Biomarker Discovery with HDAnalyzeR: : **CHEAT SHEET**

Basics

HDAnalyzeR simplifies proteomics data analysis for biomarker discovery. Starting from your omics data, you can perform complex analysis with simple steps and minimal code.

ID	Gene	Exp
1	Prot1	0.2
1	Prot2	-0.5

+

ID	Disease	Sex
1	Cancer	F
2	Healthy	M

=

Utilities

hd_initialize(dat, metadata = NULL, is_wide = FALSE, sample_id = "DAid", var_name = "Assay", value_name = "NPX")

Initializes an HDAnalyzeR object with the data, metadata, and other parameters and can be used as input to various analyses in the package.

hd_object <- hd_initialize(example_data, example_metadata)

hd_import_data(path_name)

imports data from a file (CSV, TSV, TXT, RDA, RDS, XLSX, or Parquet). It reads it and returns it as a tibble or an R object.

hd_import_data("my_data/metadata.rds")

hd_save_data(dat, path_name)

Saves a dataset or an R object in the specified format (CSV, TSV, RDS, or XLSX) in a specified directory. The recommended file type is RDS.

hd_save_data(dat, path_name)

hd_save_path(path_name, date = FALSE)

Creates a directory with in a specified path. The user can optionally choose to create another inner directory named with the current date

hd_save_path(path_name, date = FALSE)

hd_widen_data(dat, exclude = "DAid", names_from = "Assay", values_from = "NPX")

hd_long_data(dat, exclude = "DAid", names_to = "Assay", values_to = "NPX")

Transforms omics data from long to wide format or vice versa.

hd_widen_data(example_data)

Long Format

ID	Gene	Exp
1	Prot1	0.2
1	Prot2	-0.5
2	Prot1	0.3
2	Prot2	-0.2

=

Wide Format

ID	Prot1	Prot2
1	0.2	-0.5
2	0.3	-0.2

Preprocessing Data

PREPROCESSING

hd_bin_columns(dat, column_types, bins = 5, round_digits = 0)

Bins continuous variables and labels them with ranges.

hd_bin_columns(example_metadata["Age"], "continuous", 5)

hd_detect_vartype(var, unique_threshold = 5)

Detects the type of a variable based on its content.

hd_detect_vartype(example_metadata[["Age"]]

hd_log_transform(dat)

Log transforms omics data in wide format.

hd_log_transform(hd_object)

DATA NORMALIZATION & IMPUTATION

hd_normalize(dat, metadata = NULL, center = TRUE, scale = TRUE, batch = NULL, batch2 = NULL)

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

hd_normalize(hd_object, center = TRUE, scale = FALSE, batch = "Cohort")

hd_omit_na(dat, columns = NULL)

Removes rows with missing values from a dataset.

hd_omit_na(hd_object)

hd_impute_median(dat, verbose = TRUE)

hd_impute_knn(dat, k = 5, verbose = TRUE)

hd_impute_missForest(dat, maxiter = 10, ntree = 100, parallelize = "no", verbose = TRUE)

Impute missing values in a dataset using different techniques.

hd_impute_knn(hd_object, k = 3)

ID	Prot1	Prot2
1	NA	-0.5
2	0.3	NA

=

ID	Prot1	Prot2
1	0.2	-0.5
2	0.3	-0.2

Data Quality Control

QUALITY CONTROL

hd_qc_summary(dat, metadata = NULL, variable, palette = NULL, unique_threshold = 5, cor_threshold = 0.8, cor_method = "pearson", verbose = TRUE)

Summarizes the quality control results of the input data and metadata.

hd_qc_summary(hd_object, variable = "Disease", cor_threshold = 0.7)

CORRELATION & CLUSTERING

hd_correlate(x, y = NULL, use = "pairwise.complete.obs", method = "pearson")

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

Calculates the correlation matrix of the input dataset and plots heatmap.

hd_correlate(matrix)

hd_cluster(dat, distance_method = "euclidean", clustering_method = "ward.D2", cluster_rows = TRUE, cluster_cols = TRUE, normalize = 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

hd_auto_pca(dat, metadata = NULL, components = 10, by_sample = TRUE, plot_x = "PC1", plot_y = "PC2", plot_color = NULL, plot_palette = NULL)

hd_auto_umap(dat, metadata = NULL, by_sample = TRUE, plot_x = "UMAP1", plot_y = "UMAP2", plot_color = NULL, plot_palette = NULL)

Run a PCA or UMAP analysis on the data and visualize the results.

hd_auto_pca(hd_object, components = 20, plot_color = "Disease")

THE HUMAN PROTEIN ATLAS

Main Analysis

DIFFERENTIAL EXPRESSION

hd_de_limma(dat, metadata = NULL, variable = "Disease", case, control = NULL, correct = NULL, log_transform = FALSE)

hd_de_ttest(dat, metadata = NULL, variable = "Disease", case, control = NULL, log_transform = FALSE)

Perform differential expression analysis. Ability to correct for cofactors in **hd_de_limma**.

de_results <- hd_de_limma(hd_object, case = "AML", correct = c("Sex"))

hd_plot_volcano(de_object, pval_lim = 0.05, logfc_lim = 0, top_up_prot = 10, top_down_prot = 5, palette = "diff_exp", title = NULL, report_nproteins = TRUE, user_defined_proteins = NULL)

Creates volcano plots for the differential expression results.

hd_plot_volcano(de_results)

Gene	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

MACHINE LEARNING CLASSIFICATION MODELS

hd_split_data(dat, metadata = NULL, variable = "Disease", metadata_cols = NULL, ratio = 0.75, seed = 123)

Splits the data into training and test sets based on the ratio provided.

hd_split <- hd_split_data(hd_object, variable = "Disease")

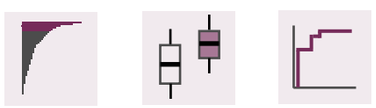
hd_model_lr(dat, variable = "Disease", case, control = NULL, balance_groups = TRUE, cor_threshold = 0.9, palette = NULL, plot_y_labels = TRUE, verbose = TRUE, plot_title = c("accuracy", "sensitivity", "specificity", "auc", "features", "top-features"), seed = 123)

hd_model_rreg(dat, variable = "Disease", case, control = NULL, balance_groups = TRUE, cor_threshold = 0.9, grid_size = 30, cv_sets = 5, mixture = NULL, palette = NULL, plot_y_labels = FALSE, verbose = TRUE, plot_title = NULL, seed = 123)

hd_model_rf(dat, variable = "Disease", case, control = NULL, balance_groups = TRUE, cor_threshold = 0.9, grid_size = 30, cv_sets = 5, palette = NULL, plot_y_labels = FALSE, verbose = TRUE, plot_title = NULL, seed = 123)

Perform classification/multi-classification/regression with logistic regression, regularized regression and random forest models and visualize results. Ability to optimize model hyperparameters.

hd_model_rreg(hd_split, variable = "Disease", case = "AML")



WEIGHTED GENE CO-EXPRESSION NETWORK ANALYSIS

hd_wgcna(dat, power = NULL)

Performs WGCNA on the provided data. The user can specify the power parameter for the analysis or the function will select an optimal power value based on the data.

hd_wgcna(hd_object)

hd_plot_wgcna(dat, metadata = NULL, wgcna, clinical_vars = NULL)

Visualizes the WGCNA results.

hd_plot_wgcna(hd_object, wgcna = wgcna_res, clinical_vars = c("Disease", "Sex", "Age", "BMI"))

Post Analysis

PUBMED LITERATURE SEARCH

hd_literature_search(feature_class_list, max_articles = 10, keywords = NULL, fields = "All Fields", api_key = NULL, verbose = TRUE)

Searches articles for gene-disease pairs in PubMed.

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

lit_search_results <- literature_search(feature_class_list)

PATHWAY ENRICHMENT ANALYSIS

hd_ora(gene_list, database = c("GO", "Reactome", "KEGG"), ontology = c("BP", "CC", "MF", "ALL"), background = NULL, pval_lim = 0.05)

hd_plot_ora(enrichment, seed = 123)

Performs over-representation analysis (ORA) and plot results.

enrich <- hd_ora(gene_list, database = "GO", ontology = "BP")

hd_plot_ora(enrichment)

hd_gsea(de_results, database = c("GO", "Reactome", "KEGG"), ontology = c("BP", "CC", "MF", "ALL"), ranked_by = "logFC", pval_lim = 0.05)

hd_plot_gsea(enrichment, seed = 123)

Performs Gene Set Enrichment Analysis (GSEA) and plot results.

enrich <- hd_gsea(de_results, database = "GO", ontology = "BP")

hd_plot_gsea(enrichment)

For more information and detailed examples check HDAnalyzeR's webpage.

VISUALIZATION

RESULT SUMMARIES

hd_plot_de_summary(de_results, variable = "Disease", class_palette = NULL, diff_exp_palette = "diff_exp", pval_lim = 0.05, logfc_lim = 0)

Creates summary visualizations of the results from multiple differential expression analyses.

res <- list("AML" = de_results_aml, "LUNGC" = de_results_lungc)

hd_plot_de_summary(res, class_palette = "cancers12")

hd_plot_model_summary(model_results, importance = 0.5, class_palette = NULL, upset_top_features = FALSE)

Creates summary visualizations of the results from multiple classification models.

res <- list("AML" = model_results_aml, "LUNGC" = model_results_lungc)

hd_plot_model_summary(res, class_palette = "cancers12")

hd_plot_de_summary(de_results, model_results, order_by, pval_lim = 0.05, logfc_lim = 0)

Creates summary visualizations of the results from multiple differential expression analyses

res_de <- list("AML" = model_results_aml, "LUNGC" = model_results_lungc)
res_model <- list("AML" = de_results_aml, "LUNGC" = de_results_lungc)

hd_plot_feature_heatmap(res_de, res_model, order_by = "AML")

OTHER VISUALIZATIONS

hd_plot_regression(dat, metadata = NULL, metadata_cols = NULL, x, y, se = FALSE, line_color = "#883268", r_2 = TRUE)

hd_plot_feature_boxplot(dat, metadata = NULL, variable = "Disease", features, case, type = "case_vs_all", points = TRUE, x_labels = TRUE, yaxis_title = "NPX", palette = NULL)

hd_plot_feature_network(feature_panel, plot_color = "Scaled_Importance", class_palette = NULL, importance_palette = NULL, seed = 123)

Visualize the biomarkers in different ways, e.g. correlation between them and clinical variables, boxplots, disease-marker network.

hd_plot_feature_boxplot(hd_object, variable = "Disease", features = c("FLT3", "EPO"), case = "AML", palette = "cancers12")

THE HUMAN PROTEIN ATLAS