument is KEGG, GO, or Reactome. test_results must contain estimates for all assays. Posthoc results can be used but should be filtered for one contrast to improve interpretability. Alternative statistical results can be used as input as long as they include the columns "OlinkID", "Assay", and "estimate". A column named "Adjusted_pal" is also needed for ORA. Any statistical results that contains one estimate per protein will work as long as the estimates are comparable to each other.

clusterProfiler is originally developed by Guangchuang Yu at the School of Basic Medical Sciences at Southern Medical University.

T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu. clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. The Innovation. 2021, 2(3):100141. doi: 10.1016/j.xinn.2021.100141

NB: We strongly recommend to set a seed prior to running this function to ensure reproducibility of the results.

A few notes on Pathway Enrichment with Olink Data

It is important to note that sometimes the proteins that are assayed in Olink Panels are related to specific biological areas and therefore do not represent an unbiased overview of the proteome as a whole. Pathways can only interpreted based on the background/context they came from. For this reason, an estimate for all assays measured must be provided. Furthermore, certain pathways cannot come up based on Olink's coverage in this area. Additionally, if only the Inflammation panel was run, then the available pathways would be given based on a background of proteins related to inflammation. Both ORA and GSEA can provide mechanistic and disease related insight and are best to use when trying to uncover pathways/annotations of interest. It is recommended to only use pathway enrichment for hypothesis generating data, which is more well suited for data on the Explore platform or on multiple Target 96 panels. For smaller lists of proteins it may be more informative to use biological annotation in directed research, to discover which significant assay are related to keywords of interest.

Value

A data frame of enrichment results. Columns for ORA include:

- ID: "character" Pathway ID from MSigDB
- Description: "character" Description of Pathway from MSigDB
- GeneRatio: "character" ratio of input proteins that are annotated in a term

- BgRatio: "character" ratio of all genes that are annotated in this term
- pvalue: "numeric" p-value of enrichment
- p.adjust: "numeric" Adjusted p-value (Benjamini-Hochberg)
- qvalue: "numeric" false discovery rate, the estimated probability that the normalized enrichment score represents a false positive finding
- geneID: "character" list of input proteins (Gene Symbols) annotated in a term delimited by "/"
- Count: "integer" Number of input proteins that are annotated in a term

Columns for GSEA:

- ID: "character" Pathway ID from MSigDB
- Description: "character" Description of Pathway from MSigDB
- setSize: "integer" ratio of input proteins that are annotated in a term
- enrichmentScore: "numeric" Enrichment score, degree to which a gene set is over-represented at the top or bottom of the ranked list of genes
- NES: "numeric" Normalized Enrichment Score, normalized to account for differences in gene set size and in correlations between gene sets and expression data sets. NES can be used to compare analysis results across gene sets.
- pvalue: "numeric" p-value of enrichment
- p.adjust: "numeric" Adjusted p-value (Benjamini-Hochberg)
- qvalue: "numeric" false discovery rate, the estimated probability that the normalized enrichment score represents a false positive finding
- rank: "numeric" the position in the ranked list where the maximum enrichment score occurred
- leading_edge: "character" contains tags, list, and signal. Tags gives an indication of the percentage of genes contributing to the enrichment score. List gives an indication of where in the list the enrichment score is obtained. Signal represents the enrichment signal strength and combines the tag and list.
- core_enrichment: "character" list of input proteins (Gene Symbols) annotated in a term delimited by "/"

See Also

- olink_pathway_heatmap for generating a heat map of results
- olink_pathway_visualization for generating a bar graph of results

Examples

```
library(dplyr)
npx_df <- npx_data1 %>% filter(!grepl("control", SampleID, ignore.case = TRUE))
ttest_results <- olink_ttest(
  df = npx_df,
  variable = "Treatment",
  alternative = "two.sided"
)
try({ # This expression might fail if dependencies are not installed
```

```
gsea_results <- olink_pathway_enrichment(data = npx_data1, test_results = ttest_results)
ora_results <- olink_pathway_enrichment(
  data = npx_data1,
  test_results = ttest_results, method = "ORA"
)
}, silent = TRUE)
```

`olink_pathway_heatmap` *Creates a heatmap of selected pathways and proteins*

Description

Creates a heatmap of proteins related to pathways using enrichment results from olink_pathway_enrichment.

Usage

```
olink_pathway_heatmap(
  enrich_results,
  test_results,
  method = "GSEA",
  keyword = NULL,
  number_of_terms = 20
)
```

Arguments

`enrich_results`	data frame of enrichment results from olink_pathway_enrichment()
`test_results`	filtered results from statistical test with Assay, OlinkID, and estimate columns
`method`	method used in olink_pathway_enrichment ("GSEA" (default) or "ORA")
`keyword`	(optional) keyword to filter enrichment results on, if not specified, displays top terms
`number_of_terms`	number of terms to display, default is 20

Value

A heatmap as a ggplot object

See Also

- `olink_pathway_enrichment` for generating enrichment results
- `olink_pathway_visualization` for generating a bar graph of results

Examples

```
library(dplyr)
# Run t-test results (see olink_ttest documentation)
npx_df <- npx_data1 %>% filter(!grepl('control',SampleID, ignore.case = TRUE))
ttest_results <- olink_ttest(df=npx_df,
                             variable = 'Treatment',
                            alternative = 'two.sided')

try({ # This expression might fail if dependencies are not installed
#  Run olink_pathway_enrichment (see documentation)
gsea_results <- olink_pathway_enrichment(data = npx_data1, test_results = ttest_results)
ora_results <- olink_pathway_enrichment(data = npx_data1,
                                        test_results = ttest_results, method = "ORA")
olink_pathway_heatmap(enrich_results = gsea_results, test_results = ttest_results)
olink_pathway_heatmap(enrich_results = ora_results, test_results = ttest_results,
                      method = "ORA", keyword = "cell")
})
```

---

olink_pathway_visualization

Creates bargraph of top/selected enrichment terms from GSEA or ORA results from olink_pathway_enrichment()

Description

Pathways are ordered by increasing p-value (unadjusted)

Usage

```
olink_pathway_visualization(
  enrich_results,
  method = "GSEA",
  keyword = NULL,
  number_of_terms = 20
)
```

Arguments

enrich_results	data frame of enrichment results from olink_pathway_enrichment()
method	method used in olink_pathway_enrichment ("GSEA" (default) or "ORA")
keyword	(optional) keyword to filter enrichment results on, if not specified, displays top terms
number_of_terms	number of terms to display, default is 20

Value

A bargraph as a ggplot object

See Also

- olink_pathway_enrichment for generating enrichment results
- olink_pathway_heatmap for generating a heat map of results

Examples

```
library(dplyr)
# Run olink_ttest or other stats test (see documentaiton )
npx_df <- npx_data1 %>% filter(!grepl('control',SampleID, ignore.case = TRUE))
ttest_results <- olink_ttest(df=npx_df,
                             variable = 'Treatment',
                             alternative = 'two.sided')

try({ # This expression might fail if dependencies are not installed
# Run olink_pathway_enrichment (see documentation)
gsea_results <- olink_pathway_enrichment(data = npx_data1, test_results = ttest_results)
ora_results <- olink_pathway_enrichment(data = npx_data1,
                                        test_results = ttest_results, method = "ORA")
olink_pathway_visualization(enrich_results = gsea_results)
olink_pathway_visualization(enrich_results = gsea_results, keyword = "immune")
olink_pathway_visualization(enrich_results = ora_results, method = "ORA", number_of_terms = 15)
})
```

---

olink_pca_plot *Function to plot a PCA of the data*

Description

Generates a PCA projection of all samples from NPX data along two principal components (default PC2 vs. PC1) including the explained variance and dots colored by QC_Warning using stats::prcomp and ggplot2::ggplot.

Usage

```
olink_pca_plot(
  df,
  color_g = "QC_Warning",
  x_val = 1,
  y_val = 2,
  label_samples = FALSE,
  drop_assays = FALSE,
```

```
  drop_samples = FALSE,
  n_loadings = 0,
  loadings_list = NULL,
  byPanel = FALSE,
  outlierDefX = NA,
  outlierDefY = NA,
  outlierLines = FALSE,
  label_outliers = TRUE,
  quiet = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

df	data frame in long format with Sample Id, NPX and column of choice for colors
color_g	Character value indicating which column to use for colors (default QC_Warning)
x_val	Integer indicating which principal component to plot along the x-axis (default 1)
y_val	Integer indicating which principal component to plot along the y-axis (default 2)
label_samples	Logical. If TRUE, points are replaced with SampleID (default FALSE)
drop_assays	Logical. All assays with any missing values will be dropped. Takes precedence over sample drop.
drop_samples	Logical. All samples with any missing values will be dropped.
n_loadings	Integer. Will plot the top n_loadings based on size.
loadings_list	Character vector indicating for which OlinkID's to plot as loadings. It is possible to use n_loadings and loadings_list simultaneously.
byPanel	Perform the PCA per panel (default FALSE)
outlierDefX	The number standard deviations along the PC plotted on the x-axis that defines an outlier. See also 'Details"
outlierDefY	The number standard deviations along the PC plotted on the y-axis that defines an outlier. See also 'Details"
outlierLines	Draw dashed lines at +/- outlierDefX and outlierDefY standard deviations from the mean of the plotted PCs (default FALSE)
label_outliers	Use ggrepel to label samples lying outside the limits set by the outlierLines (default TRUE)
quiet	Logical. If TRUE, the resulting plot is not printed
verbose	Logical. Whether warnings about the number of samples and/or assays dropped or imputed should be printed to the console.
...	coloroption passed to specify color order.

Details

The values are by default scaled and centered in the PCA and proteins with missing NPX values are by default removed from the corresponding assay. Unique sample names are required. Imputation by the median is done for assays with missingness <10\ The plot is printed, and a list of ggplot objects is returned.

If byPanel = TRUE, the data processing (imputation of missing values etc) and subsequent PCA is performed separately per panel. A faceted plot is printed, while the individual ggplot objects are returned.

The arguments outlierDefX and outlierDefY can be used to identify outliers in the PCA. Samples more than +/- outlierDefX and outlierDefY standard deviations from the mean of the plotted PC will be labelled. Both arguments have to be specified.

Value

A list of objects of class "ggplot", each plot contains scatter plot of PCs

Examples

```
library(dplyr)
npx_data <- npx_data1 %>%
    filter(!grepl('CONTROL', SampleID))

#PCA using all the data
olink_pca_plot(df=npx_data, color_g = "QC_Warning")

#PCA per panel
g <- olink_pca_plot(df=npx_data, color_g = "QC_Warning", byPanel = TRUE)
g[[2]] #Plot only the second panel

#Label outliers
olink_pca_plot(df=npx_data, color_g = "QC_Warning",
               outlierDefX = 2, outlierDefY = 4) #All data
olink_pca_plot(df=npx_data, color_g = "QC_Warning",
               outlierDefX = 2.5, outlierDefY = 4, byPanel = TRUE) #Per panel

#Retrieve the outliers
g <- olink_pca_plot(df=npx_data, color_g = "QC_Warning",
                    outlierDefX = 2.5, outlierDefY = 4, byPanel = TRUE)
outliers <- lapply(g, function(x){x$data}) %>%
    bind_rows() %>%
    filter(Outlier == 1)
```

`olink_plate_randomizer`

Randomly assign samples to plates

Description

Generates a scheme for how to plate samples with an option to keep subjects on the same plate and/or to keep studies together.

Usage

```
olink_plate_randomizer(
  Manifest,
  PlateSize = 96,
  Product,
  SubjectColumn,
  iterations = 500,
  available.spots,
  num_ctrl = 8,
  rand_ctrl = FALSE,
  seed,
  study = NULL
)
```

Arguments

`Manifest`	tibble/data frame in long format containing all sample ID's. Sample ID column must be named SampleID.
`PlateSize`	Integer. Either 96 or 48. 96 is default.
`Product`	String. Name of Olink product used to set PlateSize if not provided. Optional.
`SubjectColumn`	(Optional) Column name of the subject ID column. Cannot contain missing values. If provided, subjects are kept on the same plate. This argument is used for longitudinal studies and must be a separate column from the SampleID column.
`iterations`	Number of iterations for fitting subjects on the same plate.
`available.spots`	Numeric. Number of wells available on each plate. Maximum 40 for T48 and 88 for T96. Takes a vector equal to the number of plates to be used indicating the number of wells available on each plate.
`num_ctrl`	Numeric. Number of controls on each plate (default = 8)
`rand_ctrl`	Logical. Whether controls are added to be randomized across the plate (default = FALSE)
`seed`	Seed to set. Highly recommend setting this for reproducibility.
`study`	String. Optional. Name of column that includes study information. For when multiple studies are being plated and randomizing within studies. If `study` column is present in manifest, within study randomization will be performed.

Details

Variables of interest should if possible be randomized across plates to avoid confounding with potential plate effects. In the case of multiple samples per subject (e.g. in longitudinal studies), Olink recommends keeping each subject on the same plate. This can be achieved using the SubjectColumn argument.

Value

A "tibble" including SampleID, SubjectID etc. assigned to well positions. Columns include same columns as Manifest with additional columns:

- plate: Plate number
- column: Column on the plate
- row: Row on the plate
- well: Well location on the plate

See Also

- olink_displayPlateLayout() for visualizing the generated plate layouts
- olink_displayPlateDistributions() for validating that sites are properly randomized

Examples

```
#Generate randomization scheme using complete randomization
randomized.manifest_a <- olink_plate_randomizer(manifest, seed=12345)

#Generate randomization scheme that keeps subjects on the same plate (for longitudinal studies)
randomized.manifest_b <- olink_plate_randomizer(manifest,SubjectColumn="SubjectID",
                                                available.spots=c(88,88), seed=12345)

# Generate randomization scheme that keeps samples from the same study together
randomized.manifest_c <- olink_plate_randomizer(manifest, study = "Site")

#Visualize the generated plate layouts
olink_displayPlateLayout(randomized.manifest_a, fill.color = 'Site')
olink_displayPlateLayout(randomized.manifest_a, fill.color = 'SubjectID')
olink_displayPlateLayout(randomized.manifest_b, fill.color = 'Site')
olink_displayPlateLayout(randomized.manifest_b, fill.color = 'SubjectID')
olink_displayPlateLayout(randomized.manifest_c, fill.color = 'Site')

#Validate that sites are properly randomized
olink_displayPlateDistributions(randomized.manifest_a, fill.color = 'Site')
olink_displayPlateDistributions(randomized.manifest_b, fill.color = 'Site')
```

---

olink_qc_plot *Function to plot an overview of a sample cohort per Panel*

Description

Generates a facet plot per Panel using ggplot2::ggplot and ggplot2::geom_point and stats::IQR plotting IQR vs. median for all samples. Horizontal dashed lines indicate +/-IQR_outlierDef standard deviations from the mean IQR (default 3). Vertical dashed lines indicate +/-median_outlierDef standard deviations from the mean sample median (default 3).

Usage

```
olink_qc_plot(
  df,
  color_g = "QC_Warning",
  plot_index = FALSE,
  label_outliers = TRUE,
  IQR_outlierDef = 3,
  median_outlierDef = 3,
  outlierLines = TRUE,
  facetNrow = NULL,
  facetNcol = NULL,
  ...
)
```

Arguments

df	NPX data frame in long format. Must have columns SampleID, NPX and Panel
color_g	Character value indicating which column to use as fill color (default QC_Warning)
plot_index	Boolean. If FALSE (default), a point will be plotted for a sample. If TRUE, a sample's unique index number is displayed.
label_outliers	Boolean. If TRUE, an outlier sample will be labelled with its SampleID.
IQR_outlierDef	The number of standard deviations from the mean IQR that defines an outlier (default 3)
median_outlierDef	The number of standard deviations from the mean sample median that defines an outlier. (default 3)
outlierLines	Draw dashed lines at +/-IQR_outlierDef and +/-median_outlierDef standard deviations from the mean IQR and sample median respectively (default TRUE)
facetNrow	The number of rows that the panels are arranged on
facetNcol	The number of columns that the panels are arranged on
...	coloroption passed to specify color order

Value

An object of class "ggplot". Scatterplot shows IQR vs median for all samples per panel

Examples

```
library(dplyr)

olink_qc_plot(npx_data1, color_g = "QC_Warning")

#Change the outlier threshold to +-4SD
olink_qc_plot(npx_data1, color_g = "QC_Warning", IQR_outlierDef = 4, median_outlierDef = 4)

#Identify the outliers
qc <- olink_qc_plot(npx_data1, color_g = "QC_Warning", IQR_outlierDef = 4, median_outlierDef = 4)
```

```
outliers <- qc$data %>% filter(Outlier == 1)
```

`olink_ttest` *Function which performs a t-test per protein*

Description

Performs a Welch 2-sample t-test or paired t-test at confidence level 0.95 for every protein (by OlinkID) for a given grouping variable using stats::t.test and corrects for multiple testing by the Benjamini-Hochberg method ("fdr") using stats::p.adjust. Adjusted p-values are logically evaluated towards adjusted p-value<0.05. The resulting t-test table is arranged by ascending p-values.

Usage

```
olink_ttest(df, variable, pair_id, ...)
```

Arguments

`df`	NPX data frame in long format with at least protein name (Assay), OlinkID, UniProt and a factor with 2 levels.
`variable`	Character value indicating which column should be used as the grouping variable. Needs to have exactly 2 levels.
`pair_id`	Character value indicating which column indicates the paired sample identifier.
`...`	Options to be passed to t.test. See `?t.test` for more information.

Value

A "tibble" containing the t-test results for every protein. Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- estimate: "numeric" difference in mean NPX between groups
- Group 1: "numeric" Column is named first level of variable when converted to factor, contains mean NPX for that group
- Group 2: "numeric" Column is named second level of variable when converted to factor, contains mean NPX for that group
- statistic: "named numeric" value of the t-statistic
- p.value: "numeric" p-value for the test
- parameter: "named numeric" degrees of freedom for the t-statistic
- conf.low: "numeric" confidence interval for the mean (lower end)

- conf.high: "numeric" confidence interval for the mean (upper end)
- method: "character" which t-test method was used
- alternative: "character" describes the alternative hypothesis
- Adjusted_pval: "numeric" adjusted p-value for the test (Benjamini&Hochberg)
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

npx_df <- npx_data1 %>% filter(!grepl('control',SampleID, ignore.case = TRUE))

ttest_results <- olink_ttest(df=npx_df,
                             variable = 'Treatment',
                             alternative = 'two.sided')

#Paired t-test
npx_df %>%
   filter(Time %in% c("Baseline","Week.6")) %>%
   olink_ttest(variable = "Time", pair_id = "Subject")
```

---

olink_umap_plot *Function to make a UMAP plot from the data*

Description

Computes a manifold approximation and projection using umap::umap and plots the two specified components. Unique sample names are required and imputation by the median is done for assays with missingness <10\

Usage

```
olink_umap_plot(
  df,
  color_g = "QC_Warning",
  x_val = 1,
  y_val = 2,
  config = NULL,
  label_samples = FALSE,
  drop_assays = FALSE,
  drop_samples = FALSE,
  byPanel = FALSE,
  outlierDefX = NA,
  outlierDefY = NA,
  outlierLines = FALSE,
```

```
  label_outliers = TRUE,
  quiet = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

df	data frame in long format with Sample Id, NPX and column of choice for colors
color_g	Character value indicating which column to use for colors (default QC_Warning)
x_val	Integer indicating which UMAP component to plot along the x-axis (default 1)
y_val	Integer indicating which UMAP component to plot along the y-axis (default 2)
config	object of class umap.config, specifying the parameters for the UMAP algorithm (default umap::umap.defaults)
label_samples	Logical. If TRUE, points are replaced with SampleID (default FALSE)
drop_assays	Logical. All assays with any missing values will be dropped. Takes precedence over sample drop.
drop_samples	Logical. All samples with any missing values will be dropped.
byPanel	Perform the UMAP per panel (default FALSE)
outlierDefX	The number standard deviations along the UMAP dimension plotted on the x-axis that defines an outlier. See also 'Details"
outlierDefY	The number standard deviations along the UMAP dimension plotted on the y-axis that defines an outlier. See also 'Details"
outlierLines	Draw dashed lines at +/- outlierDefX and outlierDefY standard deviations from the mean of the plotted PCs (default FALSE)
label_outliers	Use ggrepel to label samples lying outside the limits set by the outlierLines (default TRUE)
quiet	Logical. If TRUE, the resulting plot is not printed
verbose	Logical. Whether warnings about the number of samples and/or assays dropped or imputed should be printed to the console.
...	coloroption passed to specify color order.

Details

The plot is printed, and a list of ggplot objects is returned.

If byPanel = TRUE, the data processing (imputation of missing values etc) and subsequent UMAP is performed separately per panel. A faceted plot is printed, while the individual ggplot objects are returned.

The arguments outlierDefX and outlierDefY can be used to identify outliers in the UMAP results. Samples more than +/- outlierDefX and outlierDefY standard deviations from the mean of the plotted UMAP component will be labelled. Both arguments have to be specified. NOTE: UMAP is a non-linear data transformation that might not accurately preserve the properties of the data. Distances in the UMAP plane should therefore be interpreted with caution.

Value

A list of objects of class "ggplot", each plot contains scatter plot of UMAPs

Examples

```
library(dplyr)
npx_data <- npx_data1 %>%
    mutate(SampleID = paste(SampleID, "_", Index, sep = ""))
try({ # Requires umap package dependency
#UMAP using all the data
olink_umap_plot(df=npx_data, color_g = "QC_Warning")

#UMAP per panel
g <- olink_umap_plot(df=npx_data, color_g = "QC_Warning", byPanel = TRUE)
g$Inflammation #Plot only the Inflammation panel

#Label outliers
olink_umap_plot(df=npx_data, color_g = "QC_Warning",
                outlierDefX = 2, outlierDefY = 4) #All data
olink_umap_plot(df=npx_data, color_g = "QC_Warning",
                outlierDefX = 3, outlierDefY = 2, byPanel = TRUE) #Per panel

#Retrieve the outliers
g <- olink_umap_plot(df=npx_data, color_g = "QC_Warning",
                     outlierDefX = 3, outlierDefY = 2, byPanel = TRUE)
outliers <- lapply(g, function(x){x$data}) %>%
    bind_rows() %>%
    filter(Outlier == 1)
})
```

---

olink_volcano_plot *Easy volcano plot with Olink theme*

Description

Generates a volcano plot using the results of the olink_ttest function using ggplot and ggplot2::geom_point. The estimated difference is plotted on the x-axis and the negative 10-log p-value on the y-axis. The horizontal dotted line indicates p-value=0.05. Dots are colored based on the Benjamini-Hochberg adjusted p-value cutoff 0.05 and can optionally be annotated by OlinkID.

Usage

```
olink_volcano_plot(p.val_tbl, x_lab = "Estimate", olinkid_list = NULL, ...)
```

Arguments

p.val_tbl	a data frame of results generated by olink_ttest()
x_lab	Optional. Character value to use as the X-axis label
olinkid_list	Optional. Character vector of proteins (by OlinkID) to label in the plot. If not provided, default is to label all significant proteins.
...	Optional. Additional arguments for olink_color_discrete()

Value

An object of class "ggplot", plotting significance (y-axis) by estimated difference between groups (x-axis) for each protein.

Examples

```
library(dplyr)

npx_df <- npx_data1 %>% filter(!grepl('control',SampleID, ignore.case = TRUE))
ttest_results <- olink_ttest(df=npx_df,
                             variable = 'Treatment',
                             alternative = 'two.sided')
olink_volcano_plot(ttest_results)
```

---

olink_wilcox *Function which performs a Mann-Whitney U Test per protein*

Description

Performs a Welch 2-sample Mann-Whitney U Test at confidence level 0.95 for every protein (by OlinkID) for a given grouping variable using stats::wilcox.test and corrects for multiple testing by the Benjamini-Hochberg method ("fdr") using stats::p.adjust. Adjusted p-values are logically evaluated towards adjusted p-value<0.05. The resulting Mann-Whitney U Test table is arranged by ascending p-values.

Usage

```
olink_wilcox(df, variable, pair_id, ...)
```

Arguments

df	NPX or Quantified_value data frame in long format with at least protein name (Assay), OlinkID, UniProt and a factor with 2 levels.
variable	Character value indicating which column should be used as the grouping variable. Needs to have exactly 2 levels.
pair_id	Character value indicating which column indicates the paired sample identifier.
...	Options to be passed to wilcox.test. See ?wilcox_test for more information.

Value

A data frame containing the Mann-Whitney U Test results for every protein.

Columns include:

- Assay: "character" Protein symbol
- OlinkID: "character" Olink specific ID
- UniProt: "character" UniProt ID
- Panel: "character" Name of Olink Panel
- estimate: "numeric" median of NPX differences between groups
- statistic: "named numeric" the value of the test statistic with a name describing it
- p.value: "numeric" p-value for the test
- conf.low: "numeric" confidence interval for the median of differences (lower end)
- conf.high: "numeric" confidence interval for the median of differences (upper end)
- method: "character" which wilcoxon method was used
- alternative: "character" describes the alternative hypothesis
- Adjusted_pval: "numeric" adjusted p-value for the test (Benjamini&Hochberg)
- Threshold: "character" if adjusted p-value is significant or not (< 0.05)

Examples

```
library(dplyr)

npx_df <- npx_data1 %>% filter(!grepl('control',SampleID, ignore.case = TRUE))

wilcox_results <- olink_wilcox(df = npx_df,
                               variable = 'Treatment',
                               alternative = 'two.sided')

#Paired Mann-Whitney U Test
npx_df %>%
   filter(Time %in% c("Baseline","Week.6")) %>%
   olink_wilcox(variable = "Time", pair_id = "Subject")
```

---

print_and_capture *Capture the output of printing an object*

Description

Capture the output of printing an object

Usage

```
print_and_capture(x)
```

Arguments

x	printable object

Value

string representation of the provided object

Examples

```
OlinkAnalyze:::print_and_capture(npx_data1)
```

---

read_flex *Read in flex data*

Description

Called by read_NPX

Usage

```
read_flex(filename)
```

Arguments

filename	where the file is located

Value

tibble of data

---

read_NPX *Function to read NPX data into long format*

Description

Imports an NPX or QUANT file exported from Olink Software. No alterations to the output format is allowed.

Usage

```
read_NPX(filename)
```

Arguments

filename	Path to Olink Software output file.

Value

A "tibble" in long format. Columns include:

- SampleID: Sample ID
- Index: Index
- OlinkID: Olink ID
- UniProt: UniProt ID
- Assay: Protein symbol
- MissingFreq: Proportion of sample below LOD
- Panel_Version: Panel Version
- PlateID: Plate ID
- QC_Warning: QC Warning Status
- LOD: Limit of detection
- NPX: Normalized Protein Expression

Additional columns may be present or missing depending on the platform

Examples

```
file <- system.file("extdata", "Example_NPX_Data.csv", package = "OlinkAnalyze")
read_NPX(file)
```

`read_npx_csv` *Helper function to read in Olink Explore csv or txt files*

Description

Helper function to read in Olink Explore csv or txt files

Usage

```
read_npx_csv(filename)
```

Arguments

`filename`	Path to Olink Software output txt of csv file.

Value

A "tibble" in long format. Some of the columns are:

- SampleID: Sample ID
- Index: Index
- OlinkID: Olink ID
- UniProt: UniProt ID
- Assay: Protein symbol
- MissingFreq: Proportion of sample below LOD
- Panel_Version: Panel Version
- PlateID: Plate ID
- QC_Warning: QC Warning Status
- LOD: Limit of detection
- NPX: Normalized Protein Expression

Additional columns may be present or missing depending on the platform

Examples

```
file <- system.file("extdata", "Example_NPX_Data.csv", package = "OlinkAnalyze")
read_NPX(file)
```

`read_npx_parquet` *Helper function to read in Olink Explore parquet output files*

Description

Helper function to read in Olink Explore parquet output files

Usage

```
read_npx_parquet(filename)
```

Arguments

`filename`	Path to Olink Software parquet output file.

Value

A "tibble" in long format. Some of the columns are:

- SampleID: Sample ID
- OlinkID: Olink ID
- UniProt: UniProt ID
- Assay: Protein symbol
- PlateID: Plate ID
- Count: Counts from sequences
- ExtNPX: External control normalized counts
- NPX: Normalized Protein Expression

Additional columns may be present or missing depending on the platform

Examples

```
file <- system.file("extdata", "Example_NPX_Data.csv", package = "OlinkAnalyze")
read_NPX(file)
```

---

read_npx_zip *Helper function to read in Olink Explore zip csv files*

Description

Helper function to read in Olink Explore zip csv files

Usage

```
read_npx_zip(filename)
```

Arguments

filename	Path to Olink Software output zip file.

Value

A "tibble" in long format. Some of the columns are:

- SampleID: Sample ID
- Index: Index
- OlinkID: Olink ID
- UniProt: UniProt ID
- Assay: Protein symbol

- MissingFreq: Proportion of sample below LOD
- Panel_Version: Panel Version
- PlateID: Plate ID
- QC_Warning: QC Warning Status
- LOD: Limit of detection
- NPX: Normalized Protein Expression

Additional columns may be present or missing depending on the platform

Examples

```
try({ # May fail if dependencies are not installed
file <- system.file("extdata", "Example_NPX_Data.csv", package = "OlinkAnalyze")
read_NPX(file)
})
```

---

set_plot_theme *Function to set plot theme*

Description

This function sets a coherent plot theme for functions.

Usage

```
set_plot_theme(font = "Swedish Gothic Thin")
```

Arguments

font	Font family to use for text elements. Depends on extrafont package.

Value

No return value, used as theme for ggplots

Examples

```
library(ggplot2)

ggplot(mtcars, aes(x = wt, y = mpg, color = as.factor(cyl))) +
  geom_point(size = 4) +
  set_plot_theme()

ggplot(mtcars, aes(x = wt, y = mpg, color = as.factor(cyl))) +
  geom_point(size = 4) +
```

```
set_plot_theme(font = "")
```

Index