Protocol V01_02

Author: Pierre-Alexandre HO Last Update: February, 9th 2024

XICformat MANUAL

Abstract

The Aim of this protocol is to describe the usage of the python script XICformat in order to improve the reporting after a PROSTAR processing. First the installation process of python and packages are explained then the good practice for using this script. XICformat script can be used by both DDA and DIA analysis. In a second part, this protocol documents the code.

History

Date	Author	Version	Descriptions
09/02/2024	Pierre-Alexandre	01	Creation
04/11/2024	Pierre-Alexandre	01_02	Adjustment for public access

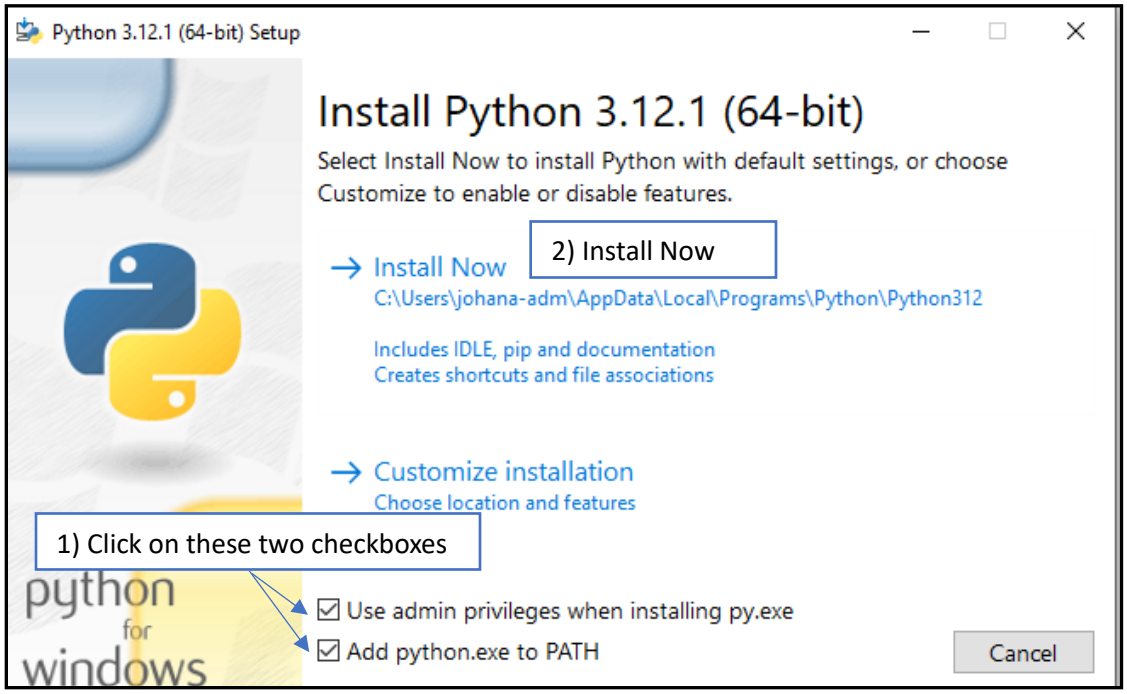
Prerequisites

Python Installation

The script run with **Python 3.12.0**

The installation of python is mandatory to use the script. It can be download directly from the official website: https://www.python.org/.

Copy-paste the application file in your destock. Then run the installation process by a double click or right click and “Open as admin”.



The installation window appears. Before running the installation, check the following boxes:

- Use admin privileges when installing py.exe
- Add python.exe to PATH

To start the installation clink on “Install Now”. The Setup process will only take few seconds.

After this step the installation control cans ask to disable path length limit. Click on it !

In order to check if python was well installed, go to the searching toolbox of window and write **IDLE**. It's an integrated development environment (IDE) for Python programming language. IDLE comes bundled with the standard Python distribution, making it easily accessible to users without requiring additional installations. It is also possible to check the installation with the command prompt (cmd) on Windows with the command.

Packages Installation

The script needs some additional packages in order to run correctly (*package == version)*:

- pandas == 2.1.1
- numpy == 1.26.0
- scikit-learn == 1.4.0
- openpyxl == 3.1.2
- plotly == 5.18.0
- matplotlib == 3.8.0
- seaborn == 0.13.2

To install them, it is possible to use the prompt command or the Anaconda tools. Anaconda is a distribution of Python, specifically geared towards data science and machine learning applications. It includes a package manager, environment manager, and other tools for simplifying the process of installing, managing, and working with Python and its dependencies. Anaconda comes with many pre-installed libraries commonly used in data science, such as NumPy, pandas, matplotlib, scikit-learn, and more. It can be download at this following address: https://www.anaconda.com/download

The second way to install the package is to use the prompt of windows with the **pip** command. write “cmd” in the searching toolbox of window. Then use the following syntax: *pip install package == version*. The version is not mandatory, so the most recent version will be installed. The easiest way is to copy/past the command line below:

```
pip install pandas==2.1.1 numpy==1.26.0 scikit-learn==1.4.0 openpyxl==3.1.2 plotly==5.18.0 matplotlib==3.8.0 seaborn==0.13.2 statsmodels==0.14.1
```

If it’s not working, it is possible to write line by line

```
pip install pandas==2.1.1
pip install numpy==1.26.0
pip install scikit-learn==1.4.0
pip install openpyxl==3.1.2
pip install plotly==5.18.0
pip install matplotlib==3.8.0
pip install seaborn==0.13.2
pip install statsmodels==0.14.1
```

NOTE: If you download the most recent package and the script is not running well you can try do download the same version used in the script with the syntax seen above.

To check if the package is correctly installed, open IDLE and write the command below:

- import name_of_package

If the package is properly installed, a new line (symbolize by >>>) will appears

If the package is not properly installed a error message will appears as "No module founds".

Remark: for scikit-learn use the following command: import sklearn

After this step, the computer is ready to read and execute the scripts.

Script Usage

Preparation of the folder

To execute the script, it is mandatory to have in the same folder this two Excel files:

- PROSTAR report with 3 sheets:
  - Quantitative Data sheet #1
  - Samples Meta Data sheet #2
  - Feature Meta Data sheet #3
- Template Final result report with at least 1 sheet
  - Quanti XIC (the name must to be this one)

  Note: Copy the "Template_Quanti_XIC.xlsx" find in the root of the directory and paste it in the desired directory.

Additional excel files can be also load:

- PROLINE file:
  - Protein set sheet #3
- Go Anotation file:
  - 1 sheet
  - The annotation file must to be converted to an excel file before using it

Script

There are several script (which be details below). Use the **XICformat.py** by double clicking. A command prompt will appear. The Initialization start (could take few seconds).

A succession of input will appear

```
Write directory link (windows link):  ➔Mandatory
Write the name of the PROSTAR excel file (without .xlsx): ➔Mandatory
Write the name of the PROLINE excel file (without .xlsx): ➔ Non Mandatory
Write the name of the annotation GO excel file (without .xlsx) ➔ Non Mandatory
Define the log10 adjusted p-value threshold: ➔Mandatory
Define the log2 Fold Change threshold in absolute value: ➔Mandatory
Write the name of the results Excel file (without .xlsx): ➔Mandatory
```

! WARNING !
If something is not well written, the script could crash. Also if there is nothing written in mandatory part the script will crash.
DO NOT write the extension part of the Excel file (.xlsx).
The threshold must be written in **absolute log values**: log10 for the *p*-value, in general 1.3 (0.05%) and log2 for the Fold Change, in general 1 (for 2 and 0.5)

Export

The merged tables are directly print into the Quanti XIC sheet in the final results excel file. A Statistic sheet is created with tables with some statistics values.

Et new directory is created with some pictures, heatmap, PCA and interactive pictures (.htlm): 3D PCA, multiple PCA, volcano plot.

A txt file with txt with tabulation separator is created containing the accession and description column then the Fold Change and the PROSTAR *p*-value

Modify the code

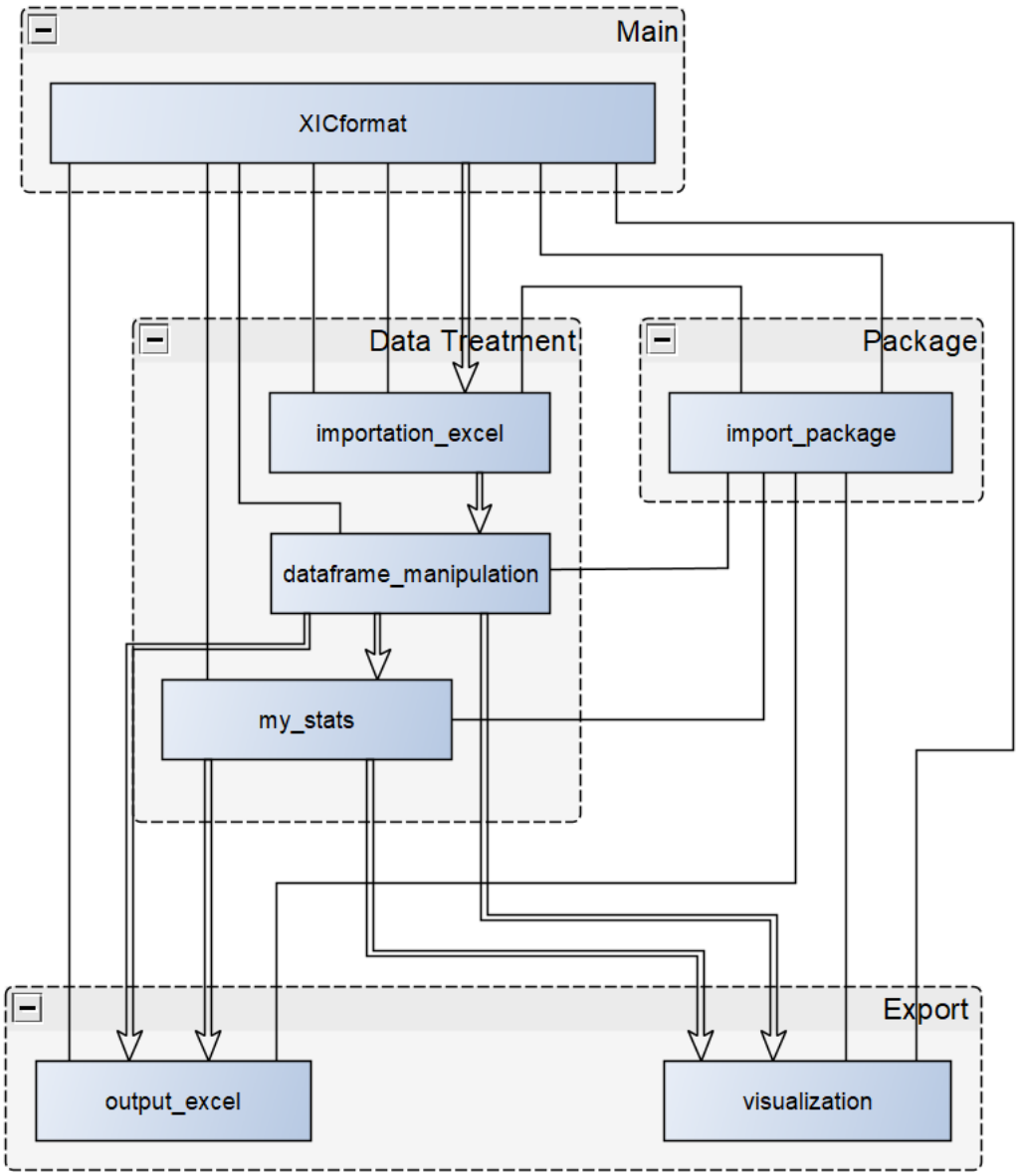
In order to modify the code, an integrated development environment (IDE) is needed. Spyder is one of the easiest. It is very closed to RStudio and it also has a variable explorer. Download spider: https://www.spyder-ide.org/

Note: All the code is comment. For more details, look directly into the script.

Code description

Architecture

The workflow is composed of 7 python files. The main file is the is **XICformat.py** which is connected to the other via the import function. All this files must to stay in the same directory.



XICformat.py

The main file, call the functions from the different modules:

1) Call input functions from XICformat.py
   - Create the PATH to the excel files (import and export)
   - Create the export directory
   - Define the threshold Log10 p-adjusted *p*-value & absolute Log2 FoldChange
2) Call import function from importation_excel.py
   - Create dataframe from the different excel tables
3) Call functions to manipulate the dataframe from dataframe_manipulation.py
   - Selection of the interested column
   - Addition of the adjusted *p*-value by Benjamini-Hoshberg method
   - Merge the tables for the statistics part and the export in excel
4) Call statistical function from my_stats.py
   - Give the percentage of each imputation method
   - Give the percentage of the significant protein for each 2 by 2 comparison
   - Count the number of identified protein and the number of spectrums in with PROSTAR process
   - Count the number of identified protein and the number of spectrums in with PROLINE process
   - Perform the PCA
5) Call visualization functions from visualizaition.py
   - Create an heatmap of all proteins (.svg)
   - Create interactive volcano plot of all comparison (.html)
   - Create PCA plot of PC1 and PC2 (.svg & .png 300 dpi)
   - Create interactive 3D PCA (.html)
   - Create interactive multiplot PCA (.html)
   - All the picture are save to the export directory
6) Export of the table
   - Call function to export the table to the Template Final result report from output_excel.py
   - Create a txt file export_beatrice to the export directory

import_package.py

This file contained all the import of the different packages needed to run the script.

- import os *# manage the prompt*
- import pandas as pd *# manage the dataframe (table) manipulation*
- import numpy as np *# manage the operation on vectors and matrix*
- import openpyxl *# manage the excel manipulation*
- import subprocess *# Open the folder*
- from statsmodels.stats.multitest import multipletests *# Manage some statistics analysis*
- import seaborn as sns *# Plot statistical clustering*
- from sklearn.decomposition import PCA *# Principal Component Analysis*
- from sklearn.preprocessing import StandardScaler *# Standardized the PCA*
- import matplotlib.pyplot as plt *# Plot statistics graphics*
- import plotly.express as px *# Plot interactive graphics*
- import plotly.graph_objs *as* go *# Plot interactives graphics*

This file contain also a function progress_bar designed to generate a progress bar in the console/terminal.

This file is linked to the other files thanks to the command below:

- from import_package import *

importation_excel.py

Contain two functions:

- **import_excel**: used to catch a specific excel sheet in a pandas dataframe.
- **transform_into_df**: used to define a dataframe following the deisered sheet. This function call import_excel() for each sheet desired and return all the dataframe:
  - **quanti** : df from “Quantitative data”, sheet n°0 of PROSTAR export
    - gives the normalized XIC abundance of the each protein for each sample
      - Columns: Samples
      - Ligne: Proteins
  - **meta**: df from “Samples Meta Data”, sheet n°1 of PROSTAR export
    - gives the different conditions each samples
    - 3 columns, first for the samples second for the condition, third for the BioRep (not used)
  - **feature**: df from “Feature Meta Data”, sheet n°2 of PROSTAR export
    - gives all information of each preotein included:
      - metatdata like imputed method (metacell_abundance columns), description, accession….
      - numerous values: spectral count, raw data…
    - gives comparison 2 by 2 results
      - Log_2 Fold Change
      - *p-value non adjusted*
  - **go_set:** df from sheet n°0 of Go annotation excel file
    - Not mandatory
    - If it not existing return None
  - **prot_set**: df from “Protein set”, sheet n°3 of PROLINE export
    - Not mandatory
    - If it not existing return None
    - gives all information of each protein for each sample included spectral count

In the main scrip, only **transform_into_df** is called and return the corresponding dataframe

dataframe_manipulation.py

Call by the main script as manip.

This file contain 6 functions in order to manipulate the dataframe and transform them for a better export or statistics analysis:

- **keep_only_col**: keep the desired column of the dataframe
  - Create a list of the name of columns desired then return a dataframe with the columns of this list. If there is an empty value in a cell, it is replaced by 0
    - At minimum there is the columns 'accession', 'description', 'samesets_accessions', 'subsets_accessions'
    - the columns are kept if they stat with a specific name (*name_col*)
      - if there is not *name_col*: return just the first 4 columns
    - The 2 by 2 comparison results are the last columns in the dataframe. If needed, add compare=True (False by default)
      - They start after the last metacell column
- **scientific**: transform the value into a scientific number if it is strictly <0.05
- **manipulation**: manipulate the feature an prot_set dataframe and return specific ones
  - **metacell_comp**: df from feature and keep only the column of imputed method stating with “metacell” and the comparison 2 b 2 results
    - Columns: metacell samples + comparison 2 b 2 results with Benjamini-Hochberg adjusted *p*-value in scientific mode

        - Ligne: Proteins
    - **psm:** df of spectral count aboundance of PROSTAR export via feature df
        - Columns: Samples
        - Ligne: Proteins
    - **beginning**: only the 4 first columns
        - 'accession', 'description', 'samesets_accessions', 'subsets_accessions'
        - Ligne: Proteins
    - **spc:** df of spectral count aboundance of PROLINE export via prot_set df
        - Columns: Samples
        - Ligne: Proteins
    - **looking_adjpv**: df with only Benjamini-Hochberg adjusted *p*-value of the 2 by 2 comparison results
- **quanti_meta_transform**: return dataframe in order to an easier manipulation during statistics treatment
    - **trans_quanti**: transposed of quanti dataframe
    - **meta**: remove Bio.Rep
- **merging**: return merged dataframe
    - **full_table**: dataframe for the export in Excel file
    - **table_stats**: dataframe for statistics treatment
        - Columns: condtion + Proteins
        - Ligne: Samples
- **comparison2x2_and_exp_bea**: Simplified dataframe comparison + table for the beatrice export.
    - **looking_FC**: focusing on only Fold Change columns
        - Columns: Comparison
        - Ligne: Proteins
    - **looking_2x2compare**: columns 'accession', 'description' +[Fold Change columns + log10 adjusted *p*-value + Significant (Yes) or not (No) following a specific threshold] for each comparison in this order
        - Columns: Comparison
        - Ligne: Proteins
    - **export_beatrice**: columns 'accession', 'description' +[Fold Change columns + non-transformed *p*-value] for each comparison in this order
        - Columns: Comparison
        - Ligne: Proteins

Note: All variables in the following format: looking_XXX means that the dataframe is focus on the XXX property.

my_stats.py

Call by the main script as st.

Functions to counts some properties and to performed the PCA calculation.

- **counting**: count some properties into datraframe and return datraframe of the results
    - percentage_imputation: gives the percentage of each method of imputation for each samples.
    - **percentage_signif2x2**: gives the percentage of number of significant proetein (Yes) and non-significant (No) for each comparison.
    - **summary_psm**: gives the number of identified protein and the number of spectra by spectral count method in the PROSTAR export.

    - **summary_spc**: if the PROLINE export excel file exists, gives the number of identified protein and the number of spectra by spectral count method in the PROLINE export.
- **find_M**: for each sample, find the first M (Missing Entire Condition) imputation in looking_metacell dataframe.
    - **imputation_value**: Return a dataframe with the value of this imputation for each sample
- **my_pca**: perform the PCA statistical analysis where proteins explain the samples
        - Columns: Proteins
        - Ligne: Samples
    - **table_pca**: dataframe of the principal component values
        - Columns: Conditions + the 5 first principal component
        - Ligne: Proteins
    - **scree**: dataframe of the statistics of the PCA
        - Columns: Percentage of explained variance + percentage of cumulative explained variance
        - Ligne: principal component
    - **pca**: result of the PCA

visualization.py

Call by the main script as viz.

Functions create graphics

- **fullheatmap**: create a static heatmap of proteins based on the normalized abundance: *quanti*
    - use the function clustermap from seaborn package
    - clustering method = average
    - distance calculation = eulidiean
- **volcano_plot**: function to create **one interactive** volcano plot
    - return a figure in html format
- **volcanos**: function called in the main script to create all the volcano plots of the 2 by 2 comparison
    - call the volcano_plot function for each column of looking_fc (dataframe of fold change columns)
    - search the column of looking_fc in looking2x2compare then catch adjusted *p*-value and the significant columns following their coordinate in the dataframe
- **scree_plot**: plot static bar plot of explain variance and the cumulative explain variance of the PCA
- **plot_PCA**: different graphics to represent the PCA analysis
    - graph: static plot of principal component 1 vs principal component 2
    - multiple: gives all interactive 2D plot of the 5 first principal component
    - 3D: interactive plot of the 3 first principal component
    - loading: interactive bar plot depicting the implication of each protein for the first five principal components. One bar plot by principal component

output_excel.py

Functions to write the dataframes into the Template Final result report directly in “Quant XIC” sheet and by creating the “statistic” sheet.

- **sheet_exists**: check if the sheet exist. If no not create it
- **add_table**: add one dataframe into the a sheet
- **ouput**: called by the main script to implement all the dataframe in differents sheets
    - call add_table function for each dataframe
- **styling**: define the size of some columns in the excel file