Protocol for MicrobioLink: a tool for predicting the upstream and downstream effect of microbes in host cells and tissues

Lejla Gul[1,4*], Anna Julia Elias[2], Tanvi Tambaku[1], Balazs Bohar[1], Matthew Madgwick[3], Tamas Korcsmaros[1,5**]

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

Summary

Analyzing interspecies interactions between the host and microbes is crucial for understanding how alterations in the microbiota composition can disrupt host homeostatic processes, potentially leading to diseases. Here, we present the MicrobioLink pipeline, an *in silico* prediction tool of host-microbe protein-protein interactions and their downstream effect on host cellular processes.

For complete details on the use and execution of this protocol, please refer to (Andrighetti et al. 2020) and (Treveil et al. 2021).

Highlights

- A protocol for the MicrobioLink pipeline, a computational tool that analyzes the impact of host-microbe interaction on downstream signaling in human cells and tissues
- *In silico* structural-composition-based protein-protein interaction prediction using experimentally verified domain-motif interactions
- Mapping the cellular signaling network using transcriptomic data to follow the signal from the bacteria-targeted membrane proteins till the (differentially) 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.
This section includes recommendations for the hardware specifications, and instructions for the pipeline localisation, the environmental setup, and downloading the pipeline.

Downloading multi-omic data for the case study

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

MicrobioLink requires human transcriptomic data in a normalized gene count matrix format and bacterial proteomics/metaproteomics describing UniProt IDs (details in the Step-by-step guideline). 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). The GitHub folder (https://github.com/korcsmarosgroup/MicrobioLink2) includes all input data for the protocol study.

Downloading and installing the pipeline and the required software

TIMING: 1-2 h

1. Download Anaconda - details are described here: https://docs.anaconda.com/free/anaconda/install/index.html

2. Within a working directory, download the pipeline from GitHub: https://github.com/korcsmarosgroup/MicrobioLink2 by downloading the zip archive and then extracting it, or by using the git-clone command:

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

3. To use the pipeline, we recommend to create a conda environment, thus, before the environment initiation, it is essential to have Conda installed on the system (details in Step 1).

4. Create a conda virtual environment with the required packages and activate that environment (more details are available here: https://conda.io/projects/conda/en/latest/commands/create.html)

   ```
   >cd MicrobioLink2

   >conda create --name microbiolink

   >conda activate microbiolink
   ```

5. Open Anaconda terminal by going to the "Environments" – select the appropriate environment and "Open Terminal"

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	http://conference.scipy.org.s3-website-us-east-1.amazonaws.com/proceedings/scipy2010/pdfs/mckinney.pdf	https://pandas.pydata.org; RRID:SCR_018214
numpy	(Harris et al. 2020)	http://www.numpy.org; RRID:SCR_008633
scipy	(Virtanen et al. 2020)	http://www.scipy.org/; RRID:SCR_008058
mygene	(Wu et al. 2013)	RRID:SCR_018660
omnipath	(Türei et al. 2021)	https://omnipathdb.org/
gget	(Luebbert and Pachter 2023)	https://github.com/pachterlab/gget
pyfasta	https://pypi.org/project/pyfasta/0.2.9/	https://github.com/brentp/pyfasta

biopython	(Cock et al. 2009)	http://biopython.org; RRID:SCR_007173
Python v3.8 or later version	(Python Software Foundation, 2016)	https://www.python.org/; RRID:SCR_008394
Cytoscape v3.10.1	Shannon et al.13	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. The required=True parameter indicates that the argument is mandatory. The exact way of executing each script will be described below.

Preparing input files for the MicrobioLink pipeline

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

This step involves preparing the necessary input files for predicting interactions between human and microbial proteins based on domain-motif interactions.

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

This is an optional step before you download the human protein sequences. This script takes gene expression data (normalized average expression count matrices) as input, 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. "-i", "--input_file", required=True, help="Input CSV file with gene expression data."
b. "-o", "--output_file", required=True, help="Path to the output CSV file for filtered results."
c. "-zscore", "--zscore", required=False, default = -3, help="Z-score cut-off to filter lowly expressed genes"

```
>cd MicrobioLink2/workflow

>python z-score_filter_terminal.py -i “path_to_input_file.csv” -o “path_to_output_file.csv” -zscore <zscore_value>
```

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 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 script retrieves protein sequences based on a gene/protein list from a file, with options for specifying identifiers (gene symbol or Uniprot ID), and filtering for membrane-based or secreted proteins. It uses external libraries (mygene, omnipath) to gather gene and protein information.

Run the script with the necessary command-line arguments:

a. “-genes”,”--gene_expression”, required=True, type=str, help=”Path to the transcriptomics data”
b. “-id”,"--id_type", required=True, choices=["genesymbol", "uniprot"], help="Type of gene identifier (genesymbol or uniprot)"
c. "-s","--sep", required=True, help="Field separator in the protein list file"
d. "-lfl", "--location_filter_list", required=False, nargs="+", default=None, help="Location filter list (options:plasma_membrane_transmembrane and/or plasma_membrane_peripheral and/or secreted), (required format: without "", separated by spaces)
e. "-of", "--output_folder", required=True, default=".", help="Output folder for result files"
f. "-oseq", "--output_sequences", required=True, default="protein_sequences.fasta", help="Output file for protein sequences"

```
>cd MicrobioLink2/workflow

>python get_human_fasta.py -genes “path_to_gene_list_file” -id "uniprot" or "genesymbol" -s "separator_of_your_file" -lfl <list of your location filters> -o “path_to_output_file.fasta”
```

Note: The ID (Uniprot or Gene symbol) 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"

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

This script downloads protein information, including the corresponding domain structures, from the UniProt database based on provided UniProt or UniProt Proteome (UP) IDs. It offers two options: downloading the entire proteome or downloading a list of proteins with specific UniProt IDs.

Run the script with the necessary command-line arguments:

a. "--id_list", required=True, help="Path to an existing FILE describing Uniprot or UP IDs"
b. "--sep", required=True, help="Field separator"
c. "--id_type", required=True, help="Type of ID - Uniprot or UniprotProteome (UP)", choices=["Uniprot","UP"]
d. "--id_column",required=True, help="Column number for proteome/protein ID", type=int
e. "--output", required=True, help="Path to the output file"

```
>cd MicrobioLink2/workflow

>python download_bacterial_proteins.py --id_list "path_to_your_id_file" --sep
"separator_of_your_id_file" --id_type "UP" or "Uniprot" --id_column
your_column_number --output "path_to_your_output"
```

Note: Python starts to count by 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 the protein-protein interaction prediction based on human and microbial domain-motif interactions. Additionally, it involves a quality control sub-step, excluding those human targets, where the motif is located outside the disordered region minimizing the number of false positive interactions.

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

The script takes input files containing data on human proteins (Step 2), motif regular expression (regex) patterns and motif-domain interactions from the Eukaryotic Linear Motif (ELM) database, and bacterial protein domains (Step 3). It processes this data to predict interactions between human and microbial proteins based on shared motifs and protein domains, writing the results to an output file. The script utilizes external libraries and modules (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. "-fasta", "--fasta_file", required=True, help="Path to human protein FASTA file"
b. "-motif", "--elm_regex_file", required=True, help="Path to ELM regex file" "-interaction"
c. "--motif_domain_file", required=True, help="Path to motif-domain interaction file"
d. "-domain", "--bacterial_domain_file", required=True, help="Path to bacterial protein domain file"
e. "-o", "--output_file", required=True, help="Path to output file"

```
>cd MicrobioLink2/workflow

>python DMI.py -genes “path_to_human.fasta” -motif “path_to_elm_regex.tsv”
-interaction “path_to_motif_domain.tsv” -domain “path_to_bacterial_domain.tsv” -o
“path_to_output.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 repo (resources folder) 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 repo (resources folder) 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. (see above in Step 3)

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

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

This step focuses on ensuring the quality of predicted PPIs by excluding those human targets, where the motif is located outside disordered regions or within globular domains (Mészáros et al. 2018). The quality control step contributes to reducing the number of false positive interactions The disordered region filter is based on the IUPred pipeline (Mészáros et al. 2018). It takes the output of DMI.py (Step 4) and the human protein FASTA file (Step 2) as an input and results in a filtered dataset based on the above criteria.

Run the script with the necessary command-line arguments:

a. "--hmi_prediction", required=True, help="Path to an existing FILE"
b. "--fasta_file", required=True, help="Path to an existing FILE"
c. "--resources", required=True, help="Path to resources FILES"
d. "--results", required=True, help="Path to results"
e. "--output", required=True, help="Path to output"

```
>cd MicrobioLink2/workflow

>python idr_prediction.py --hmi_prediction “path_to_hmi_prediction_file.csv” --fasta_file “path_to_fasta_file.fasta” --resources “path_to_resources_folder” --results “path_to_results_folder” --output “output_file.csv”
```

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

Note: The script needs special data needed for the disordered region prediction. These files are included in the Github repo (resource/iupred_data folder)

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 using the output of the DMI.py script (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 (details 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, we use various network resources and network propagation tools to analyze the effect of the interactions between microbial and host proteins on other host processes further downstream. For this purpose, the TieDIE tool (Paull et al. 2013), built into the pipeline, is used to infer the subnetwork that connects the host membrane-based proteins with the differentially expressed genes. The second part of the pipeline consists of three main steps: preparing the input files, carrying out the network propagation analysis, and processing the output files.

6. Preparing input files (tiedie_input_processing.py)

This script prepares the input files in the required format for the TieDIE tool. It takes the human transcriptomic data file (optionally filtered based on Z-score in Step 1), the filtered HMI prediction file (Step 5) and the endpoint file (containing the differentially expressed genes) as inputs and creates three essential files for the next step. Upstream.input (3-column tab-separated format contains a column for the gene name, the input heat, and the expected functional effect of the perturbation (+/-)), pathway.sif (3-column tab-separated <source> interaction> <target>) and downstream.input (the format is the same as the upstream.input file, but the third column refers to the inferred activity) (Details in: (Paull et al. 2013)) are created. Additionally it prepares the regulator-target network file, which will be used for TieDIE output processing (see below at Step 8). The script uses external libraries and databases (OmniPath (Türei et al. 2021) and CollecTri (Müller-Dott et al. 2023)) to gain information on interaction data, including protein-protein interactions and transcription factor-target gene interactions.

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

a. "--transcriptomics_file", required=True, help="Path to transcriptomics file"
b. "--value_column", required=True, help="Column number for expression values in the transcriptomic file", action="store", type=int
c. "--sep_transcriptomics", required=True, help="Separator for the transcriptomic file"
d. "--endpoint_file", required=True, help="Path to endpoint file"
e. "--endpoint_pvalue_column", required=False, help="Column number for the adjusted p-value/FDR value in the differentially expressed gene file", type=int
f. "--endpoint_value_column", required=False, help="Column number for log2FC/Expression value in the endpoint file", type=int
g. "--sep_endpoint", required=True, help="Separator for the endpoint file"
h. "--hmi_prediction_file", required=True, help="Path to hmi prediction output file"
i. "--output_dir", required=True, help="Path to the output directory (resource)"
j. "--upstream_input_filename", required=True, type=str, default="upstream.input", help="Custom name for upstream output file"

k. "--downstream_input_filename", required=True, type=str, default="downstream.input", help="Custom name for downstream output file"
l. "--pathway_input_filename", required=True, type=str, default="pathway.sif", help="Custom name for pathway output file"

```
>cd MicrobioLink2/workflow

>python tiedie_input_processing.py --transcriptomics_file “path_to_transcriptomics_file” --value_column X --sep_transcriptomics "sep" --endpoint_file “path_to_endpoint_file” --sep_endpoint "sep" --hmi_prediction_file “path_to_hmi_prediction_file.csv” --output_dir “path_to_output_folder”
```

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

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 (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: The script results in two additional outputs (“contextualised_regulators_of_targets.txt” and “contextualised_regulator-target_network.txt”). The latter is used for processing the TieDIE output files (see Step 8.).

7. Running the TieDIE analysis (tiedie.py)

TieDIE (Tied Diffusion Through Interacting Events) is a tool for network analysis aimed at discovering causal relationships within biological networks (details in: (Paull et al. 2013)). It takes the upstream, pathway and downstream inputs (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.

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

a. "-u","--up_heats", required=True, default=None, help="File with upstream heats: <gene> <input heat (0-100)> <sign (+/-)>"
b. "-d","--down_heats", required=True, type="string", default=None, help="File with downstream heats: <gene> <input heat (0-100)> <sign (+/-)>"
c. "-n","--network", required=True, default=None, help=".sif network file for the curated pathway to search. <source> <(-a>,-a|,-t>,-t|,-component>)> <target>"
d. "-o", "--output folder", required=True, help=”Path to the output folder”

```
>cd MicrobioLink2/workflow

>python tiedie.py --network “path_to_network_file.sif” --up_heats
“path_to_upstream_heats.txt” --down_heats “path_to_downstream_heats.txt”
--output_folder “path_to_output_directory”
```

Note: The code runs several external scripts (kernel.py; kernel_scipy.py; master_reg.py; permute.py; ppr.py; and tiedie_util.py) that are in the Github repo (workflow/TieDie/TieDie) - these should be placed in the same folder as tiedie.py.

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 in 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 Output Files (tiedie_output_processing.py)

The script combines TieDie outputs (tiedie.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 file of differentially expressed genes into a network table and a node attribute table for further visualization purposes.

Run the script with the necessary command-line arguments.

a. "--tiedie_file", required=True, type=str, default="tiedie.cn.sif", help="Path to tiedie.cn.sif file"
b. "--heats_file", required=True, type=str, default="heats.NA", help="Path to heats.NA file"
c. "--hmi_file", required=True, type=str, help="Path to the host-microbe interaction file"
d. "--tf_tg_file", required=True, type=str, help="Path to contextualized regulatory network file"
e. "--endpoint_file", required=True, type=str, help="Path to the endpoint gene list/DEG file"
f. "--endpoint_pvalue_column", required=False, help="Column number for the adjusted p-value/FDR value in the differentially expressed gene file", type=int
g. "--endpoint_value_column", required=False, help="Column number for log2FC/Expression value in the endpoint file", type=int
h. "--sep_endpoint", required=True, help="Separator for the endpoint file", required=True
i. "--network_output", required=True, type=str, help="Output path and name for the network file"
j. "--node_attr_output", required=True, type=str, help="Output path and name for the node attribute file"

```
>cd MicrobioLink2/workflow

>python tiedie_output_processing.py --tiedie_file “path_to_tiedie.cn.sif” --heats_file
“path_to_heats.NA” --hmi_file “path_to_hmi_file.csv” --tf_tg_file “path_to_tf_tg_file.txt”
--endpoint_file “path_to_endpoint_file” --sep_endpoint "," --endpoint_value_column X
--endpoint_pvalue_column Y --network_output “path_to_network_output_file”
--node_attr_output “path_to_node_attr_output_file”
```

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 (Step 5).

Functional Analysis (Optional)

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

This step involves an optional functional enrichment analysis.

9. Enrichment analysis (enrichr_id_database_ranking.py)

This is an optional analysis performing 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 combined score or adjusted p-value and saves the results to an output table and a plot. It offers two analysis level options (HMI and TieDIE).

Run the script with the necessary command-line arguments:

a. "--background_gene_list", required=True, help="Path to the background gene file (Gene expression with/without Z-score filter.")
b. "--sep", required=True, help="Separator of the background gene file."
c. "--id_background", required=True, choices=["genesymbol", "uniprot"], help="ID type of background genes."
d. "--target_gene_list", required=True, help="Path to the target gene file."
e. "--analysis_level", required=True, help="The level in the process where you perform enrichment analysis", choices=["HMI", "TieDIE"]
f. "--output_image", required=True, help="Path to the output image."
g. "--output_file", required=True, help="Path to the output file."
h. "--database", required=True, default="Reactome_2022", help="Database to use as a reference for the enrichment analysis."

Supported shortcuts (and their default database):

"pathway" (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. "--ranking", required=True, default="combined_score", help="Plot will be ranked based on \"combined_score\" or \"adj_p\"."

```
>cd MicrobioLink2/workflow

>python enrichr_id_database_ranking.py --background_gene_list
“path_to_background_genes.csv”  --sep “;" --id_background "genesymbol" or “uniprot”
--target_gene_list “path_to_target_genes.csv” --analysis_level HMI or TieDIE
--output_image “path_to_output.png” --output_file “path_to_output.csv” --database
Reactome_2022 --ranking  “combined_score” or “adj_p”
```

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 (TieDIE analysis level): The script processes the tiedie.cn.sif file (output of tiedie.py - 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 (HMI analysis level): The script processes the filtered host-microbe PPI prediction file (output of idr_prediction.py - 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".

Note: The default ranking method is based on "combined_score". Plot can be ranked based on "combined_score" or "adj_p"." (combined score: natural log of the p-value multiplied by the z-score)

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 located outside cells that play an important role in intercellular communication, including host-microbe interactions. Based on the results of previous research on Bt's ability to ameliorate inflammation in mouse models of IBD, a better understanding of its underlying molecular mechanisms may help to identify promising therapeutic targets for IBD.
The proteomic data from Bt BEVs were gained from this paper (Gul et al. 2022). We used 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 input and output files are in the GitHub repo (case_study_input/case_study_output).

1. The first main outcome of the pipeline is a host-microbe protein-protein interaction network (Step 2). Figure 1. shows an example where bacterial proteins are connected to human proteins predicted by domain-motif interactions representing the interaction between them. The network was generated with the Cytoscape tool (Shannon et al. 2003).

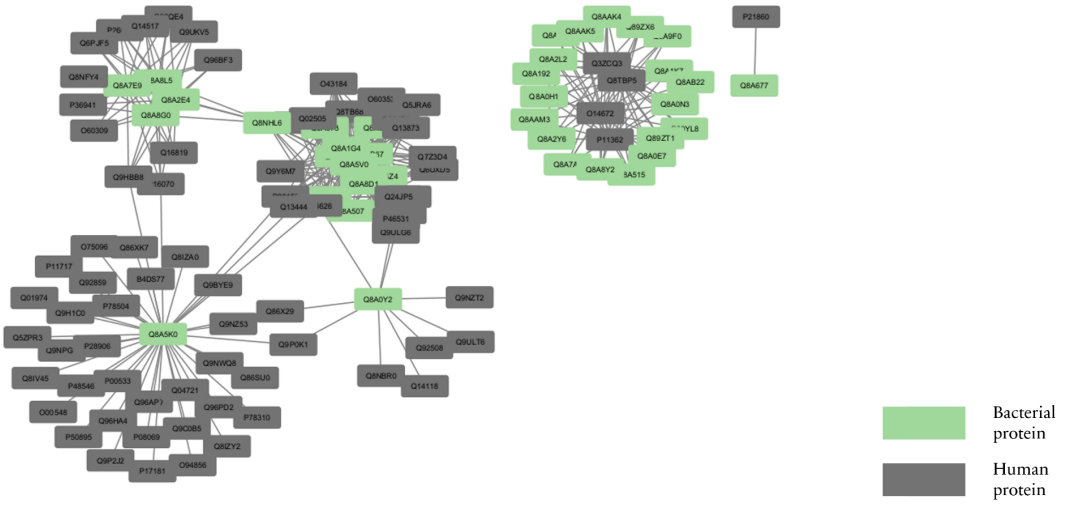

Figure 1. Predicted protein-protein interactions between B. *thetaiotaomicron* and colon enterocyte cells in Crohn's disease.

2. The second outcome of the pipeline is a multilayered protein-protein interaction network (Step 3). Figure 2. shows an example for the multilayered network describing the impact of bacterial proteins on differential gene expression comparing the healthy and inflamed conditions. The network was visualized in the Cytoscape tool using the clusterProfiler package.

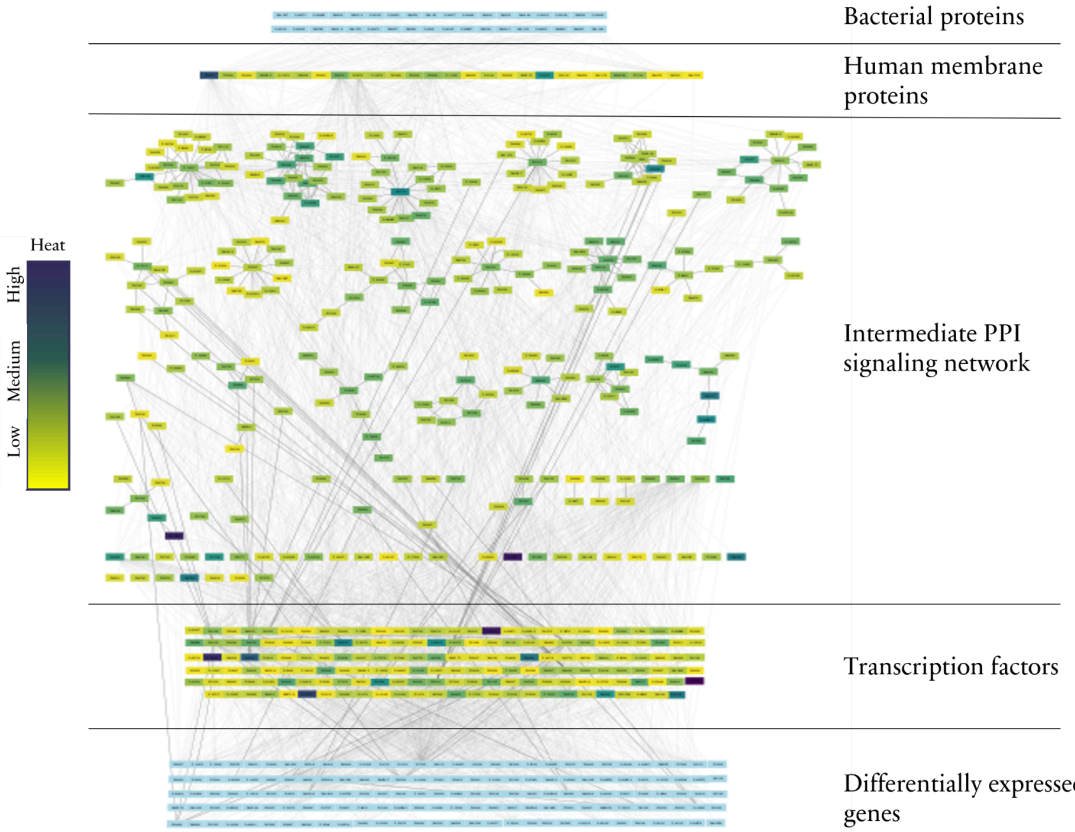

Figure 2. The potential downstream effect of proteins of B. *thetaiotaomicron*-derived extracellular vesicles on Crohn's diseased patient-derived colon enterocytes.

3. The third outcome of the pipeline is a bar plot showing the results of the enrichment analysis with top 20 enriched pathways based on the combined score (Step 9). It can be used to analyze the direct effect of microbial proteins on host targets and also on the downstream signaling network to highlight the pathways that mediate the impact of bacteria. Figure 3. shows the result of the enrichment analysis on the downstream signaling network. This plot is automatically generated by the pipeline running the optional enrichment analysis.

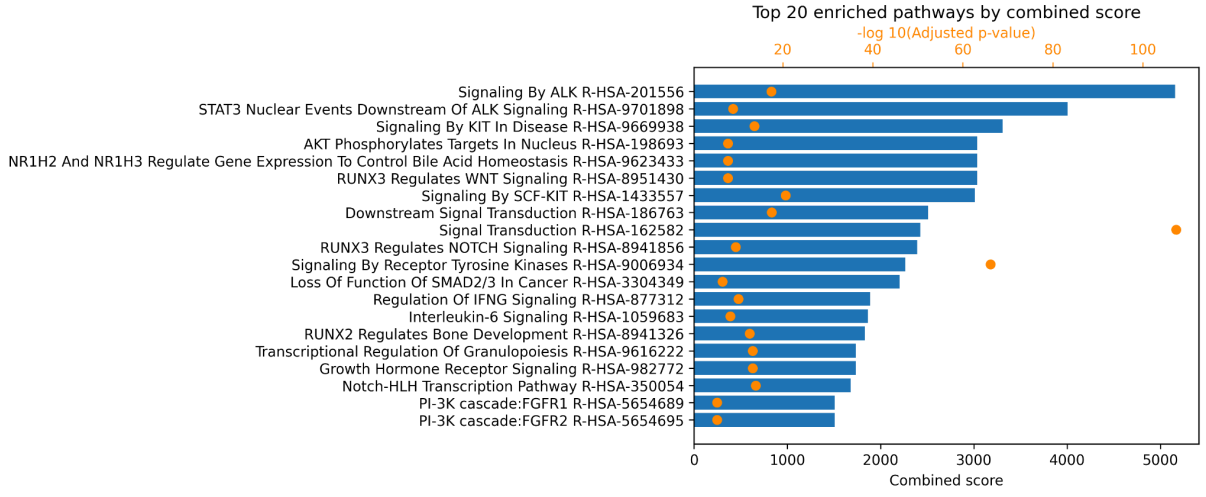

Figure 3. The top 20 enriched pathways based on combined scores for B. *thetaiotaomicron* and colon enterocyte cells in Crohn's disease based on the Reactome database.

Limitations

MicrobioLink accepts processed human transcriptomics and bacterial proteomics files in a specific format. When working with analogous datasets, researchers may need to prepare their multi-omic files according to the case study. Additionally, for the host-microbe protein-protein interaction prediction the pipeline uses domain information from the Pfam database, and domain-motif interactions from the ELM database that lacks bacteria-specific domain interactions. Furthermore, for functional analysis, MicrobioLink is relying on public databases. While these databases offer the great advantage of quickly linking genes with functional terms, they also carry the risk of oversimplification and potential misinterpretation.

Troubleshooting

Problem

Some software or packages are unavailable.

Potential solution

Ensure every step is executed within the designated computing environment, which should be set up correctly as outlined in "Before you begin".

Problem

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

Potential solution

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 (see Key resources table) or the functions' documentation in Python.

Problem

You need to install Cytoscape for network visualization.

Potential solution

Download and install Cytoscape version 3.10.1: https://cytoscape.org/.

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

If a required parameter is not defined, the help message provides further instructions.

Problem

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

Potential solution

Delete the .pkl file in the workflow/TieDie/TieDie folder.