alphaviz

Mann Labs, MPIB

Aug 02, 2022

CONTENTS

AlphaViz is an automated visualization pipeline to link the identifications found in the analysis with the original raw MS data and easily assess their individual quality or the overall quality of whole samples. AlphaViz is freely available, open-source and available on all major Operating Systems. It can be used with a graphical user interface (GUI) or as a regular Python package.

This documentation is intended as an API for direct Python use. For more information, see AlphaViz on GitHub.

CHAPTER

ONE

ALPHAVIZ.GUI

CHAPTER

TWO

ALPHAVIZ.IO

This module provides functions to read MQ/DiaNN/AlphaPept output files and other IO supplementary functions.

Functions:

create_ap_peptides_table(ap_df)	
create_ap_proteins_table(ap_df, fasta)	
create_diann_peptides_table(diann_df)	Extract information about peptides from the loaded main DIANN output .tsv file.
create_diann_proteins_table(diann_df, fasta)	Extract information about genes, proteins and protein groups from the loaded main DIANN output .tsv file.
get_filenames_from_directory(directory, ...)	Search for files with the specified extension in the repository and return a list of all file names with that extention.
import_alphapept_output(...)	
import_diann_output(...)	Load two files from the DiaNN output folder and returns the data frames containing information about proteins, peptides, and summary information about the whole experiment.
import_diann_stats(filepath, experiment)	Read the DIANN output .stats.tsv file.
import_mq_all_peptides(filepath)	Read some columns from the output file allPeptides.txt of MaxQuant software.
import_mq_evidence(filepath, experiment)	Read some columns from the output file evidence.txt of MaxQuant software.
import_mq_msms(filepath, experiment)	Read some columns from the output file msms.txt of MaxQuant software.
import_mq_output(necessary_files, ...)	Read all specified files from the MQ output folder and returns the data frames for each of the files.
import_mq_protein_groups(filepath, experiment)	Read the output file proteinGroups.txt of MaxQuant software.
import_mq_summary(filepath)	Read the output file summary.txt of MaxQuant software.
read_fasta(filepath)	Read the fasta file using the pyteomics package.
read_file(filepath, column_names)	Enable reading the file and retrieving the values from the specified columns.

alphaviz.io.**create_ap_peptides_table**(*ap_df: DataFrame*)

alphaviz.io.**create_ap_proteins_table**(*ap_df: DataFrame*, *fasta: object*)

alphaviz.io.**create_diann_peptides_table**(*diann_df: DataFrame*)

Extract information about peptides from the loaded main DIANN output .tsv file.

Parameters
diann_df (*pd.DataFrame*) – The original data frame after loading the main .tsv DIANN output file and filter by the experiment name.

Returns
The output data frame contains information about peptides.

Return type
pd.DataFrame

alphaviz.io.**create_diann_proteins_table**(*diann_df: DataFrame*, *fasta: object*)

Extract information about genes, proteins and protein groups from the loaded main DIANN output .tsv file.

Parameters

- **diann_df** (*pd.DataFrame*) – The original data frame after loading the main .tsv DIANN output file and filter by the experiment name.
- **fasta** (*pyteomics.fasta.IndexedUniProt*) – The object containing information about all proteins from the fasta file.

Returns
The output data frame contains information about genes, proteins and proteins groups.

Return type
pd.DataFrame

alphaviz.io.**get_filenames_from_directory**(*directory: str*, *extensions_list: list*) → list

Search for files with the specified extension in the repository and return a list of all file names with that extention.

Parameters

- **directory** (*str*) – Path to the repository to search in.
- **extensions_list** (*list*) – A list of extensions, e.g. ['d', 'hdf'].

Returns
The list of filtered file names based on their extensions.

Return type
list

alphaviz.io.**import_alphapept_output**(*path_ap_output_folder: str*, *experiment: str*, *fasta: object*)

alphaviz.io.**import_diann_output**(*path_diann_output_folder: str*, *experiment: str*, *fasta: object*)

Load two files from the DiaNN output folder and returns the data frames containing information about proteins, peptides, and summary information about the whole experiment.

Parameters

- **path_diann_output_folder** (*str*) – Path to the DIANN output folder with all output files needed.
- **experiment** (*str*) – The name of the experiment.
- **fasta** (*pyteomics.fasta.IndexedUniProt*) – The object containing information about all proteins from the fasta file.

Returns
The function returns three pandas data frame with the extracted information about proteins, peptides, and summary information about the whole experiment.

Return type
list of pd.DataFrames

alphaviz.io.**import_diann_stats**(*filepath: str*, *experiment: str*)

Read the DIANN output .stats.tsv file.

Parameters

- **filepath** (*str*) – Full path to the .stats.tsv file.
- **experiment** (*str*) – The name of the experiment.

Returns
The output data frame contains summary information about the whole experiment.

Return type
pd.DataFrame

alphaviz.io.**import_mq_all_peptides**(*filepath: str*) → DataFrame

Read some columns from the output file allPeptides.txt of MaxQuant software.

Parameters
filepath (*str*) – Full path to the allPeptides.txt file.

Returns

The output data frame contains information about the following MQ columns:

- 'Pasef MS/MS IDs' ('list' type),
- 'MS/MS scan number' ('int' type).

The rows of the data frame with missing 'MS/MS scan number' values are dropped.

Return type
pd.DataFrame

alphaviz.io.**import_mq_evidence**(*filepath: str*, *experiment: str*) → DataFrame

Read some columns from the output file evidence.txt of MaxQuant software.

Parameters

- **filepath** (*str*) – Full path to the evidence.txt file.
- **experiment** (*str*) – The name of the experiment.

Returns

The output data frame contains information about the following MQ columns:

- 'Sequence',
- 'Length' ('int' type),
- 'Acetyl (Protein N-term)' (renamed to 'Acetylation (N-term)') ('int' type),
- 'Oxidation (M)' ('int' type),
- 'Proteins',
- 'Retention time' ('float:.4d' type),
- 'Mass' ('float:.4d' type),
- 'm/z' ('float:.4d' type),
- 'Charge' ('category' type),

- 'Intensity' ('int' type),
- '1/K0' ('float:.4d' type),
- 'MS/MS count' ('category' type),
- 'MS/MS scan number' ('int' type),
- 'Gene names' ('category' type),
- 'Score' (renamed to 'Andromeda score') ('int' type),
- 'Raw file' ('category' type),
- 'Uncalibrated mass error [ppm]' ('float:.4d' type),
- 'Mass error [ppm]' ('float:.4d' type),
- 'Modified sequence'.

Renamed columns are marked as is the output data type of all columns. The rows of the data frame with missing 'MS/MS scan number' values are dropped.

Return type
pd.DataFrame

alphaviz.io.**import_mq_msms**(*filepath: str*, *experiment: str*) → DataFrame

Read some columns from the output file msms.txt of MaxQuant software.

Parameters
filepath (*str*) – Full path to the msms.txt file.

Returns

The output data frame contains information about the following MQ columns:

- 'Scan number' ('int' type),
- 'Matches',
- 'Masses',
- 'Mass deviations [Da]',
- 'Mass deviations [ppm]'.

Return type
pd.DataFrame

alphaviz.io.**import_mq_output**(*necessary_files: list*, *path_mq_output_folder: str*, *experiment: str*)

Read all specified files from the MQ output folder and returns the data frames for each of the files.

Parameters

- **necessary_files** (*list*) – A list of strings containing the names of the MQ output files with extensions, e.g. ['allPeptides.txt', 'msms.txt'].
- **path_mq_output_folder** (*str*) – Path to the MaxQuant output folder with all output files needed.
- **experiment** (*str*) – The name of the experiment.

Returns
For each of the specified MQ output files, the function returns a pandas data frame with the extracted information.

Return type
generator

alphaviz.io.**import_mq_protein_groups**(*filepath: str*, *experiment: str*) → DataFrame

Read the output file proteinGroups.txt of MaxQuant software.

Parameters

- **filepath** (*str*) – Full path to the proteinGroups.txt file.
- **experiment** (*str*) – The name of the experiment.

Returns

- *# pd.DataFrame*
- *# The output data frame contains information about the following MQ columns*
- *# - 'Protein IDs',*
- *# - 'Protein names',*
- *# - 'Gene names',*
- *# - 'Number of proteins' (renamed to '# proteins'),*
- *# - 'Mol. weight [kDa]' (renamed to 'Mol weight, kDa'),*
- *# - f'Peptides Exp_{experiment}' (renamed to '(EXP) # peptides'),*
- *# - f'Unique peptides Exp_{experiment}' (renamed to '(EXP) # unique peptides'),*
- *# - f'Sequence coverage Exp_{experiment} [%]' (renamed to '(EXP) Seq coverage, %'),*
- *# - 'MS/MS count' (renamed to '# MS/MS'),*
- *# - 'Sequence lengths',*
- *# Renamed columns are marked. The rows of the data frame with missing 'Gene names' values are dropped.*

alphaviz.io.**import_mq_summary**(*filepath: str*) → DataFrame

Read the output file summary.txt of MaxQuant software.

Parameters
filepath (*str*) – Full path to the msms.txt file.

Returns
The output data frame contains summary information of all the experiments.

Return type
pd.DataFrame

alphaviz.io.**read_fasta**(*filepath: str*) → dict

Read the fasta file using the pyteomics package.

Parameters
filepath (*str*) – Full path to the .fasta file.

Returns
The output object allows access to all available information in the fasta file using the protein ID.

Return type
pyteomics.fasta.IndexedUniProt object

alphaviz.io.**read_file**(*filepath: str*, *column_names: list*) → DataFrame

Enable reading the file and retrieving the values from the specified columns. Compared to function pd.read_csv() it gains significant time if the file is huge and is only a few ms slower for small files.

Parameters

- **filepath** (*str*) – Full path to the file.
- **column_names** (*list*) – A list of column names to be read.

Returns
This data frame contains data from all columns of the specified file.

Return type
pd.DataFrame

CHAPTER

THREE

ALPHAVIZ.PLOTTING

CHAPTER

FOUR

ALPHAVIZ.PREPROCESSING

This module provides functions that are helping to preprocess the data.

Functions:

convert_diann_ap_mod(sequence)	Convert DIA-NN style modifications to AlphaPept style modifications.
convert_diann_mq_mod(sequence)	Convert DIA-NN style modifications to MaxQuant style modifications.
filter_df(df, pattern, column, software)	Filter the data frame based on the pattern (any value) in the specified column.
get_aa_seq(protein_id, fasta)	Extract the leading razor protein sequence for the list of proteinIDs of the protein group from the pyteomics.fasta.IndexedUniProt object.
get_identified_ions(values, sequence, ion_type)	For the specified peptide sequence extract all identified in the experiment ions and based on the specified ion_type return a list of booleans containing information for the b-ions whether the peptide is breaking after aligned amino acid or for the y-ion whether is breaking before aligned amino acid.
get_mq_ms2_scan_data(msms, ...)	Extract MS2 data as a data frame for the specified MSMS scan number and precursor ID from the 'msms.txt' MQ output file and raw file.
get_mq_unique_proteins(filepath)	Extract unique "Protein names" from the specified MaxQuant output file.
get_protein_info(fasta, protein_ids)	Get the name and the length of the protein(s) from the fasta file specifying the protein id(s).
get_protein_info_from_fastaheader(string, ...)	Extract information about protein IDs, protein names and gene names from the "Fasta headers" column of the MQ output tables.
sort_naturally(line[, reverse])	Sort the string natural to humans, e.g.

alphaviz.preprocessing.**convert_diann_ap_mod**(*sequence: str*) → str

Convert DIA-NN style modifications to AlphaPept style modifications.

Parameters

sequence (*str*) – A peptide sequence with a DIA-NN style modification.

Returns

A peptide sequence with AlphaPept style modification.

Return type

str

alphaviz.preprocessing.**convert_diann_mq_mod**(*sequence: str*) → str

Convert DIA-NN style modifications to MaxQuant style modifications.

Parameters
sequence (*str*) – A peptide sequence with a DIA-NN style modification.

Returns
A peptide sequence with MaxQuant style modification.

Return type
str

alphaviz.preprocessing.**filter_df**(*df: DataFrame*, *pattern: str*, *column: str*, *software: str*) → DataFrame

Filter the data frame based on the pattern (any value) in the specified column.

Parameters

- **df** (*pd.DataFrame*) – The original data frame.
- **pattern** (*str*) – The string to be used to filter of the data frame column.
- **column** (*str*) – The column to be used to filter.
- **software** (*str*) – The name of the software tool where the filtering is used.

Returns
The filtered data frame.

Return type
pd.DataFrame

alphaviz.preprocessing.**get_aa_seq**(*protein_id: str*, *fasta*) → str

Extract the leading razor protein sequence for the list of proteinIDs of the protein group from the pyteomics.fasta.IndexedUniProt object.

Parameters

- **protein_ids** (*str*) – String containing the proteinIDs of all protein isoforms.
- **fasta** (*pyteomics.fasta.IndexedUniProt object*) – The pyteomics.fasta.IndexedUniProt object.

Returns
Protein sequence for the leading razor protein, e.g. from the list of proteinIDs 'Q15149;Q15149-7;Q15149-9;Q15149-5' the AA sequence for protein Q15149 will be returned.

Return type
str

alphaviz.preprocessing.**get_identified_ions**(*values: list*, *sequence: str*, *ion_type: str*) → list

For the specified peptide sequence extract all identified in the experiment ions and based on the specified ion_type return a list of booleans containing information for the b-ions whether the peptide is breaking after aligned amino acid or for the y-ion whether is breaking before aligned amino acid.

E.g. for the peptide 'NTINHN' having the unique ion values ['b2-H2O', 'b2', 'b3', 'b5-NH3'] it will return the following list of booleans: [False,True,True,False,True,False].

Parameters

- **values** (*list*) – The list of all unique identified ions for the peptide in the experiment.
- **sequence** (*str*) – Peptide sequence.
- **ion** (*str*) – Ion type, e.g. 'b' or 'y'. Other ion types are not implemented.

Returns
List of peptide length of booleans with True for the presenting ion and False for a missing one.

Return type
list

alphaviz.preprocessing.**get_mq_ms2_scan_data**(*msms: DataFrame*, *selected_msms_scan: int*, *raw_data*, *precursor_id: int*) → DataFrame

Extract MS2 data as a data frame for the specified MSMS scan number and precursor ID from the 'msms.txt' MQ output file and raw file.

Parameters

- **msms** (*pd.DataFrame*) – Pre-loaded 'msms.txt' MQ output file.
- **selected_msms_scan** (*int*) – MSMS scan number.
- **raw_data** (*AlphaTims TimsTOF object*) – AlphaTims TimsTOF object.
- **precursor_id** (*int*) – The identifier of the precursor.

Returns

For the specified MSMS scan and precursor ID, the extracted data frame contains the following columns:

- 'mz_values'
- 'intensity_values'
- 'ions'
- 'wrong_dev_value': whether the mass_deviation specified in the MQ table was incorrect.

Return type
pd.DataFrame

alphaviz.preprocessing.**get_mq_unique_proteins**(*filepath: str*) → list

Extract unique "Protein names" from the specified MaxQuant output file.

Parameters
filepath (*str*) – Full path to the file.

Returns
A list of unique protein names from the specified file.

Return type
list

alphaviz.preprocessing.**get_protein_info**(*fasta: dict*, *protein_ids: str*)

Get the name and the length of the protein(s) from the fasta file specifying the protein id(s).

Parameters

- **fasta** (*pyteomics.fasta.IndexedUniProt object*) – The Pyteomics object contains information about all proteins from the .fasta file.
- **protein_ids** (*str*) – The list of the protein IDs separated by comma.

Returns
The name and the length of the specified protein(s).

Return type
tuple of strings

alphaviz.preprocessing.**get_protein_info_from_fastaheader**(*string: str*, ***kwargs*)

Extract information about protein IDs, protein names and gene names from the "Fasta headers" column of the MQ output tables.

Parameters

string (*str*) – A 'Fasta header' string from the MQ output table for one protein group (e.g. from the proteinGroups.txt file).

E.g. a complex one: 'sp|Q3SY84|K2C71_HUMAN Keratin, type II cytoskeletal 71 OS=Homo sapiens OX=9606 GN=KRT71 PE=1 SV=3;;sp|Q14CN4|K2C72_HUMAN Keratin, type II cytoskeletal 72 OS=Homo sapiens OX=9606 GN=KRT72 PE=1 SV=2;;;sp|Q7RTS7|K2C74_HUMAN Keratin, type II cytoskeletal 74 OS'

Returns

The function returns a tuple of three strings containing information about the protein names, protein IDs and gene names.

Return type

a tuple of strings

alphaviz.preprocessing.**sort_naturally**(*line: str*, *reverse: bool = False*) → str

Sort the string natural to humans, e.g. 4,1,6,11 will be sorted as 1,4,6,11 and not like 1,11,4,6.

Parameters

- **line** (*str*) – The string to be sorted.
- **reverse** (*bool*) – Whether to apply the reverse option or not. Defaults: False.

Returns

A naturally sorted string.

Return type

str

CHAPTER

FIVE

ALPHAVIZ.UTILS

Functions:

calculate_mass(mono_mz, charge)	Calculate the precursor mass from mono mz and charge.
calculate_mz(prec_mass, charge)	Calculate the precursor mono mz from mass and charge.
check_analysis_file(file)	
check_github_version([silent])	Checks and logs the current version of AlphaViz. Check if the local version equals the AlphaViz GitHub master branch. This is only possible with an active internet connection and if no credentials are required for GitHub. :param silent: Use the logger to display the obtained conclusion. Default is False. :type silent: str.
get_frag_dict(parsed_pep, mass_dict)	Calculate the masses of the fragment ions :param parsed_pep: the list of amino acids and modified amono acids.
get_fragmass(parsed_pep, mass_dict)	Calculate the masses of the fragment ions :param parsed_pep: the list of amino acids and modified amono acids.
get_mass_dict([modfile, aasfile, verbose])	Function to create a mass dict based on tsv files.
get_precmass(parsed_pep, mass_dict)	Calculate the mass of the neutral precursor :param parsed_pep: the list of amino acids and modified amono acids.
parse(peptide)	Parser to parse peptide strings :param peptide: modified peptide sequence.

alphaviz.utils.**calculate_mass**(*mono_mz: float*, *charge: int*) → float

Calculate the precursor mass from mono mz and charge. :param mono_mz: mono m/z. :type mono_mz: float :param charge: charge. :type charge: int

Returns
precursor mass.

Return type
float

alphaviz.utils.**calculate_mz**(*prec_mass: float*, *charge: int*) → float

Calculate the precursor mono mz from mass and charge. :param prec_mass: precursor mass. :type prec_mass: float :param charge: charge. :type charge: int

Returns
mono m/z.

Return type
float

alphaviz.utils.**check_analysis_file**(*file*)

alphaviz.utils.**check_github_version**(*silent=False*) → str

Checks and logs the current version of AlphaViz. Check if the local version equals the AlphaViz GitHub master branch. This is only possible with an active internet connection and if no credentials are required for GitHub. :param silent: Use the logger to display the obtained conclusion.

Default is False.

Returns
The version on the AlphaViz GitHub master branch. "" if no version can be found on GitHub

Return type
str

alphaviz.utils.**get_frag_dict**(*parsed_pep: list*, *mass_dict: dict*) → dict

Calculate the masses of the fragment ions :param parsed_pep: the list of amino acids and modified amono acids. :type parsed_pep: list or numba.typed.List of str :param mass_dict: key is the amino acid or the modified amino acid, and the value is the mass. :type mass_dict: numba.typed.Dict

Returns
float}: key is the fragment type (b1, b2, …, y1, y2, …), value is fragment mass.

Return type
dict{str

alphaviz.utils.**get_fragmass**(*parsed_pep: list*, *mass_dict: Dict*) → tuple

Calculate the masses of the fragment ions :param parsed_pep: the list of amino acids and modified amono acids. :type parsed_pep: numba.typed.List of str :param mass_dict: key is the amino acid or the modified amino acid, and the value is the mass. :type mass_dict: numba.typed.Dict

Returns
the fragment masses and the fragment types (represented as np.int8). For a fragment type, positive value means the b-ion, the value indicates the position (b1, b2, b3…); the negative value means the y-ion, the absolute value indicates the position (y1, y2, …).

Return type
Tuple[np.ndarray(np.float64), np.ndarray(np.int8)]

alphaviz.utils.**get_mass_dict**(*modfile: str = 'data/modifications.tsv'*, *aasfile: str = 'data/amino_acids.tsv'*, *verbose: bool = True*)

Function to create a mass dict based on tsv files. This is used to create the hardcoded dict in the constants notebook. The dict needs to be hardcoded because of importing restrictions when using numba. More specifically, a global needs to be typed at runtime. :param modfile: Filename of modifications file. :type modfile: str :param aasfile: Filename of AAs file. :type aasfile: str :param verbose: Flag to print dict. :type verbose: bool, optional

Returns
Returns a numba compatible dictionary with masses.

Raises
FileNotFoundError – If files are not found.

alphaviz.utils.**get_precmass**(*parsed_pep: list*, *mass_dict: Dict*) → float

Calculate the mass of the neutral precursor :param parsed_pep: the list of amino acids and modified amono acids. :type parsed_pep: list or numba.typed.List of str :param mass_dict: key is the amino acid or the modified amino acid, and the value is the mass. :type mass_dict: numba.typed.Dict

Returns
the peptide neutral mass.

Return type
float

alphaviz.utils.**parse**(*peptide: str*) → List

Parser to parse peptide strings :param peptide: modified peptide sequence. :type peptide: str

Returns
a list of animo acids and modified amono acids

Return type
List (numba.typed.List)

PYTHON MODULE INDEX

a

INDEX