MMinte is an application that uses metabolic models of the species present in a community under particular nutritional conditions to infer what are the types of interactions occurring between pairs of species. The user provides data about associations between pairs of organisms that allows the creation of a visual network of associations and 16S rDNA sequences for representative OTUs and MMinte assigns a type of interaction to each link between the organisms. Our application is composed of seven widgets that run sequentially, with each widget utilizing as the default input the file created in the previous widget. While MMinte may be run as a streamlined pipeline, due to its compartmentalized nature, the user is given the ability to better control the full analysis. The user may start the application at any of the seven widgets, as long as the data provided has the adequate structure. Furthermore, the user can access the output files of each widget, and verify the quality of the data produced at each step of the analysis, as well as explore it with alternative tools.

The team involved in the development of **MMinte** is composed of scientists and developers that work at the Mayo Clinic and Harvard Medical School. We would love to hear from you, so send us your comments, concerns, suggestions, wonderments, (cake recipes if you have any), and thoughts to microbialmetabolicinteractions@gmail.com or just send us a chat invite to that email.

MMinte was developed to work in MacOSX and Linux Machines. We didn’t test it in Windows machines, but we’re not saying it won’t work.

(While we will be developing a more complete version of MMinte in the next few months, if you want to help make a Windows version, that would be awesome! If you want to help improve MMinte that would be equally awesome! We would love to make MMinte a community driven project! So, just fork MMinte and let us know to the address microbialmetabolicinteractions@gmail.com !)

What do I need to have installed in my machine before setting up MMinte?

- **Python 2.7**

    - **MMinte** was only tested in Python 2.7. I'm not saying it won't work on Python 3, but you have to try at your own risk. To see which version of Python you have on your machine, open the terminal window and type `python --version`. You can find instructions on how to download and install Python on your machine from https://www.python.org/downloads/ .

- **pip**

    - pip is awesome. You can easily install all **MMinte**'s requirements by calling the install command of pip. You can find instructions on how to download and install pip on your machine from http://pip.readthedocs.org/en/stable/ .

If you have neither **Python 2.7** or **pip** installed you can either install them separately as suggested above or you can use home-brew from the terminal in **Mac** (if you don't have homebrew installed, follow the instructions on the webpage http://brew.sh/ -it's really just one line of code) and type `brew install pip` . You can type `apt-get install python-pip` if you are using **Linux** instead.

- **virtualenv** - To make sure the versions of all of **MMinte**'s dependencies are correct and no conflicts occur, we suggest you run **MMinte** in a virtual environment. **virtualenv** is pretty neat and, from the site of the developers on https://virtualenv.readthedocs.org/en/latest/# , "is a tool to create isolated Python environments". You can find the instructions for downloading and installing **virtualenv** on that page. In summary, you can just type `pip install virtualenv` (see how **pip** is so handy?)

- **Firefox** - We tested **MMinte** in Chrome and had problems only with the visualization of the networks Widget 7, thus the suggestion you use Firefox. If you don't have Firefox installed, go to www.mozilla.org/firefox , download Firefox and follow the installation instructions)

How do I setup MMinte?

1. Download the **MMinte** package. To do this go to

http://github.com/mendessoares/MMinte and click on Download ZIP button on the right side of the screen. This will (likely) download the ZIP file to your Downloads folder.

2. Your Downloads folder now has a MMinte_package.zip file. You can uncompress this file by double clicking the file. This will create the folder MMinte_package on your downloads folder. Move this folder to where you would like to keep it permanently. I suggest you move it to your Documents folder. You can do this by dragging the folder to Documents folder, or through command line. Here's how to do the latest. Open terminal, then type: `cd ~/Downloads/` and press Enter. Then type `mv MMinte_package/ ~/Documents/` and press Enter. The MMinte_package folder should now be on your Documents folder.
3. Use the terminal on your machine to go to the MMinte_package folder by typing `cd ~/Documents/MMinte_package/`
4. Go to the **MMinte** folder by typing `cd MMinte`
5. Create a virtual environment by typing `virtualenv env`.
6. This will create a folder in **MMinte** called env. This is where all the packages and modules you need for doing all the analysis will be stored. This avoids potential conflicts between the versions of packages and modules needed for **MMinte** and any you may have in your system. You can see this folder listed if you type `ls` on the terminal
7. Enter your virtual environment by typing `source env/bin/activate` . This command assumes you are right outside your env folder. If you are not sure where you are, type `cd ~/Documents/MMinte_package/MMinte/` , press enter, and try entering your virtual environment again.
8. We will now install all the dependencies required for **MMinte** to run. If you are working on a MacOS system, type `bash install-requiredPKGMacOS.sh` . If you are working on a Linux system, type `bash install-requiredPKGLinux.sh` . Wait for everything to get installed. The Python module **libsbml** takes quite a while to get installed, so the whole process takes around 10 minutes on an average speed internet connection. If no warnings show up on the terminal window when the prompt in the terminal (`$`) shows up, then all was successfully

installed. If you get an error message from BioPython saying that numpy is not installed, don't worry. numpy is not needed for **MMinte** to work.

9. You are now ready to run **MMinte** from your virtual environment.
10. If you don't want to run **MMinte** at this point, you can deactivate your virtual environment by typing `deactivate` and closing the terminal. If you want to go ahead and run **MMinte**, go to the next section, **How do I run MMinte** and start from step 4.

How do I run MMinte?

1. Open the terminal window on your computer.

2. Go to the **MMinte** folder by typing `cd ~/Documents/MMinte_package/MMinte/`.

3. From this location type `source env/bin/activate`. This will activate your virtual environment. If you are not sure if your virtual environment is activated or not, you can check if the line of your prompt in the terminal has `(env)` in the beginning.

4. Go to the site folder by typing `cd site`.

5. Launch **MMinte** by typing `python launchMMinte.py`

6. Open a Firefox browser and go to the address http://127.0.0.1:8080/ . A window asking if you want to accept incoming network connections may pop up. You can just say yes to that. The following message shows up in the terminal, but you don't have to worry about it: `CherryPy Checker: The Application mounted at ' ' has an empty config.`

7. Play with **MMinte**.

8. When you are done, close the browser window and press CTRL+C on your terminal. This will terminate your **MMinte** session. You should get your prompt back on the terminal window. You can also close the browser and use CTRL+C on your terminal at anytime to terminate **MMinte**. It may take a little longer for you to get your prompt back if you do this thought, since

MMinte will be cleaning up before shutting down.

9. You can then leave your virtual environment by typing `deactivate`.
10. At this point, you can also close the terminal window (if you really want to.)

What does each widget do?

MMinte is composed of 7 widgets that may be run sequentially or individually:

- *Widget 1*: Using information about pairs of operational taxonomic units (OTUs) that are associated to some degree and a list of 16S rDNA sequences for the OTUs in a particular community, a file is created containing only the sequences for representative OTUs significant to the analyses.
- *Widget 2*: The representative OTUs are identified and assigned a genome ID using BLAST and a local database containing the 16S rDNA sequences of species with whole genome sequences.
- *Widget 3*: Metabolic models for each genome ID are reconstructed and gap-filled using ModelSEED and downloaded to the user's local machine.
- *Widget 4*: Metabolic models of 2- species communities are created using COBRA Tools for the python computational framework (COBRApy).
- *Widget 5*: Under defined nutritional conditions, the growth rates of each species in isolation and when in the presence of another species in the community are estimated.
- *Widget 6*: The kinds of interactions occurring between the pairs of species on the nutritional conditions defined Widget 5 are predicted. The interactions are either positive (commensalism, mutualism) or negative (parasitism, amensalism, competition).
- *Widget 7*: A network is plotted with D3.js using the initial information of associations between the pairs of OTUs provided by the used, and the kinds of interactions predicted to be occurring.

What kind of files does MMinte need as an input?

The input files that are essential for **MMinte** to start running are two:

1. File with associations between pairs of Operational Taxonomic Units (OTUs). This is a text file (.txt) that has, for each row, the identity of 2 OTUs and a measure of association between them. You can find and example file (corrs.txt) in the folder called userFiles . This file will contain only the information for the pairs of OTUs you, as a user, are interested in further exploring.
2. File with the sequences of representative OTUs in the FASTA format (.fasta). An example of this file (outs.fasta) can be found in the folder called userFiles . This file may contain the sequences for all the OTUs of a previous analysis you may have run, and so, a lot more OTU sequences than are really needed for the analysis. Widget 1 basically takes all these sequences and the information in file in point 1. to reduce the number of sequences to only the ones that are really necessary for the rest of the analysis

However, you can provide your own files for **MMinte** to analyze in each widget. You can see some examples of these files in the folder exampleOutputs in the supportFiles folder. To know which file each widget takes as input, just open the widget and see which file is mentioned in the box on the left. If you want to make sure the widgets work, transfer the files in the exampleOutputs folder to the userOutput folder. **An exception** is the file data4plot.json which you need to transfer to the site folder.

Where should my files go?

Your files can go anywhere in your system really. As long as you tell each widget where they are by writing the path to them in the box on the left. However, I would suggest you keep them in the userFiles folder. This means that the path you need to write on the box will be `../userFiles/nameOfFile` where nameOfFile stands for the real name of your file (including extension.)

Where do the files with the results of each widget go?

Right now, by default, all results will show up in the userOutput folder with a default name. Because the user can’t choose what to name the files, your old result file will be overwritten if you re-run any **MMinte** wigdet. I suggest you move the files you are interested in keeping to another folder before you re-run a widget. That way you'll be able to keep all the results of different analysis.

Sometimes it seems that MMinte crashed. Is there a way for me to check how the program is going?

1. Go to the supportFiles folder
2. You can open the file logError_file.txt on any text editor, BUT,
3. If you want to keep track of things happening in real time, open the terminal and type the following `cd ~/Documents/MMinte_package/MMinte/supportFiles`, press enter, then type `tail - logError_file.txt` and press enter. Your terminal will show you the last few lines of the logError_file.txt in real time, so you can see what the analysis is doing. When you don't want to see this anymore just press CTRL-C and you will be back at the prompt. If you still think there are problems, contact us at microbialmetabolicinteractions@gmail.com, and send us the logError_file.txt file, so we can try to help. We want you to be able to use **MMinte** and also make improvements, so please do not hesitate in contacting us.

What Python modules does MMinte require?

Most of us are scientists with little or no formal training in software package development. The only reason we were able to put **MMinte** together is because other really smart people in the Python community have developed really neat packages to make our lives easier. Listed below are the packages and versions used by **MMinte**. We highly recommend you check the development pages for these packages (you can just Google the package name and follow the links) to see other ways you can potentially use them in your own research, or to learn more about the work of the developers. Here is the list of packages/modules:

biopython==1.66

CherryPy==3.8.0

cobra==0.4.0b3

cycler==0.9.0

```
DataSpyre==0.2.0

Jinja2==2.8

MarkupSafe==0.23

matplotlib==1.5.0

numpy==1.10.1

pandas==0.17.1

pyparsing==2.0.6

python-dateutil==2.4.2

python-libsbml==5.12.0

pytz==2015.7

scipy==0.16.1

six==1.10.0

wheel==0.24.0

requests==2.8.1
```