Plasma Proteomics with HDAnalyzeR: : **CHEAT SHEET**

Basics

With **HDAnalyzeR** you can perform complex plasma proteomics analysis with simple steps. Starting from Olink data and metadata via simple functions to biomarker discovery!

DAid	Assay	NPX
1	Prot1	0.2
1	Prot2	-0.5

+

DAid	Disease	Sex
1	Cancer	F
2	Healthy	M

=

Remember that the Olink data should contain the columns:
DAid, Assay, and NPX

The metadata should contain the columns:
DAid, Disease (column with case-control groups), and Sex (M or F)

Utilities

create_dir(dir_name, date = FALSE)

Creates a directory with a specified name. The user can choose to create another inner directory with the current date as its name.

create_dir("my_directory", date = FALSE)

save_df(df, file_name, dir_name, date = FALSE, file_type = c("csv", "tsv", "rda"))

Saves a dataset in the specified format (CSV, TSV, or RDA) in a specified directory. The recommended file type is RDA.

save_df(example_metadata, "metadata", "my_data", file_type = "rda")

import_df(file_path)

Imports a dataset from a file. The file format can be CSV, TSV, TXT, RDA, RDS, XLSX, or Parquet format. It returns the dataset as tibble.

import_df("my_data/metadata.rda")

widen_data(olink_data)

Transforms Olink data from long to wide format.

widen_data(example_data)

DAid	Assay	NPX
1	Prot1	0.2
1	Prot2	-0.5
2	Prot1	0.3
2	Prot2	-0.2

=

DAid	Prot1	Prot2
1	0.2	-0.5
2	0.3	-0.2

* To improve performance, it is recommended to use tidied data throughout the analysis. Initially, widen the data in your script.

Preprocessing Data

PREPROCESSING

clean_data(df_in, keep_cols = c("DAid", "Assay", "NPX"), cohort = NULL, filter_plates = NULL, filter_assays = NULL, filter_assay_warning = FALSE, remove_na_cols = c("DAid", "NPX"), replace_w_na = c(0, "0", "", "Unknown", "unknown", "none", NA, "na"))

clean_metadata(df_in, keep_cols = c("DAid", "Assay", "Sex", "Age", "BMI"), remove_na_cols = c("DAid", "Disease"), replace_w_na = c(0, "0", "", "Unknown", "unknown", "none", NA, "na"))

Select columns and filter rows based on the specified criteria.

clean_data(example_data, filter_plates = c("Plate1", "Plate2"))

generate_df(long_data, metadata = NULL, join = TRUE, metadata_cols = c("DAid", "Disease", "Sex", "Age", "BMI"), save = TRUE)

Creates wide and join dataframes from long Olink data and metadata.

generate_df(example_data, example_metadata, save = FALSE)

DATA NORMALIZATION & IMPUTATION

normalize_data(olink_data, metadata = NULL, wide = TRUE, center = TRUE, scale = TRUE, batch = NULL, batch2 = NULL, return_long = FALSE, save = FALSE, file_name = "normalized_data")

normalizes the data by scaling them and removing their batch effects.

normalize_data(example_data, example_metadata, wide = FALSE)

impute_median(olink_data, wide = TRUE, exclude_cols = c("DAid", "Disease"), show_na_percentage = TRUE)

impute_knn(olink_data, wide = TRUE, k = 5, exclude_cols = c("DAid", "Disease"), show_na_percentage = TRUE)

impute_missForest(olink_data, wide = TRUE, maxiter = 10, ntree = 100, parallelize = "variables", ncores = 4, exclude_cols = c("DAid", "Disease"), show_na_percentage = TRUE)

impute_mice(olink_data, wide = TRUE, m = 5, maxit = 5, method = "pmm",exclude_cols = c("DAid", "Disease"), show_na_percentage = TRUE)

Impute missing values in a dataset using different techniques.

impute_knn(example_data, wide = FALSE, k = 3)

DAid	Prot1	Prot2
1	NA	-0.5
2	0.3	NA

=

DAid	Prot1	Prot2
1	0.2	-0.5
2	0.3	-0.2

Data Quality Control

QUALITY CONTROL

qc_summary_data(df, wide = TRUE, threshold = 0.8, report = TRUE)

qc_summary_metadata(metadata, categorical = "Sex", numeric = "Age", disease_palette = NULL, categ_palette = "sex_hpa", report = T)

Summarize the quality control results of data and metadata. Plot distributions of metadata variables.

qc_summary_data(example_data, wide = FALSE, threshold = 0.7)

CORRELATION & CLUSTERING

create_corr_heatmap(x, y = NULL, use = "pairwise.complete.obs", method = "pearson", threshold = 0.8, cluster_rows = TRUE, cluster_cols = TRUE)

Calculates correlation matrix and creates heatmap. Returns list of protein pairs that correlate above the defined threshold.

create_corr_heatmap(wide_data, threshold = 0.7)

cluster_data(df, distance_method = "euclidean", clustering_method = "ward.D2", cluster_rows = TRUE, cluster_cols = TRUE, wide = TRUE)

Takes a dataset and returns the same dataset ordered according to the hierarchical clustering of the rows and columns

cluster_data(example_data, wide = FALSE)

DAid	Prot1	Prot2	Prot3
1	0.2	-0.5	1.2
2	0.3	-0.2	1.3
3	0.2	-0.4	1.1

=

DAid	Prot1	Prot3	Prot2
2	0.3	1.3	-0.2
1	0.2	1.2	-0.5
3	0.2	1.1	-0.4

DIMENSIONALITY REDUCTION

do_pca(olink_data, metadata = NULL, pcs = 5, color = "Disease", palette = NULL, wide = TRUE, assay = FALSE, impute = TRUE, plots = TRUE, x = "PC1", y = "PC2", npcs = 4, nproteins = 8, loadings = FALSE, save = FALSE)

do_umap(olink_data, metadata = NULL, color = "Disease", palette = NULL, wide = TRUE, assay = FALSE, impute = TRUE, plots = TRUE, save = FALSE)

Run a PCA or UMAP analysis on the the data. Visualize the data points on 2D planes.

do_pca(example_data, example_metadata, wide = FALSE, color = "Disease", palette = "cancers12")

THE HUMAN PROTEIN ATLAS

 Konstantinos Antonopoulos • konstantinos.antonopoulos@scilifelab.se • Learn more at hda1472.github.io/DA_RPackage • HDAnalyzeR 0.1.0 • Updated: 2024-09

Main Analysis

DIFFERENTIAL EXPRESSION

do_limma(olink_data, metadata, variable = "Disease", case, control, correct = c("Sex", "Age"), correct_type = c("factor", "numeric"), wide = TRUE, only_female = NULL, only_male = NULL, volcano = TRUE, pval_lim = 0.05, logfc_lim = 0, top_up_prot = 40, top_down_prot = 10, palette = "diff_exp", report_nproteins = TRUE, subtitle = NULL, save = FALSE)

Perform differential expression analysis. Ability to correct for cofactors.

do_limma(example_data, example_metadata, case = "AML", control = c("CLL", "MYEL"), wide = FALSE)

do_ttest(olink_data, metadata, variable = "Disease", case, control, wide = TRUE, only_female = NULL, only_male = NULL, volcano = TRUE, pval_lim = 0.05, logfc_lim = 0, top_up_prot = 40, top_down_prot = 10, palette = "diff_exp", report_nproteins = TRUE, subtitle = NULL, save = FALSE)

Perform differential expression analysis by comparing the groups with t-test.

do_ttest(example_data, example_metadata, case = "AML", control = c("CLL", "MYEL"), wide = FALSE)

do_limma_continuous(olink_data, metadata, variable, correct = c("Sex"), correct_type = c("factor"), wide = TRUE, volcano = TRUE, pval_lim = 0.05, logfc_lim = 0, top_up_prot = 40, top_down_prot = 10, palette = "diff_exp", report_nproteins = TRUE, subtitle = NULL, save = FALSE)

Perform differential expression analysis against continuous variable.

do_limma_continuous(example_data, example_metadata, "Age", wide = FALSE)

Assay	logFC	pval	Av. Expr
Prot1	0.01	0.01	1.2
Prot2	-0.2	0.02	1.3
Prot3	0.04	0.06	1.1

VISUALIZE RESULTS

Summarize the results from differential expression and classification models. Plot boxplots and correlation plots with regression on top.

plot_de_summary(de_results, disease_palette = NULL, diff_exp_palette = "diff_exp")

plot_features_summary(ml_results, importance = 50, upset_top_features = FALSE, case_palette = NULL, feature_type_palette = c(`all-features` = "pink", `top-features` = "darkblue"))

plot_protein_boxplot(join_data, variable = "Disease", proteins, case, points = TRUE, xaxis_names = FALSE, palette = NULL)

plot_scatter_wth_regression(plot_data, x, y, se = FALSE, line_color = "black", r_2 = TRUE)

MACHINE LEARNING CLASSIFICATION MODELS

do_rreg(olink_data, metadata, variable = "Disease", case, control, wide = TRUE, strata = TRUE, balance_groups = TRUE, only_female = NULL, only_male = NULL, exclude_cols = "Sex", ratio = 0.75, type = "lasso", cor_threshold = 0.9, cv_sets = 5, grid_size = 10, ncores = 4, hypopt_vis = TRUE, palette = NULL, vline = TRUE, subtitle = c("accuracy", "sensitivity", "specificity", "auc", "features", "top-features", "mixture"), varimp_yaxis_names = FALSE, nfeatures = 9, points = TRUE, boxplot_xaxis_names = FALSE, seed = 123)

do_rf(olink_data, metadata, variable = "Disease", case, control, wide = TRUE, strata = TRUE, balance_groups = TRUE, only_female = NULL, only_male = NULL, exclude_cols = "Sex", ratio = 0.75, cor_threshold = 0.9, normalize = TRUE, cv_sets = 5, grid_size = 10, ncores = 4, hypopt_vis = TRUE, palette = NULL, vline = TRUE, subtitle = c("accuracy", "sensitivity", "specificity", "auc", "features", "top-features"), varimp_yaxis_names = FALSE, nfeatures = 9, points = TRUE, boxplot_xaxis_names = FALSE, seed = 123)

Perform binary classification with regularized regression (elastic net, lasso, ridge) and random forest models. Ability to optimize model hyperparameters.

do_rf(example_data, example_metadata, case = "AML", control = c("CLL", "MYEL"), wide = FALSE, palette = "cancers12", cv_sets = 5, grid_size = 10)

do_rreg_multi(olink_data, metadata, variable = "Disease", wide = TRUE, strata = TRUE, exclude_cols = "Sex", ratio = 0.75, type = "lasso", cor_threshold = 0.9, cv_sets = 5, grid_size = 10, ncores = 4, hypopt_vis = TRUE, palette = NULL, seed = 123)

do_rf_multi(olink_data, metadata, variable = "Disease", wide = TRUE, strata = TRUE, exclude_cols = "Sex", ratio = 0.75, cor_threshold = 0.9, normalize = TRUE, cv_sets = 5, grid_size = 10, ncores = 4, hypopt_vis = TRUE, palette = NULL, seed = 123)

Perform multi classification with regularized regression (elastic net, lasso, ridge) and random forest models. Ability to optimize model hyperparameters.

do_rf_multi(example_data, example_metadata, wide = FALSE, palette = "cancers12", cv_sets = 5, grid_size = 5)

do_lreg(olink_data, metadata, variable = "Disease", case, control, wide = TRUE, strata = TRUE, balance_groups = TRUE, only_female = NULL, only_male = NULL, exclude_cols = "Sex", ratio = 0.75, cor_threshold = 0.9, normalize = TRUE, cv_sets = 5, ncores = 4, palette = NULL, points = TRUE, boxplot_xaxis_names = FALSE, seed = 123)

Perform binary classification with logistic regression model. Ideal for data with single predictor. If data contain multiple predictors prefer to use do_rreg

do_lreg(single_predictor_data, example_metadata, case = "AML", control = "CLL", wide = FALSE, ncores = 1, palette = "cancers12")

Post Analysis

LITERATURE SEARCH

literature_search(prot_dis_list, max_articles = 10)

searches for articles for protein-disease pairs in PubMed.

prot_dis_list <- list("acute myeloid leukemia" = c("FLT3", "EPO"))

lit_search_results <- literature_search(prot_dis_list, max_articles = 1))

ENRICHMENT ANALYSIS

do_ora(protein_list, database = c("GO", "Reactome"), background = NULL, pval_lim = 0.05)

plot_ora(enrichment, protein_list, pval_lim = 0.05, ncateg = 10, fontsize = 10)

Performs over-representation analysis (ORA) using the clusterProfiler package and plot results.

control = c("BRC", "CLL", "CRC", "CVX", "ENDC", "GLIOM", "LUNGC", "LYMPH", "MYEL", "OVC", "PRC")
de_res <- do_limma(example_data, example_metadata, case = "AML", control = control, wide = FALSE)

sig_up_proteins_aml <- de_res$de_results |>
 dplyr::filter(sig == "significant up") |>
 dplyr::pull(Assay)

do_ora(sig_up_proteins_aml, database = "GO")

Enrichment <- plot_ora(enrichment, protein_list, pval_lim = 0.05, ncateg = 10, fontsize = 10)

do_gsea(de_results, database = c("GO", "Reactome"), pval_lim = 0.05)

plot_gsea(enrichment, de_results, pval_lim = 0.05, ncateg = 10, fontsize = 10)

Performs Gene Set Enrichment Analysis (GSEA) using the clusterProfiler package and plot results.

control = c("BRC", "CLL", "CRC", "CVX", "ENDC", "GLIOM", "LUNGC", "LYMPH", "MYEL", "OVC", "PRC")
de_res <- do_limma(example_data, example_metadata, case = "AML", control = control, wide = FALSE)

de_results <- de_res$de_results
Enrichment <- do_gsea(de_results, database = "GO", pval_lim = 0.9)

plot_gsea(enrichment, de_results, pval_lim = 0.9, ncateg = 7, fontsize = 7)

THE HUMAN PROTEIN ATLAS