# mBONITA tutorial

## Requirements

The *mBONITA* tool is written in Python3 and C. I strongly recommend that mBONITA be run on a computing cluster such as the University of Rochester's BlueHive, and that jobs are submitted using a scheduler such as SLURM. Dependencies are listed in the conda environment file (SPECIFY FILENAME HERE).

**Minor caveat -** *mBONITA* is not a Python package like numpy or scipy, which allow users to import individual functions and (re)use them in custom code. mBONITA is an all-in-one pipeline that doesn't allow function import or much customization beyond the pre-specified parameters. I welcome advanced users to modify code and submit pull requests, but this is beyond what most users will need.

*mBONITA* requires the following inputs (Step 0):

- A pre-preprocessed multi-omics dataset from matched samples, prepared in a combined matrix format as in (link to Python notebook here)
- A conditions file in matrix format, which specfies the experimental conditions for each sample in the training dataset above
- A contrast file that specifies which pairs of experimental conditions are to be compared during pathway analysis

to perform the following tasks:

- Download and prepare KEGG pathways for pathway analysis (Step 1)
- Infer Boolean regulatory/signaling rules for KEGG pathways using the combined multi-omics dataset (Step 2)
- Perform topology-informed pathway analysis for user-specified pairs of experimental conditions (Step 3)

This tutorial will go through the mBONITA pipeline using a multi-omics dataset of transcriptomics, proteomics, and phosphoproteomics from RAMOS B cells, as described in the mBONITA publication.

## Step 0: Process multi-omics data and generate conditions and contrast files

I expect that most users will begin with 2 or more processed datasets from separate multi-omics datasets. These datasets will usually be log2-normalized. The Jupyter notebook (**Figure1.ipynb**) outlines how to combine log2-normalized proteomics, phosphoproteomics and transcriptomics datasets as in the mBONITA publication and prepare them in a matrix format for mBONITA.

mBONITA also requires a condition and contrast file for pathway analysis. An example of how to prepare these files is in (**Figure1.ipynb**).

Briefly, if your dataset looks something like this:

| Genes | Condition1 replicate1 proteomics | Condition1 replicate2 proteomics | Condition2 replicate1 proteomics | Condition2 replicate2 proteomics | Condition1 replicate1 phosphoproteomics | Condition1 replicate2 phosphoproteomics | Condition2 replicate1 phosphoproteomics |
|---|---|---|---|---|---|---|---|
| Gene1 | - | - | - | - | - | - | - |
| Gene2 | - | - | - | - | - | - | - |
| Gene3 | - | - | - | - | - | - | - |
| Gene4 | - | - | - | - | - | - | - |

Then your condition file will look like this:

| Sample | Condition1 | Condition2 |
|---|---|---|
| Condition1_replicate1_proteomics | 1 | 0 |
| Condition1_replicate2_proteomics | 1 | 0 |

| Sample | Condition1 | Condition2 |
|---|---|---|
| Condition2_replicate1_proteomics | 0 | 1 |
| Condition2_replicate2_proteomics | 0 | 1 |
| Condition1_replicate1_phosphoproteomics | 1 | 0 |
| Condition1_replicate2_phosphoproteomics | 1 | 0 |
| Condition2_replicate1_phosphoproteomics | 0 | 1 |
| Condition2_replicate2_phosphoproteomics | 0 | 1 |

And your contrast file will look like this:

Condition1 | Condition2 |

# Step 1: Download and prepare KEGG pathways for pathway analysis

Use the command `python3 pathway_analysis_setup.py --help` for more information on each parameter. The examples below cover most use cases.

- Option 1: On a gmt of human pathways BONITA needs omics data, gmt file, and an indication of what character is used to separate columns in the file

**comma separated**

```
python pathway_analysis_setup.py -gmt Your_gmt_file -sep , Your_omics_data
```

**tab separated**

#python pathway_analysis_setup.py -t -gmt Your_gmt_file Your_omics_data

- Option 2: On all KEGG pathways for any organism BONITA needs omics data, organism code, and an indication of what character is used to separate columns in the file.

**comma separated, human:** *MOST COMMON USAGE*

```
python pathway_analysis_setup.py -org hsa -sep , Your_omics_data
```

**comma separated, mouse**

```
python pathway_analysis_setup.py -org mmu -sep , Your_omics_data
```

**tab separated:**

```
python pathway_analysis_setup.py -sep , -org "hsa" --data "phospho_LSP1.csv"
```

- Option 3: On a list of KEGG pathways for any organism BONITA needs omics data, organism code, the list of pathways, and an indication of what character is used to separate columns in the file.

**comma separated, human**

```
python pathway_analysis_setup.py -org hsa -sep , -paths Your_pathway_list Your_omics_data
```

**comma separated, mouse**

```
python pathway_analysis_setup.py -org mmu -sep , -paths Your_pathway_list Your_omics_data
```

**tab separated**

```
python pathway_analysis_setup.py -t -org Your_org_code -paths Your_pathway_list Your_omics_data
```

# Step 2: Infer Boolean regulatory/signaling rules and calculate node importance scores for KEGG pathways using the combined multi-omics dataset

Simply run the script **find_rules_pathway_analysis.sh** which will automatically submit appropriate jobs to a SLURM queue:

```
bash find_rules_pathway_analysis.sh
```

Please note that these scripts are written for SLURM. **find_rules_pathway_analysis.sh** loops over all networks to execute the script **calcNodeImportance.sh**, which in turn executes the Python script **pathway_analysis_score_nodes.py**. I'm open to writing these scripts for other job scheduling managers. The Python script can also be run by itself on a desktop, but I advise doing this only for small networks/training datasets.

# Step 3: Perform topology-informed pathway analysis for user-specified pairs of experimental conditions

Run the Python script pathway_analysis_score_pathways_mBonita.py with the following parameters. An example is listed below.

- path to training dataset file (concatenated)
- conditions file
- contrast file

For file formats, please refer to Step 0.

Here is an example command:

```
python3 pathway_analysis_score_pathways_mBonita.py concatenated_datasets.csv concatenated_conditions.csv
contrasts.csv -sep ,
```

# Analysis of the mBONITA output

## Inferred Boolean rules

Jupyter notebook: **Figure4.ipynb**

Python script: **Figure4.py**

These scripts contain code to open the local1.pickle files generated during the rule inference process (these files contain the inferred network model in a slightly complex data structure) and process the information into a single dataframe.

**One row in the dataframe contains information for one node. The dataframe has the following columns:**

- Network name - readable, descriptive KEGG network name
- Method name - subfolder of the main directory in which the pickle was found
- andNodeList - indices of parent nodes
- andNodeInvertList - a bitstring encoding the activation and inhibition edges. True implies that the edge from the corresponding parent node in the andNodeList is an inhibitory edge
- ruleLengths - length (ie, size) of the ERS for the node
- equivs - bitstring representation of the equivalent rule set
- plainRules - plain text representation of the rules in the ERS
- randomERSIndividual - random individual from the ERS
- minLocalSearchError - lowest error for the rules tried for each node

## Node importance scores

Importance scores are stored as node attributes in the **xyz_rules.graphml** files generated after the node importance score calculation step (Step 2 above). These graphml files can be visualized in software such as Gephi or Cytoscape.

Alternatively, Figure4.py has some suggestions for reading in these graphml files and aggregating these node importance scores using pandas and networkx and generating a single dataframe.

## Pathway analysis

Results are returned as a single csv file, **pvalues.csv**.

See **Figure4.py** for some suggestions on plotting the results.