Protocol for MicrobioLink: a tool for predicting host-microbe interactions and their downstream effect on host cells

Lejla Gul[1,2,5*], Anna Julia Elias[2,3], Tanvi Tambaku[1], Marton Olbei[1], Emily Watters[1], Balazs Bohar[1], Dezso Modos[1], Matthew Madgwick[2,4], Tamas Korcsmaros[1,2,6**]

[1] Department of Metabolism, Digestion and Reproduction, Faculty of Medicine, Imperial College London, London, W12 0NN, United Kingdom
[2] Quadram Institute, Norwich, Norfolk, NR4 7UQ, United Kingdom
[3] Department of Morphology and Physiology, Faculty of Health Sciences, Semmelweis University, Budapest, 1086, Hungary
[4] Earlham Institute, Norwich, Norfolk, NR4 7UZ, United Kingdom
[5] Technical contact
[6] Lead contact
*Correspondence: l.potari-gul@imperial.ac.uk
**Correspondence: t.korcsmaros@imperial.ac.uk

Summary

Analyzing interspecies interactions between hosts and microbes is essential for understanding how changes in the microbiota can disrupt host homeostasis and lead to disease. MicrobioLink is a computational pipeline that predicts host-microbe protein-protein interactions and their downstream effects on host cellular processes, providing insights into how microbial components influence human signaling pathways. Unlike existing protocols, MicrobioLink integrates multi-omic data and structural interaction predictions using network biology approaches, enhancing our understanding of these complex interactions, particularly in disease contexts.

For complete details on the use and execution of this protocol, please refer to Poletti et al. 2022 and Gul et al., 2022.

Highlights

- A protocol for MicrobioLink to analyze host-microbe impact on human cell signaling.
- Predicting protein-protein interactions using structural data and domain-motif links
- Mapping signaling networks from bacteria-targeted proteins to expressed genes.

Before you begin

The protocol below describes a detailed step-by-step guideline to predict host-microbe protein-protein interactions and analyzes their downstream impact on the host cellular signaling network. Setting up the computational environment is crucial to ensure compatibility and smooth execution of the MicrobioLink pipeline. Proper configuration of hardware, software, and required packages minimizes potential errors during the analysis and optimizes performance, particularly when handling large biological datasets. This section includes recommendations for the hardware specifications, outlines steps for localizing the pipeline, and explain the process for setting up the necessary environment. It also details how to download and install essential software tools.

Note: To effectively visualize networks generated by this analysis, ensure that Cytoscape version 3.10.2 is installed on your system. Cytoscape will be used to create interactive graphical representations of host-microbe interaction networks, facilitating interpretation of the results.

For general troubleshooting issues related to software or package unavailability, missing parameters, empty output files, or directory errors, please refer to the "*Troubleshooting*" section at the end of this protocol. This section provides solutions to common setup and execution problems.

Downloading multi-omic data for the case study

TIMING: 5-60 mins (depending on the size of the datasets)

MicrobioLink combines host transcriptomics and bacterial proteomics that enables a more comprehensive analysis of host-microbe interactions, linking microbial influence directly to changes in gene expression and protein function in the host. The pipeline requires three main input files with specific file formats:

1. **Human transcriptomic data:** This dataset provides gene expression profiles, which are crucial for identifying how microbial interactions might influence gene regulation within the host. The pipeline uses average gene expression counts from processed single-cell or bulk RNA-seq data, enabling downstream analysis to focus on potential gene expression changes.

2. **Endpoint file:** The endpoint file provides a focus for the analysis by defining the target genes that the pipeline seeks to connect to bacterial proteins, with the aim of exploring which signaling pathways may be perturbed to influence these specific genes. The endpoint file can contain genes that are differentially expressed between conditions (p-value < 0.05) along with their fold change values. In this case, MicrobioLink identifies signaling pathways potentially impacted by bacterial interactions that could influence differential gene expression. Alternatively, the endpoint file may include a list of genes derived from a single condition. For example, it could contain genes encoding secreted ligands described in inflamed condition. In this case, MicrobioLink assesses how bacterial interactions might influence cellular communication mediated through these secreted ligands.

3. **Bacterial proteomics or metaproteomics data:** This dataset includes proteins from the bacterial proteome, representing the microbial components that may interact with host proteins. The pipeline accepts either a list of bacterial UniProt IDs or a UniProt Proteome (UP)

ID, which allows the pipeline to download and analyze the complete proteome of a bacterial strain. The UP option is particularly useful for large-scale analyses where all potential proteins from a specific bacterial strain are of interest. Analyzing these interactions across bacterial strains provides insights into the broader effects of microbes on host cellular processes, including both pathogenic and probiotic effects.

For the case study, we use public single-cell data from (Kong et al., 2023) and proteomic data derived from *Bacteroides thetaiotaomicron* extracellular vesicles published by (Gul et al., 2022). All necessary input data for running this protocol are available in the GitHub repository (https://github.com/korcsmarosgroup/MicrobioLink2).

Downloading and installing the pipeline and the required software

TIMING: 1-2 h

To ensure optimal performance, we recommend the following hardware specifications:

- **RAM (Memory)**: At least 8GB of RAM is recommended for typical datasets, with 16GB preferred for large-scale analyses. Adequate memory helps prevent slowdowns or crashes during data processing, especially when performing network analyses.
- **Storage**: Ensure at least 1GB of free disk space to accommodate input files, intermediate data, and output files. The exact storage needs may vary depending on dataset size, so additional space may be required for extensive data.
- **Graphics requirements**: Network visualization in Cytoscape or other visualization tools requires a system with modern integrated graphics to render large or complex networks smoothly.
- **Operating System**: The pipeline is compatible with Windows, macOS, and Linux.

The MicrobioLink pipeline depends on a range of Python libraries and bioinformatics tools for handling large datasets, performing statistical computations, and visualizing complex networks. Installing the correct versions of these packages will help avoid compatibility issues and ensure that the pipeline runs efficiently. Key software and libraries required for the pipeline include:

- **Pandas** (v2.2.2), **numpy** (v1.26.4), and **scipy** (v1.13.1): These libraries support data manipulation and numerical operations which are fundamental for processing large biological datasets.
- **MyGene (**v3.2.2) and **OmniPath** (v1.0.8): MyGene provides fast access to gene annotations, while OmniPath offers the primary knowledge network of protein-protein interactions, essential for the downstream network analysis.
- **Pyfasta** (v0.5.2) and **Biopython** (v1.84): These tools facilitate the handling and processing of FASTA files, which are standard in bioinformatics for storing protein and nucleotide sequences.
- **Cytoscape** (v3.10.1): Cytoscape is required for visualizing the interaction networks generated by the pipeline, allowing users to interpret host-microbe interaction patterns and downstream effects visually. Make sure Cytoscape version 3.10.1 is installed to ensure compatibility with the pipeline’s output formats.

Main steps for downloading and installing the pipeline and required software:

1. Download Anaconda (recommended): Anaconda is a widely used package and environment manager that simplifies the installation and management of required software packages. Details on installation are described here: https://docs.anaconda.com/free/anaconda/install/index.html

   Alternative: If users cannot use the Conda environment, proceed directly with Python environment setup as described below.

2. Download the MicrobioLink pipeline from GitHub (https://github.com/korcsmarosgroup/MicrobioLink2) by either downloading the zip archive and then extracting it, or by using the git-clone command:

   ```
   >git clone https://github.com/korcsmarosgroup/MicrobioLink2.git
   ```

   Note: Ensure that the microbiolink_env.yml file, which specifies required packages, is located within the workflow folder of the repository. This file will facilitate the automatic installation of compatible software versions within the environment.

3. Setting up the environment:

- **Using Anaconda (recommended):** Before the environment initiation, it is essential to have Conda installed on the system, details are described in Step 1. To create a Conda virtual environment with the required packages, navigate to the downloaded MicrobioLink2 directory and execute the following commands:

  ```
  >cd MicrobioLink2

  >conda create --name microbiolink --file
  workflow/microbiolink_env.yml
  ```

- **Alternative setup without Conda:** For users unable to use Conda, they can manually create a Python environment using venv and install each package individually:

  ```
  >python3 -m venv microbiolink_env
  ```

  Install the required packages as listed in the workflow/microbiolink_env.yml using pip.

4. Activating the environment:

- **Using Anaconda (recommended):** Open the Anaconda terminal by going to the "Environments" – select the appropriate environment and click "Open Terminal".

  ```
  >conda activate microbiolink
  ```

- **Alternative activation without Conda:** Use the following command in Terminal/Command Line to activate the virtual environment:

```
> source microbiolink_env/bin/activate # On Windows:
`microbiolink_env\Scripts\activate`
```

Key resources table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Human single-cell RNA-seq data (processed)	(Kong et al., 2023)	https://github.com/korcsmarosgroup/MicrobioLink2/tree/main/case_study_input/input/human_protein/Kong_et_al_avg
Bacterial proteomics	(Gul et al., 2022)	https://github.com/korcsmarosgroup/MicrobioLink2/tree/main/case_study_input/input/bacterial_protein
Software and algorithms		
Pandas v2.2.2	http://conference.scipy.org.s3-website-us-east-1.amazonaws.com/proceedings/scipy2010/pdfs/mckinney.pdf	https://pandas.pydata.org; RRID:SCR_018214
Numpy v1.26.4	(Harris et al., 2020)	http://www.numpy.org; RRID:SCR_008633
Scipy v1.13.1	(Virtanen et al., 2020)	http://www.scipy.org/; RRID:SCR_008058
Mygene v3.2.2	(Wu et al., 2013)	RRID:SCR_018660

OmniPath v1.0.8	(Türei et al., 2021)	https://omnipathdb.org/
Gget v0.28.6	(Luebbert and Pachter, 2023)	https://github.com/pachterlab/gget
Pyfasta v0.5.2	https://pypi.org/project/pyfasta/0.2.9/	https://github.com/brentp/pyfasta
Biopython v1.84	(Cock et al., 2009)	http://biopython.org; RRID:SCR_007173
Python v3.9	(Python Software Foundation, 2016)	https://www.python.org/; RRID:SCR_008394
Cytoscape v3.10.1	(Shannon et al., 2003)	https://cytoscape.org; RRID: SCR_003032
Other		
Code to reproduce the analysis	This publication	https://github.com/korcsmarosgroup/MicrobioLink2/tree/main/workflow

Step-by-step method details

All parts of the MicrobioLink pipeline can be run from the terminal with the necessary command-line argument defined by each script. Each parameter is defined with a specific help message, providing users with guidance on the expected format and purpose of the argument. Example usage of each step is shown using files from the provided GitHub repository ("*case_study_input/input*" and "*case_study_output/output*" folders), allowing users to follow along with a real use case. The following sections include step-by-step instructions and visual examples to ensure users can accurately format their inputs and avoid common errors.

Preparing input files for the MicrobioLink pipeline

TIMING: 2 x 5-10 mins depending on the size of data

To ensure accurate predictions of host-microbe interactions and downstream effects, input files should be carefully prepared and formatted to meet the requirements of the pipeline. Below, we outline the preparation steps for each input file.

1. Z-score filtering for human transcriptomic data (z-score_filter_terminal.py)

This step involves filtering gene expression data to reduce noise by excluding lowly expressed genes. Filtering gene expression data to exclude genes below a set Z-score threshold ensures that only genes with relatively high expression levels are included in the analysis. This script takes bulk or single-cell gene expression data in normalized average expression count matrix format as input. This file must be formatted as a CSV with gene symbols in the first column and normalized average expression counts in subsequent columns. A sample format is provided in Figure 1, showing the correct structure for efficient processing. The script performs log2 transformation, filters out lowly expressed genes based on z-score cut-off, and outputs the filtered results into a CSV file.

Run the script with the necessary command-line arguments:

a. The parameter **"--input_file**" is required and is used to specify the path for the input CSV file that contains the gene expression data the program will process.

b. The parameter "**--output_file**" is required to specify the name and path for the output CSV file where filtered results will be saved.

c. The "**--zscore**" parameter is used to set a Z-score cut-off for filtering out lowly expressed genes in the dataset. If this parameter is not provided, it defaults to -3. By adjusting this parameter, users can control the threshold for filtering based on gene expression levels. The parameter expects a numeric value (e.g., -3, -2.5, -1), typically representing the number of standard deviations below the mean expression level, with higher values filtering out more genes.

```
>cd MicrobioLink2/workflow

>python z-score_filter_terminal.py --input_file
“case_study_input/input/human_transcriptomics/colon_BEST4_enterocyt
e_CD.csv” --output_file
“case_study_input/input/human_transcriptomics/enterocyte_colon_CD_z
score.csv” --zscore -3
```

Optional: This step may be skipped if the user does not want to filter lowly expressed genes, or if the necessary filtering has already been performed.

Note: Z-score filtering only works well for data with a normal distribution. If the data does not follow normal distribution, the least 10% of expressed genes may be excluded from the analysis. It is the user’s responsibility to assess this.

Note: Gene identifiers (gene symbol or UniProt) must be in the first column in the provided dataset.

Note: Z-score default cut-off value is set to -3 (Hart et al., 2013) but can be changed by the user.

2. Obtain the human protein FASTA file (get_human_fasta.py)

This step retrieves protein sequences for human genes in FASTA format. This file is essential for the protein interaction predictions, as it provides the sequences used to identify domain binding motifs and predict host-microbe interactions. The input file must contain the identifiers (gene symbol or UniProt ID) in the first column while corresponding expression values must be placed in the second column (Figure 2). The script offers a filtering for membrane-based or secreted proteins that is crucial when analyzing host-microbe interactions involving extracellular bacteria, as their interactions with the host typically occur through the cell membrane.

Run the script with the necessary command-line arguments:

a. The "**--gene_expression**" parameter is a required argument that specifies the file path to the transcriptomics data. If the z-score filter is applied then this file is the output of Step 1, otherwise the normalized average gene count matrix (described in the “*Before you begin*” section).

b. The "**--id_type**" parameter is a required argument that specifies the type of gene identifier used in the transcriptomics data file. This parameter restricts input to specific options, offering only two valid choices: "genesymbol" or "uniprot". By setting choices=["genesymbol", "uniprot"], the program ensures that users select one of these options, avoiding potential errors in data interpretation.

c. The "**--sep**" parameter is a required argument that specifies the field separator (e.g., ";", "\t", or "|") used in the input file, allowing the program to correctly parse and interpret the data.

d. The "**--location_filter_list**" parameter allows users to narrow down protein analysis based on specific cellular locations. This parameter accepts multiple location filters as a space-separated list. If no filter list is provided, the program will default to *None*, meaning no location-based filtering will occur, and all proteins will be included in the analysis. Available filter options include:

   i. "*plasma_membrane_transmembrane*" for proteins located within the plasma membrane and spanning across it,
   ii. "*plasma_membrane_peripheral”* for proteins associated with the peripheral site of the plasma membrane, and
   iii. "*secreted*" for proteins that are secreted into the extracellular space.

e. The "**--output_folder**" parameter is a required argument that designates the folder where the script will save its output files. The default value is ".", which represents the current directory, hence if a specific folder path is not provided, the script will save the output files to the directory from which it was run.

f. The "**--output_sequences"** parameter is a required argument that specifies the filename for saving protein sequences. If no filename is provided, the program will save protein sequences to a file named "*protein_sequences.fasta*" in the specified output folder or the current directory.

```
>cd MicrobioLink2/workflow

>python get_human_fasta.py --gene_expression
“case_study_input/input/human_transcriptomics/enterocyte_colon_CD_z
score.csv” --id_type "genesymbol" --sep "," --location_filter_list
[plasma_membrane_transmembrane plasma_membrane_peripheral] --
output_folder “case_study_input/input/human_transcriptomics/” --
output_sequences “protein_sequences.fasta”
```

Note: The Uniprot or gene symbol IDs must be found in the first column of the file while corresponding expression values must be placed in the second column.

Note: Options for subcellular location filtering are the following: *plasma_membrane_transmembrane* and/or *plasma_membrane_peripheral* and/or *secreted*. Required format is separated by space and without quotation marks and brackets. E.g.: [plasma_membrane_transmembrane plasma_membrane_peripheral]. Default: None.

Note: If the script fails to fetch data for one or more UniProt IDs it prints the message "Failed to fetch data for uniprots: {uniprots}".

Note: Output filename should be in .fasta format and can be personalized using command-line arguments. Default: "protein_sequences.fasta"

Note: The following error message may appear if the mygene package is not installed properly: ‘AttributeError: partially initialized module 'charset_normalizer' has no attribute 'md__mypyc’’. Further details and solutions for this issue are provided in the "Troubleshooting" section.

3. Downloading bacterial proteins with their domain structure (download_bacterial_proteins.py)

This script downloads bacterial protein domain structures from the UniProt database. The input file should include a column with UniProt or UniProt Proteome (UP) IDs of the bacterial proteins (Figure 3). Ensure no headers or additional information is included beyond these columns, as alteration may cause compatibility issues. These bacterial proteins serve as the microbial component in host-microbe interaction predictions. Using the UP option allows for a more comprehensive analysis of potential microbial interactions across all proteins of a strain, especially useful in large-scale studies.

Run the script with the necessary command-line arguments:

a. The "**--id_list**" parameter is a required argument that specifies the file path to an existing file containing a list of identifiers, either in UniProt or UP ID format.

b. The "**--sep**" parameter is a required argument that specifies the field separator used in the input file, helping the script correctly parse and interpret the data. The argument expects a single character (e.g., ";", "\t", or "|") representing the separator between fields in the input file.

c. The "**--id_type**" parameter is a required argument that defines the type of identifier used in the input data - "Uniprot" or "UP".

d. The "**--id_column**" parameter is a required argument that specifies the column number containing the proteome or protein ID within the input file. The argument expects an integer representing the column number in the input file where the proteome or protein IDs are located. User provided "--id_column" 1 would refer to the first column.

e. The "**--output**" parameter is a required argument that specifies the file path and name where the program will save its output.

```
>cd MicrobioLink2/workflow

>python download_bacterial_proteins.py --id_list
“case_study_input/input/bacterial_protein/OMV_proteins.csv” --sep
"," --id_type "Uniprot" --id_column 1 --output
“case_study_input/input/bacterial_protein/BT_BEV_domains.tsv”
```

Note: Python starts to count at 0, therefore the user-provided column number is automatically decreased by one. Users can count as normal.

Predicting Interactions Between Human and Microbial Proteins Based on Domain-Motif Interactions

TIMING: 10-20 mins depending on the size of data

This step focuses on predicting protein-protein interactions between human and bacterial proteins through domain-motif interactions (DMIs), highly specific interaction points that are fundamental to protein binding and function. In structural biology, DMIs are recognized as essential elements to facilitate precise binding between proteins by aligning complementary structures. These interactions are essential for many biological processes, allowing proteins to communicate, modify each other's activity, or drive signaling events within cells.

Additionally, a quality control measure is applied to reduce false positives by focusing on interactions where motifs are located within disordered protein regions, as these are more likely to allow flexible interactions necessary for binding.

4. Host-microbe protein-protein interaction prediction (DMI.py)

This script uses structural data, specifically DMIs, to predict interactions between human and microbial proteins. The analysis is based on *in vitro* verified DMIs from the Eukaryotic Linear Motif (ELM) database, which provides validated motif-domain relationships in eukaryotic proteins. As shown in Figure 4, the script takes input files containing data on human protein sequences (Step 2), motif regular expression (regex) patterns and DMIs from the ELM database, and bacterial protein domains (Step 3). The script utilizes external libraries and modules, such as pyfasta and re, to process and analyze biological sequence data and interaction predictions between human and microbial proteins.

Run the script with the necessary command-line arguments:

a. The "**--fasta_file**" parameter is a required argument that specifies the path to a FASTA file containing human protein sequences.

b. The "**--elm_regex_file**" parameter is a required argument that specifies the path to a file containing motif regular expressions from the ELM database.

c. The "**--motif_domain_file**" parameter is a required argument that specifies the path to a file containing motif-domain interaction data from the ELM database.

d. The "**--bacterial_domain_file**" parameter is a required argument that specifies the path to a file containing bacterial protein domain information.

e. The "**--output_file**" parameter is a required argument that specifies the filename and path where the script will save its output file.

```
>cd MicrobioLink2/workflow
```

```
>python DMI.py --fasta_file
“case_study_input/input/human_transcriptomics/protein_sequences.fas
ta” --elm_regex_file “ case_study_input/input/elm
/elm_classes_2020.tsv” --motif_domain_file
“case_study_input/input/elm /elm_interaction_domains.tsv” --
bacterial_domain_file
“case_study_input/input/bacterial_protein/BT_BEV_domains.tsv” --
output_file
“case_study_output/output/HMI/BT_enterocyte_cd_prediction_output_us
ecase.csv”
```

Note: ELM Regex file: The input file should be in .tsv format. ELM Identifiers are required to be in the second column, while Regex information is in the fifth column. The required table is provided in the GitHub repository or can be downloaded directly from the ELM database: http://elm.eu.org/elms/elms_index.tsv

Note: ELM Interaction file: The input file should be in .tsv format. ELM Identifiers are required to be in the first column, while associated Pfams should be in the second column. The required table is provided in the GitHub repository or can be downloaded directly from the ELM database: http://elm.eu.org/interactions/as_tsv

Note: Bacterial Domain file: The input file contains Pfams in the second column separated by semicolon. The required file is automatically generated by the "get_bacterial_domain_structure.py" script, details are described in Step 3.

Note: The header of the output file will be automatically generated as "# Human Protein; Motif; Start; End; Bacterial domain; Bacteria Protein".

Note: If the user is not using Python 3.9, the following error message may appear: ‘*Error: cannot import name 'Mapping' from 'collections*”. Further details and solutions for this issue are provided in the "*Troubleshooting*" section.

5. Quality control of interactions based on target motif location (idr_prediction.py)

This step applies a quality control filter to exclude interactions where the motifs are located outside disordered regions or within globular domains of the host protein, as these interactions are less likely to be biologically significant (Mészáros et al., 2018). The disordered region filter is based on the IUPred pipeline (Mészáros et al., 2018). It takes all predicted interspecies interactions (Step 4), and the human protein FASTA files (Step 2), as an input and results in a filtered dataset based on the above criteria (Figure 5).

Run the script with the necessary command-line arguments:

a. The "**--hmi_prediction**" parameter is a required argument that specifies the path to an existing file containing host-microbe interaction prediction data (Step 4).

b. The "**--fasta_file**" parameter is a required argument that specifies the path to a FASTA file containing human protein sequences (Step 2).

c. The "**--resources**" parameter is a required argument that specifies the path to a folder containing resources needed by the program (see in Notes). This directory should be a valid, accessible path where the user has '*write*' permission.

d. The "**--results**" parameter is a required argument that specifies the path to the folder where the program will save its output results. This directory should be a valid, accessible path where the user has '*write*' permission.

e. The "**--output**" parameter is a required argument that specifies the file name and path where the program will save its outputs.

```
>cd MicrobioLink2/workflow

>python idr_prediction.py --hmi_prediction
"case_study_output/output/HMI/BT_enterocyte_cd_prediction_output_us
ecase.csv" --fasta_file
"case_study_input/input/human_transcriptomics/protein_sequences.fas
ta" --resources "case_study_input/input/" --results
"case_study_output/output/" --output
"case_study_output/output/HMI/IUPred/BT_enterocyte_idr_cd_usecase.c
sv"
```

Note: The script imports an external script named iupred2a.py. It is included in the GitHub repository -workflow/IUPred folder, and-, it should be placed in the same folder as the idr_prediction.py script.

Note: The script needs special data required for the disordered region prediction, such as energy matrices and histograms. These files, located in the Github repository (case_study_input/input/iupred_data), contain data essential for IUPred prediction of disordered regions in protein sequences. Each file in this folder has a specific role in disordered region prediction:

- *anchor2_energy_matrix*: Used by the Anchor2 tool to estimate energy changes in protein interfaces, assisting in identifying protein regions that are likely disordered.
- *anchor2_interface_comp*: A file that complements the energy matrix by containing interface comparison data needed by Anchor2.
- *iupred2_long_energy_matrix and iupred2_short_energy_matrix*: Matrices used by IUPred to predict disordered regions in proteins, tailored for different sequence lengths (long and short).
- *long_histogram and short_histogram*: Histograms providing statistical distributions that IUPred uses to analyze long and short protein sequences.

Without these files, the script cannot execute the disordered region analysis, as IUPred relies on these resources to evaluate disorder in protein structures.

Note: In the HMI prediction file, the delimiter must be “;”. Human UniProt IDs must be in the first column, while motif name, start, and stop information must be in the second, third, and fourth columns. This structure is automatically generated by the DMI.py script, described in Step 4.

Note: When running the IUPred function, the script uses "short" as a predefined method type for IUPred and ANCHOR modeling is set to True. Further details are described in (Mészáros et al., 2018).

Note: The header of the output file will be automatically generated as "# Human Protein; Motif; Start; End; Bacterial domain; Bacterial protein".

Downstream signaling modeling

TIMING: 15-30 mins depending on the size of the data

In this step, the algorithm explores the broader effect of host-microbe protein-protein interactions by modeling the downstream signaling pathways that may be affected. The initial protein interactions identified between microbial and host proteins (Step 5) serve as perturbation points to these pathways, through downstream signaling modeling, the pipeline traces how microbial interactions can influence entire networks of host cellular responses, affecting processes like gene regulation, immune responses, and cellular communication.

This analysis is crucial because it reveals the cascading effects of microbial presence within the host. By mapping these downstream pathways, MicrobioLink provides insights into how microbial interactions might drive complex host responses or contribute to disease mechanisms. The TieDIE (Tied Diffusion Through Interacting Events) tool (Paull et al., 2013) is used in this step to infer these relationships, identifying specific paths that connect bacterial targets to changes in gene expression and cellular functions within the host.

6. Preparing input files for the TieDIE analysis (tiedie_input_processing.py)

TieDIE requires specific input files to map the potential downstream effects of microbial interactions on host cellular signaling pathways. This step involves preparing these files in the correct formats to ensure that the analysis accurately models the downstream cascade initiated by host-microbe interactions. The script processes several files to generate **inputs** for TieDIE (Figure 6):

- **Human transcriptomic data file**: This file provides expression levels for human genes, either as normalized or log2-based values following the exclusion of lowely expressed genes (Step 1), which contextualise the primary knowledge network for the explored tissue (bulk transcriptomics) or cell (single-cell transcriptomics).

- **Endpoint file**: This file specifies target genes within the host that represent the endpoints or primary outputs of interest in the signaling cascade, such as genes involved in immune response or inflammation. These endpoints can be either differentially expressed genes or a list of genes relevant to specific conditions.

- **Host-microbe protein-protein interactions**: This file contains the filtered predictions between human and microbial proteins (Step 6). It serves as the starting point for identifying pathways influenced by microbial interactions.

To execute the script, run the following command in the terminal.

a. The "**--transcriptomics_file**" parameter is a required argument that specifies the path to the file containing transcriptomics data, which includes information on gene expression levels.

b. The "**--value_column**" parameter is a required argument that specifies the column number in the transcriptomics file where expression values are located. User provided *"--value_column*" 2 would refer to the second column.

c. The "--**sep_transcriptomics**" parameter is a required argument that specifies the separator (e.g., ";", "\t", or "|") used in the transcriptomics data file, enabling the program to correctly parse the data.

d. The "**--endpoint_file**" parameter is a required argument specifying the file path to an endpoint file for the analysis. This file defines where the signaling cascade terminates, representing the final targets affected by host-microbe interactions. It should contain data on differentially expressed genes significantly altered by specific conditions or genes that represent the ultimate targets or effects in the signaling pathway.

e. The "**--endpoint_pvalue_column**" parameter is an optional argument that specifies the column number in the endpoint file containing the adjusted p-value or FDR value for differentially expressed genes. This column is typically used to filter genes based on statistical significance, retaining only those with p-values below a certain threshold (e.g., 0.05). If not provided (eg. the endpoint file is not a DEG table but a list of genes of interest), the script will include all genes in the endpoint file.

f. The "**--endpoint_value_column**" parameter is a required argument that specifies the column number in the endpoint file containing the fold change or expression values for (differentially) expressed genes. This data is used in downstream analyses to calculate transcription factor activity.

g. The "**--sep_endpoint**" parameter is a required argument that specifies the separator (e.g., ";", "\t", or "|") used in the endpoint data file, enabling the program to correctly parse the data.

h. The "**--hmi_prediction_file**" parameter is a required argument that specifies the path to an existing file containing filtered host-microbe interaction predictions (Step 5).

i. The "**--output_dir**" parameter is a required argument that specifies the directory where the script will save all its output files. This directory should be a valid, accessible path where the user has ‘*write*’ permission.

j. The "**--upstream_input_filename**" parameter is a required argument that allows the user to specify a custom name for the upstream input file generated by the program. If the user does not specify this argument, the program will save the upstream input file using this default name ("*upstream.input*") in the specified output directory.

k. The "**--downstream_input_filename**" parameter is a required argument that allows the user to specify a custom name for the downstream input file generated by the script. If the user does not specify this argument, the program will save the downstream input file using this default name ("*downstream.input*") in the specified output directory.

l. The "**--pathway_input_filename**" parameter is a required argument that allows the user to specify a custom name for the pathway input file generated by the program. If the user does not specify this argument, the program will save the pathway input file using this default name (*“pathway.sif*") in the specified output directory.

```
>cd MicrobioLink2/workflow

>python tiedie_input_processing.py --transcriptomics_file
“case_study_input/input/human_transcriptomics/enterocyte_colon_CD_z
score.csv” --value_column 2 --sep_transcriptomics ";" --
endpoint_file
“case_study_input/input/human_transcriptomics/Enterocytes
BEST4_degs_fc05.csv” --sep_endpoint "," --hmi_prediction_file
“case_study_output/output/HMI/IUPred/BT_enterocyte_idr_cd_usecase.c
sv” --output_dir “case_study_output/output/” --
upstream_input_filename “usecase_upstream.input” --
downstream_input_filename “usecase_downstream.input” --
pathway_input_filename “usecase_pathway.sif”
```

The **outputs** of this step include:

- **contextualised_regulator-target_network.txt**: Contextualised transcription factor – target gene interaction table derived from CollecTRI highlighting those regulatory interactions where the transcription factor is in the transcriptomics data and the target gene is in the endpoint gene list (details above);
- **upstream.input**: 3-column tab-separated file that describes the gene name, the input heat, and the expected functional effect of the perturbation (+/-));
- **pathway.sif**: 3-column tab-separated file including the <source>, <interaction>, and <target> information;

- **downstream.input**: similar in structure to upstream.input, but the third column indicates the inferred activity of the gene (rather than the expected effect).

Note: Gene names must be in the first column of the transcriptomic file provided.

Note: Python starts to count at 0, therefore the user-provided value column number is automatically decreased by one. Users can count as normal.

Note: HMI file column names should be "#Human Protein" and "Bacterial Protein". This format is automatically achieved by using the output of idr_prediction.py script, described in Step 5.

Note: "*upstream.input*", "*pathway.sif*" and "*downstream.input*" are the default filenames of the outputs, but can be personalized using the necessary command-line arguments.

Note: If the output file is empty after running the script, it may occur if the input files do not have unified identifiers, such as UniProt IDs, across all files; details are described in the "*Troubleshooting*" section.

7. Running the TieDIE analysis (tiedie.py)

TieDIE (Tied Diffusion Through Interacting Events) is a computational tool for network analysis aimed at discovering causal relationships within biological networks (Paull et al., 2013). It takes the upstream, pathway and downstream inputs, described in Step 6, and creates two main output files. The TieDIE network file ("*tiedie.cn.sif*") contains information on source and target nodes, relationship and layer. Additionally, the Cytoscape Linker Heats file ("*heats.NA*") provides information on the network's heat values for linker nodes (Figure 7).

To execute the script, run the following command in the terminal.

a. The "**--upheats**" parameter is a required argument that specifies the path to a file containing upstream heat values (Step 6). This file contains heat values representing the impact of microbial interactions on upstream host proteins, quantified by the number of bacterial proteins targeting each molecule. The file should be formatted with three columns:

    - <gene>: The gene/protein identifier (UniProt or gene symbol);;
    - <input heat>: A numeric value indicating the amount of signal reaching the upstream nodes – number of bacterial proteins targeting the human protein;
    - <sign (+/-)>: A sign indicating whether the interactions' effect is positive (+) or negative (-).

b. The **--down_heats** parameter is a required argument that specifies the path to a file containing downstream heat values (Step 6). This file provides activity data on downstream transcription factors (TFs), showing how microbial interactions may indirectly influence gene regulation. The file should be structured with three columns:

    - <transcription factor>: The TF identifier (UniProt or gene symbol);

    - <activity>: The TF activity is calculated based on the average adjusted expression of its target genes. The adjusted expression values account for whether each TF - target interaction is stimulatory or repressive;
    - <sign (+/-)>: Positive values indicate gene activation by the TF and negative values indicate repression.

c. The **--network parameter** is a required argument that specifies the path to a .sif network file. This file represents a curated pathway network to be used for pathway search and analysis, formatted with specific directional relationships between nodes. The .sif (Simple Interaction Format) file should follow a specific structure with three columns:

    - <source>: The identifier of the source node (UniProt or gene symbol);
    - <interaction>: Specifies the type of relationship or interaction between the nodes;
    - <target>: The identifier of the target node (UniProt or gene symbol).

d. The **--output_folder** parameter is a required argument that specifies the directory path where the program will save all output files.

```
>cd MicrobioLink2/workflow

>python tiedie.py --network
“case_study_output/output/TieDIE/usecase_pathway.sif” --up_heats
“case_study_output/output/TieDIE/usecase_upstream.input” --
down_heats
“case_study_output/output/TieDIE/usecase_downstream.input” --
output_folder “case_study_output/output/”
```

The command to run TieDIE integrates the above inputs and generates several key output files, including a network file (“*tiedie.cn.sif*”) that captures connections from microbial proteins to host TFs. Additionally, TieDIE produces a heat file (“*heats.NA*”), which highlights downstream nodes potentially influenced by microbial interactions.

Note: The code runs several external scripts in the background (kernel.py; kernel_scipy.py; master_reg.py; permute.py; ppr.py; and tiedie_util.py) that are in the Github repository – workflow/TieDie/TieDie – these should be placed in the same folder as tiedie.py on the user’s computer.

Critical: After running tiedie.py you must delete the tiedie_kernel.pkl file which is created automatically and will be found in your script folder. It is important to be able to run the script again without error, details are described in the “*Troubleshooting*” section.

Note: If the output file is empty after running the script, it may occur if the input files do not have unified identifiers, such as UniProt IDs, across all files; details are described in the "*Troubleshooting*" section.

Note: The script results in several additional outcomes, but further steps do not use them directly. These are the following: Node Statistics ("*node.stats*"); Cytoscape Node Types ("*node_types.NA*"); Individual Source Neighborhood Files; Report ("*report.txt*"); Score Distribution ("*score.txt*"); Permuted Scores ("*permuted_scores.txt*"). Further details are available here: https://sysbiowiki.soe.ucsc.edu/tiedie

8. Processing TieDie output files (tiedie_output_processing.py)

After running the TieDIE analysis, the output files require processing to generate interpretable data formats for visualization and downstream analysis. This step combines TieDie outputs ("*tiedie.cn.sif*" and "*heats.NA*" - Step 7), the regulator-target network file ("*contextualised_regulator-target_network.txt*" - Step 6), the filtered HMI file (Step 5), and the endpoint (Figure 8). By processing these files, users can visualize and analyse how microbial interactions influence host cellular processes, making it easier to identify key pathways and regulatory targets affected by microbial activity. The main processed outputs include a network file and a node attribute file, which facilitate pathway visualization in Cytoscape or other network analysis tools.

Run the script with the necessary command-line arguments.

a. The "**--tiedie_file**" parameter is a required argument that specifies the path to the "*tiedie.cn.sif*" file (Step 7). This file captures the entire signaling network, detailing connections from upstream microbial targets through intermediary host proteins to downstream regulatory targets. The default file name is set to "*tiedie.cn.sif*", but the user can specify a different file path or name as needed.

b. The "**--heats_file**" parameter is a required argument that specifies the path to the "*heats.NA*" file (Step 7). This file contains node heat values that indicate the relative influence or activity level of each node in the network. The default file name is set to "*heats.NA*", but the user can specify a different file path or name as needed.

c. The "**--hmi_file**" parameter is a required argument that specifies the path to an existing file containing host-microbe interaction prediction data (Step 5). This file includes predictions between host and microbial proteins, essential for identifying potential pathways impacted by microbial interactions.

d. The "**--tf_tg_file**" parameter is a required argument that specifies the path to the *contextualized_regulatory_network.txt* (Step 6). This file shows how TFs regulate host gene expression, enhancing understanding of microbial impacts on gene expression.

e. The "**--endpoint_file**" parameter is a required argument that specifies the file path to an endpoint file. This file described the target genes of interest, such as genes with differential expression, that are relevant to specific host responses.

f. The "**--endpoint_pvalue_column**" parameter is an optional argument that specifies the column number in the endpoint file containing the adjusted p-value or FDR value for differentially expressed genes. This column is typically used to filter genes based on statistical significance, retaining only those with p-values below a certain threshold (e.g., 0.05). If not provided (eg. the endpoint file is not a DEG table but a list of genes of interest), the script will include all genes in the endpoint file.

g. The "**--endpoint_value_column**" parameter is a required argument that specifies the column number in the endpoint file containing the fold change or expression values for (differentially) expressed genes. This data is commonly used in downstream analyses to assess the direction and magnitude of gene expression.

h. The "**--sep_endpoint**" parameter is a required argument that specifies the separator (e.g., ";", "\t", or "|") used in the endpoint data file, enabling the program to correctly parse the data.

i. The "**--network_output**" parameter is a required argument that specifies the path and filename for the network output file.

j. The "**--node_attr_output**" parameter is a required argument that specifies the path and filename for saving the node attribute output.

```
>cd MicrobioLink2/workflow

>python tiedie_output_processing.py --tiedie_file
“case_study_output/output/TieDIE/tiedie.cn.sif” --heats_file
“case_study_output/output/TieDIE/heats.NA” --hmi_file
“case_study_output/output/HMI/IUPred/BT_enterocyte_idr_cd_usecase.c
sv” --tf_tg_file
“case_study_output/output/TieDIE/contextualised_regulator-
deg_network.txt” --endpoint_file
“case_study_input/input/human_transcriptomics/Enterocytes
BEST4_degs_fc05.csv” --sep_endpoint "," --endpoint_value_column 1 -
-endpoint_pvalue_column 2 --network_output
“case_study_output/output/TieDIE/usecase_final_network.txt” --
node_attr_output
“case_study_output/output/TieDIE/usecase_node_table.txt”
```

Note: The script selects the "# Human Protein" and "Bacterial protein" columns from the IUPred filtered host-microbe interaction data. This format is achieved by using the filtered HMI output, created in Step 5.

Note: Python starts to count at 0, therefore the user-provided value column number is automatically decreased by one. Users can count as normal.

Note: The following error message may appear if the mygene package is not installed properly: ‘AttributeError: partially initialized module 'charset_normalizer' has no attribute 'md__mypyc’’. Further details and solutions for this issue are provided in the "Troubleshooting" section.

Note: If the output file is empty after running the script, it may occur if the input files do not have unified identifiers, such as UniProt IDs, across all files; details are described in the “*Troubleshooting*” section.

Functional Analysis

TIMING: 5-10 mins depending on the size of data

The functional analysis step is an addition to the MicrobioLink pipeline, providing insight into the biological interpretation of the predicted host-microbe interaction network. By performing enrichment analysis on proteins potentially affected by microbial interactions, this step allows researchers to identify key biological processes that are likely influenced by microbial activity, potentially highlighting the molecular mechanisms behind host responses and disease processes.

9. Enrichment analysis (enrichr_id_database_ranking.py)

This step performs functional enrichment analysis using the gget library. It reads background and target gene lists and performs the analysis using the Enrichr database. It generates a bar plot of the top enriched pathways based on either a combined score – natural log of the p-value multiplied by the z-score –, or adjusted p-value and saves the results to an output table and a plot (Figure 9).

Functional enrichment is conducted on two levels—either directly on the human proteins targeted by microbial proteins or on the downstream signaling network inferred by TieDIE. This flexibility allows for a broad or focused approach depending on the study goals.

Run the script with the necessary command-line arguments:

a. The "**--background_gene_list**" parameter is a required argument that specifies the path to a file containing the background gene list (UniProt or gene symbol IDs). This list contains all genes expressed in the dataset, providing a reference set for enrichment analysis. The background set helps define the scope of comparison for identifying significantly enriched pathways or gene sets.

b. The "**--sep**" parameter is a required argument that specifies the separator (e.g., ";", "\t", or "|") used in the background gene file. This allows the program to correctly parse the data, as different files may use different delimiters.

c. The "**--id_background**" parameter is a required argument that specifies the type of identifier (UniProt or genesymbol) used for the background genes in the background gene file.

d. The "**--target_gene_list**" parameter is a required argument that specifies the list of target genes (UniProt or gene symbol IDs). This list includes genes specifically affected by microbial interactions, either as direct microbial targets or as downstream elements in the TieDIE-inferred network.

e. The "**--analysis_level**" parameter is a required argument that specifies the stage at which the enrichment analysis will be performed. The argument expects a choice between two options:

    - "*HMI*": Performs enrichment directly on human proteins that are targeted by bacterial interactions, useful for understanding host-microbe interactions at a high level.

    - "*TieDIE*": Focuses on the downstream signaling network inferred through the TieDIE pathway analysis, highlighting signaling pathways impacted by initial bacterial interactions.

f. The "**--output_image**" parameter is a required argument that specifies the path and filename for saving the output image generated by the enrichment analysis. This image typically visualizes the top enriched pathways, providing an accessible, graphical summary of the most significant pathways affected by the target genes. The file format should generally be suitable for graphical data, such as .png, .jpg, or .pdf, depending on user preference.

g. The "**--output_file**" parameter is a required argument that specifies the path and filename for saving the results of the enrichment analysis in a text format.

h. The "**--database**" parameter is a required argument that specifies the reference database for the enrichment analysis. This parameter enables users to choose a relevant database from which pathways or gene sets are selected to perform the analysis. The parameter supports specific shortcuts for commonly used databases, each tailored to different types of biological questions (see below). The script uses the default database ("*Reactome_2022*") for pathway enrichment analysis, if not provided. However, users can specify other databases depending on their research needs. This parameter expects a string, which can be either:

    Supported shortcuts and their default databases:

    Pathway: “*Reactome_2022* “, “*KEGG_2021_Human*”
    Transcription: “*ChEA_2016*”
    Ontology: “*GO_Biological_Process_2021*”
    Diseases_drugs: “*GWAS_Catalog_2019*”
    Celltypes: “*PanglaoDB_Augmented_2021*”
    Kinase_interactions: “*KEA_2015*”

    or any database listed under Gene-set Library at: https://maayanlab.cloud/Enrichr/#libraries

i. The **--ranking** parameter is a required argument that specifies the metric used to rank pathways in the enrichment analysis plot. If not provided, the script will use "combined_score" as the default ranking metric, which is commonly used method to combine the significance (p-value) and strength (z-score) of enrichment. The argument expects a string, with two options available:

   - "*combined_score*": Ranks pathways by the combined score, calculated as the natural log of the p-value multiplied by the z-score. This score highlights pathways that are both statistically significant and highly enriched.
   - "*adj_p*": Ranks pathways by the adjusted p-value, which is a more conservative approach, focusing strictly on statistical significance after correction for multiple testing.

```
>cd MicrobioLink2/workflow

>python enrichr_id_database_ranking.py --background_gene_list
“case_study_input/input/human_protein/Enterocyte_Manual/enterocyte_
colon_CD_expressed_genes.csv” --sep “;” --id_background
"genesymbol" --target_gene_list
“case_study_output/output/TieDIE/tiedie.cn.sif.txt” --
analysis_level “TieDIE” --output_image
“case_study_output/output/Enrichment_analysis/gget_enrichr_results_
reactome_usecase_tiedie.png” --output_file
“case_study_output/output/Enrichment_analysis/gget_enrichr_results_
reactome_usecase_tiedie.csv” --database “Reactome_2022” --ranking
“combined_score”
```

Optional: This step may be skipped if the user does not want to perform enrichment analysis on the multi-layered network.

Note: The script extracts the first column of the background_genes list, assuming it contains the gene symbol/UniProt IDs.

Note: Functional analysis of the downstream signaling network: The script processes the tiedie.cn.sif file, the output of tiedie.py created in Step 7. This assumes that the target gene IDs are in the first and third columns of each line.

Note: Functional analysis of the bacteria-targeted human proteins: The script processes the filtered host-microbe protein-protein interaction prediction file, the output of idr_prediction.py created in Step 5. This assumes that the target gene IDs are in the first columns of each line.

Note: The default reference database is "Reactome_2022", the most up-to-date version of the database (September, 2024).

Note: The default ranking method is based on "combined_score". Plot can be ranked based on "combined_score" or "adj_p"."

Note: The following error message may appear if the mygene package is not installed properly: 'AttributeError: partially initialized module 'charset_normalizer' has no attribute 'md__mypyc''. Further details and solutions for this issue are provided in the "Troubleshooting" section.

Note: If the output file is empty after running the script, it may occur if the input files do not have unified identifiers, such as UniProt IDs, across all files; details are described in the "*Troubleshooting*" section.

Expected outcomes

We performed a case study focusing on the effects of the probiotic *Bacteroides thetaiotaomicron* (Bt)-derived extracellular vesicles (BEVs) on host cellular signaling in inflammatory bowel disease (IBD). Extracellular vesicles are small membrane-coated structures, secreted by bacteria, that play an important role in intercellular communication, including host-microbe interactions. Previous research has shown that Bt can ameliorate inflammation in mouse models of IBD, making it a promising candidate for studying its underlying molecular mechanisms that may help to identify promising therapeutic targets for IBD.

To explore these effects, we used proteomic data from Bt BEVs (Gul et al., 2022) and human gene expression data from colonic enterocytes of Crohn's disease (CD) patients obtained from public single cell transcriptomic data (https://singlecell.broadinstitute.org; ID: SCP259). The case study input and output files are available in the GitHub repo – "*case_study_input*" and "*case_study_output*" folders. The pipeline generated three main outcomes, each offering insights into different aspects of the host-microbe interaction.

1. **Host-microbe protein-protein interaction network**: The first outcome is a predicted network of protein-protein interactions between bacterial proteins from Bt and human proteins, connected based on DMIs (Figure 10). The interaction network, detailed in Step 2, reveals specific contact points where Bt proteins may influence host proteins, suggesting potential molecular sites where microbial effects on host cellular processes may begin.
2. **Multilayered protein-protein interaction network**: The second outcome is a multilayered network that extends the host-microbe interactions by linking Bt-derived proteins to differential gene expression patterns. This network, detailed in Step 3 and visualized in Cytoscape using the clusterProfiler package, is shown in Figure 11. This outcome reveals how Bt BEVs could impact gene regulation differently in healthy vs inflamed conditions. This allows researchers to identify pathways and cellular functions that are probably influenced by microbes, helping to discover potential therapeutic targets.
3. **Enrichment analysis of affected pathways**: The third outcome is a bar plot of the top 20 enriched pathways influenced by Bt-BEV interactions, based on combined scores, detailed in Step 9. This plot, shown in Figure 12, is automatically generated by the pipeline running the enrichment analysis. The output offers insights into mechanisms through which Bt BEVs could potentially mitigate inflammation or promote immune balance in IBD.

Limitations

While MicrobioLink offers a robust framework to integrate human transcriptomics and bacterial proteomics data for predicting host-microbe interactions, several limitations should be considered when interpreting the results. These limitations show the areas where future improvements could enhance the accuracy and applicability of the pipeline.

- **Dependence on pre-processed input data**: Microbiolink requires pre-processed multi-omic data, such as normalized gene count matrices for human data and UniProt IDs for bacterial proteomics, to ensure compatibility. When working with similar datasets, researchers may need to format their multi-omic data according to the specific requirements outlined in the case study.
- **Reliance on domain-motif interactions from the ELM database**: The pipeline uses domain information from the Pfam database and DMIs from the ELM database, which lack information about bacteria-specific domain interactions. This limitation could lead to missed predictions of interaction sites. To address this, the integration of the AlphaFold tool, a deep learning model for predicting protein structures, is currently under development potentially improving prediction accuracy in future versions.
- **Dependence on public databases for functional analysis**: While databases, like Reactome or KEGG, provide the advantage of quickly associating genes with functional terms, they also carry the risk if certain pathways are overrepresented or underrepresented. To mitigate this bias, users are encouraged to integrate additional annotation sources, reducing dependence on a single database and enhancing the robustness of the results.
- **Limited protein-protein interactions**: The current version of MicrobioLink focuses on protein-protein interactions. However, many host-microbe interactions are mediated by metabolites, which play crucial roles in communication and signaling between the host and microbial communities (Nicholson et al., 2012). The integration of metabolite data analysis into the pipeline is currently under development.

These limitations provide context for interpreting the results generated by MicrobioLink. While the pipeline is a versatile tool for studying host-microbe interactions, addressing these limitations through future developments could enhance its accuracy and broader applicability.

Troubleshooting

Problem

Some software or packages are unavailable, leading to errors.

Potential solution

This error may arise at any step (Step 1-9), as each script relies external package dependencies. To resolve this issue, ensure all steps are executed within the designated computing environment, which should be correctly set up according to the instructions in the "*Before you begin*" section.

Verifying that the required packages are installed and accessible in this environment can prevent these availability issues.

Problem

One or multiple paths to the directories, source files, or the output folder the user defined do not exist – *"python: can't open file 'path/to/file'': [Errno 2] No such file or directory"*

Potential solution

This error may occur at any step (Step 1-9). Verify that you have provided the paths to the folders where your source files are located, respectively, without including the file names at the end of the path. Ensure that the output directory you defined exists. For more detailed information, refer to the GitHub documentation (Key resources table) or the functions' documentation in Python.

Problem

Missing parameter definition:
Error: usage:script.py [-h] -p1 PARAMETER1 -p2 PARAMETER2 -p3 PARAMETER3
script.py: error: the following arguments are required: -p3/--parameter3

Potential solution

This error may arise at any step (Step 1-9) when running the scripts from the Terminal/Command Line if a required parameter is not defined. The help message, which can be accessed by running *script.py -h*, provides additional guidance on the parameters required for each step.

Problem

Specific error message while running TieDie in Step 7 – '*Error: the universe of gene/node labels in the network file doesn't match the supplied kernel file*!'

Potential solution

This error typically occurs when there is a mismatch between the gene or node labels in the network file and the kernel file used in the TieDIE analysis, often due to a prior unsuccessful run or an outdated intermediate file. To resolve this issue, delete the .pkl file in the workflow/TieDie/TieDie folder. This file is a cached intermediate output that stores network data, and deleting it allows TieDIE to generate a new version aligned with the current network and kernel files.

Problem

Error message: cannot import name 'Mapping' from 'collections

Potential solution

This error is typically caused by an incompatibility issue with the *collections* Python package and may occur in Step 4 during the *in silico* host-microbe interaction prediction. To resolve this issue, use Python 3.9, as the *collections* package used by the pyfasta library is compatible with this version.

Problem

AttributeError: partially initialized module 'charset_normalizer' has no attribute 'md__mypyc'

Potential Solution

This error is typically caused by an incorrect installation of the *mygene* Python package, and it can occur in steps that depend on this package (Step 2, Step 8, and Step 9). Reinstalling the *charset-normalizer* package, a dependency of *mygene*, usually resolves the issue. To fix this, open the Terminal/Command Line and run the following command:

```
>pip install --force-reinstall charset-normalizer==3.1.0
```

Problem

The output file is empty.

Potential solution

This issue often arises from incorrect input formatting, mismatched molecular IDs or a lack of predicted interactions. Follow these steps to troubleshoot:

- **Incorrect input formatting:** This error may arise at any step (Step 1-9), verify that the input files follow the expected structure and format as described in each step, with gene identifiers in the correct columns, proper separators, and accurate file paths.
- **ID issues:** This issue may occur in Step 6-9; when merging multiple input files, it is essential that each file uses the same type of identifiers (e.g., UniProt IDs) for successful integration. Although the scripts are designed to be flexible, they require consistency in the identifier format across all input files to ensure proper alignment and merging of data. Before running the pipeline, verify that each input file uses a uniform ID format. If necessary, convert IDs in one or more input files to match the format used by the others. You can use tools like UniProt's ID mapping service to standardize identifiers.
- **No significant interactions detected**: If the input data is formatted correctly, an empty output may indicate that no significant interactions met the criteria for inclusion in the output in Step 4 or Step 5. To confirm, consider setting lower thresholds or checking intermediate files to ensure that earlier steps generated expected results.

Article info

Resource availability

Lead Contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Tamas Korcsmaros (t.korcsmaros@imperial.ac.uk).

Technical Contact

Questions about the technical specifics of performing the protocol should be directed to and will be answered by the technical contact, Lejla Gul (l.potari-gul@imperial.ac.uk).

Materials Availability

The case study used human single-cell transcriptomics published by (Kong et al., 2023), while the bacterial proteins derived from proteomic data published in (Gul et al., 2022).

Data and Code Availability

The code generated during this study is available on GitHub (https://github.com/korcsmarosgroup/MicrobioLink2/tree/main/workflow). The MicrobioLink tool source code was previously published (Andrighetti et al., 2020) and is available alongside extensive documentation on GitHub.

Acknowledgments

The work of L.G., B.B. and T.K. was supported by the UKRI BBSRC Institute Strategic Programme Food Microbiome and Health BB/X011054/1 and its constituent project BBS/E/F/000PR13631. T.K was also supported by the NIHR Imperial Biomedical Research Centre Organoid Facility. The views expressed are those of the authors and not necessarily those of the NIHR or the UK Department of Health and Social Care . AJ. E. is supported by the Semmelweis Science and Innovation Fund, Semmelweis Young Researchers Research Grant (STIA-MEC 2022), the Federation of European Microbiological Societies, FEMS Research and Training Grant (ID 4695) and by the Kerpel-Fronius Ödön Talent Program, Kerpel Scholarship. D.M. acknowledges financial support from Imperial College London through an Imperial College Research Fellowship grant award.

Author contributions

Script implementation: L. G., AJ. E., T. T., D.M, M.O, B. B., M. M.; Pipeline testing: L. G., AJ. E., T. T., M.O, E. W.; Case Study implementation: AJ.E; Manuscript writing: L. G., AJ. E.; Reviewing: T. K.; Supervision: T. K. All authors approved the final version.

Declaration of interests

The authors declare no competing interests.

References

Andrighetti, T., Bohar, B., Lemke, N., Sudhakar, P. and Korcsmaros, T. (2020). MicrobioLink: An Integrated Computational Pipeline to Infer Functional Effects of Microbiome-Host Interactions. *Cells* **9**,.

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**, 1422–1423.

Gul, L., Modos, D., Fonseca, S., Madgwick, M., Thomas, J. P., Sudhakar, P., Booth, C., Stentz, R., Carding, S. R. and Korcsmaros, T. (2022). Extracellular vesicles produced by the human commensal gut bacterium Bacteroides thetaiotaomicron affect host immune pathways in a cell-type specific manner that are altered in inflammatory bowel disease. *J. Extracell. Vesicles* **11**, e12189.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with NumPy. *Nature* **585**, 357–362.

Hart, T., Komori, H. K., LaMere, S., Podshivalova, K. and Salomon, D. R. (2013). Finding the active genes in deep RNA-seq gene expression studies. *BMC Genomics* **14**, 778.

Kong, L., Pokatayev, V., Lefkovith, A., Carter, G. T., Creasey, E. A., Krishna, C., Subramanian, S., Kochar, B., Ashenberg, O., Lau, H., et al. (2023). The landscape of immune dysregulation in Crohn’s disease revealed through single-cell transcriptomic profiling in the ileum and colon. *Immunity* **56**, 444-458.e5.

Luebbert, L. and Pachter, L. (2023). Efficient querying of genomic reference databases with gget. *Bioinformatics* **39**,.

Mészáros, B., Erdos, G. and Dosztányi, Z. (2018). IUPred2A: context-dependent prediction of protein disorder as a function of redox state and protein binding. *Nucleic Acids Res.* **46**, W329–W337.

Müller-Dott, S., Tsirvouli, E., Vazquez, M., Ramirez Flores, R. O., Badia-I-Mompel, P., Fallegger, R., Türei, D., Lægreid, A. and Saez-Rodriguez, J. (2023). Expanding the coverage of regulons from high-confidence prior knowledge for accurate estimation of transcription factor activities. *Nucleic Acids Res.* **51**, 10934–10949.

Nicholson, J. K., Holmes, E., Kinross, J., Burcelin, R., Gibson, G., Jia, W. and Pettersson, S. (2012). Host-gut microbiota metabolic interactions. *Science* **336**, 1262–1267.

Paull, E. O., Carlin, D. E., Niepel, M., Sorger, P. K., Haussler, D. and Stuart, J. M. (2013). Discovering causal pathways linking genomic events to transcriptional states using Tied Diffusion Through Interacting Events (TieDIE). *Bioinformatics* **29**, 2757–2764.

Poletti, M., Treveil, A., Csabai, L., Gul, L., Modos, D., Madgwick, M., Olbei, M., Bohar, B., Valdeolivas, A., Turei, D., et al. (2022). Mapping the epithelial-immune cell interactome upon infection in the gut and the upper airways. *NPJ Syst. Biol. Appl.* **8**, 15.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N.,

Schwikowski, B. and Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504.

Türei, D., Valdeolivas, A., Gul, L., Palacio-Escat, N., Klein, M., Ivanova, O., Ölbei, M., Gábor, A., Theis, F., Módos, D., et al. (2021). Integrated intra- and intercellular signaling knowledge for multicellular omics analysis. *Mol. Syst. Biol.* **17**,.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272.

Wu, C., Macleod, I. and Su, A. I. (2013). BioGPS and MyGene.info: organizing online, gene-centric information. *Nucleic Acids Res.* **41**, D561-5.

Figure legends

Figure 1: Transformation of normalized gene counts to Log2-based values with Z-score filtering.

Figure 2: Downloading FASTA sequences for expressed genes.

Figure 3: Retrieving domain structures for bacterial proteins.

Figure 4: Integrating bacterial protein domains with human protein motifs to predict host-microbe protein-protein interactions.

Figure 5: Applying IUPred disordered region filter to refine host-microbe protein-protein interactions.

Figure 6: Integrating transcriptomic data with the predicted host-microbe interactions to prepare the input files for the TieDIE analysis.

Figure 7: Running TieDIE to identify the signaling network potentially influenced by bacteria and quantify signal propagation through nodes in the network.

Figure 8: Preparing annotated interaction and node files for network visualisation in Cytoscape.

Figure 9: Functional enrichment analysing using the enrichR Python package.

Figure 10. Predicted protein-protein interactions between *B. thetaiotaomicron*-derived extracellular vesicles and colon enterocyte cells in Crohn's disease.

Figure 11. The potential downstream effect of proteins of *B. thetaiotaomicron*-derived extracellular vesicles on Crohn's disease patient-derived colon enterocytes. 1. Bacterial proteins 2. Human membrane proteins 3. Intermediate protein-protein interaction signaling network 4. Transcription factors 5. Differentially expressed genes.

Figure 12. The top 20 enriched pathways based on combined scores for B. *thetaiotaomicron* and colon enterocyte cells in Crohn's disease based on the Reactome database.