PLANET: A Web Server for Analysing Inter-taxa Dependencies in Metagenomic Data Using Differential Proportionality Networks

INDIVIDUAL REPORT

Yunsoo Kim, Virginia Fairclough, Dr John W. Pinney

March 21, 2016

IMPERIAL COLLEGE LONDON

THEORETICAL SYSTEMS BIOLOGY GROUP

MSC BIOINFORMATICS AND THEORETICAL SYSTEMS BIOLOGY

Acknowledgments

I would like to acknowledge my supervisor, Doctor John Pinney, for his suggestions and the opportunity to work with him on the project. I would also like to thank Virginia Fairclough for her support, insights, and supervision. Last, but not least, I would like to express my gratitude to my teammates, Laura de Arroyo Garcia and Ana-Isabella Tanase for their hard work, cooperation, and support throughout the project.

Abbreviations

BIOM	Biological Observation Matrix (BIOM)
CSS	Cascading Style Sheets
EBI	European Bioinformatics Institute
FDR	False Discovery Rate
GPU	Graphic Processing Unit
HTML	HyperText Markup Language
ID	Identifier
JSON	JavaScript Object Notation
NaN	Not a Number
OTU	Operational Taxonomic Unit
PLANET	ProportionaLity Analysis of NEtworks Tool
RGB	Red Green Blue
rRNA	Ribosomal Ribonucleic Acid
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

Contents

Abstract

Here my contributions for the development of a freely available web server, PLANET, are described. My major contributions are focused on the development of the back-end, which can be divided into three parts: input file parsers, a method to generate a differential network using proportionality measures, and servers development. The back-end connects all the components of the front-end with the method designed by Ana-Isabella Tanase, and uses outputs of the method to display an interactive differential network developed by Laura de Arroyo Garcia.

1 Background

The microbiome is the collection of genetic materials from microbial communities. Until recently, microbiome research had limitations because many microorganisms do not survive under ordinary laboratory conditions (*Handelsman, 2004*). However, advances in sequencing technology have allowed sequencing of the microbiome *in situ* (*Bragg and Tyson, 2014*). Thus, metagenomics, which is the study of the microbiome, has uncovered the information regarding uncultured microbiotas, which are microbial communities, in various environments.

16S ribosomal RNA (rRNA) gene-based profiling is often used to study the diversity of a microbiota (*Woese and Fox, 1977, Zoetendal et al., 2008*). The 16S rRNA gene is present in all prokaryotes because the 30S small subunit of prokaryotic ribosomes contains the 16S protein product. This gene has conserved regions, which are used to design primers, whilst the variable regions between the conserved regions are used for detecting the differences in microorganism sequences (*Rajendhran and Gunasekaran, 2011*). Thus, the 16s rRNA gene is a widely used molecular marker for identifying species within the microbiota. Operational taxonomic units (OTU)s are clusters of sequences that have been clustered by sequence similiarity (*Chen et al., 2013, Kuczynski et al., 2012*).

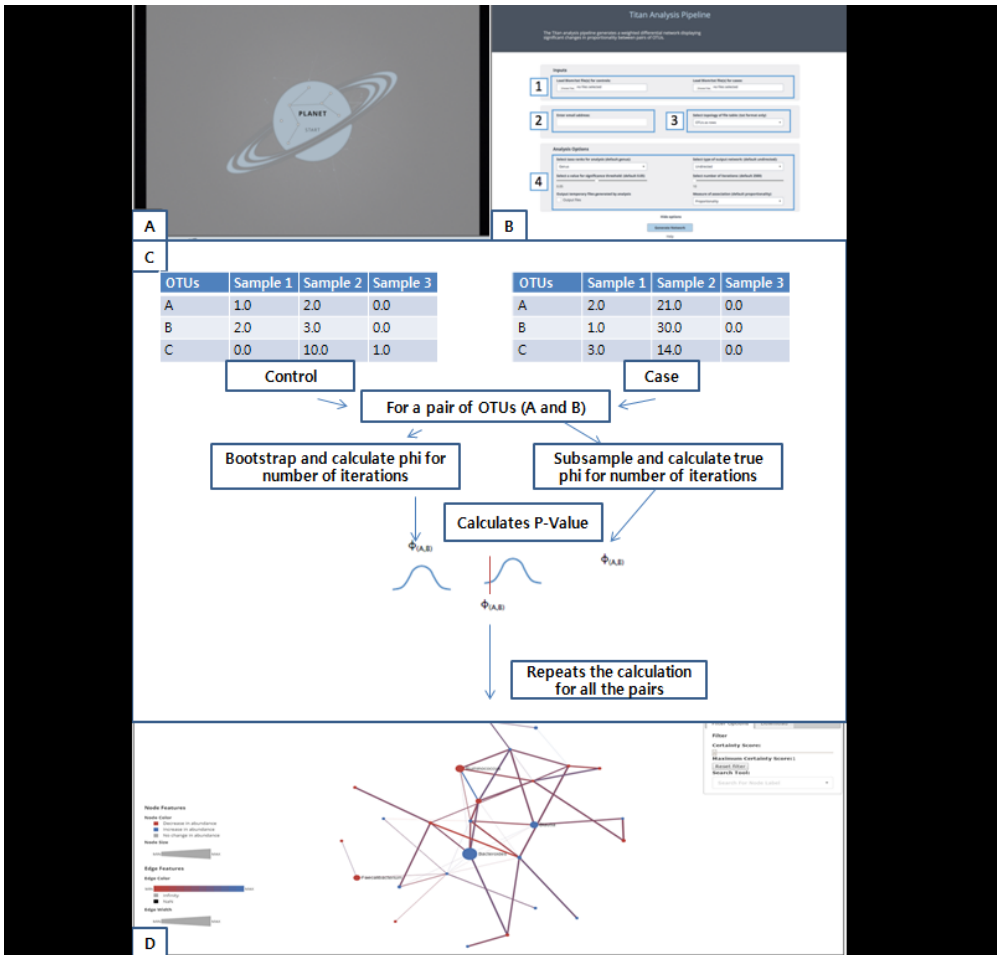
In fact, many studies used 16s rRNA gene-based profiling to establish a catalogue of taxonomic and functional compositions of a microbiota (*Walker et al., 2015*). Recent studies on the human gut microbiome highlighted the relationships between microbiomes and disease phenotypes, and portended the possibility of applications such as new drug targets (*Walker et al., 2015*); (*Langille et al., 2013*); (*Greenblum et al., 2011*). Many studies used interactions and dependencies between microorganisms to draw conclusions regarding the phenotypes (*Langille et al., 2013*); (*Greenblum et al., 2011*). Many of them used correlation for the calculation of changes in dependencies (*Langille et al., 2013*); (*Greenblum et al., 2011*). However, it was recently argued that correlation is not a proper method to detect changes in dependencies between microorganisms because the abundance data is relative data (*Lovell et al., 2015*). Lovell *et al.* suggested that a measure of goodness of fit to proportionality is more appropriate for analyzing dependencies between relative data because it guarantees a relationship between relative data and absolute data (i.e. a linearity in relative data would represent a linearity in absolute data) which correlation fails to draw.

We developed a freely available web server, PLANET (ProportionaLity Analysis Network Tool), in which significant changes in inter-taxa dependencies between OTUs are analyzed using proportionality. Our website generates a differential network to represent the analyzed changes in dependencies. As the website provides a general overview of the changes in interactions between OTUs, researchers can use the website as a hypothesis generation tool.

Here I describe my individual contributions to the development of the web server. I developed parsers for the input files, a method to measure significant changes in dependencies using proportionality, and created the back-end servers for the website. Other minor contributions are described briefly as well. Most of my contributions were focused on the back-end of PLANET, but I also worked on some of the front-end features such as the homepage of the website, the contact page with emailing functionality, and some components of the network visualization page.

2 Development of PLANET

PLANET web server constitutes of two main parts: the front-end and the back-end (Fig. 1.). The front-end is the user interface of the web server. In other words, the front-end is the user-friendly website built with HyperText Markup Language (HTML) templates. Its aim is to provide an accessible and visually appealing website where researchers can use their own metagenomic abundance data to generate an interactive differential network, which is also presented at the front-end. PLANET back-end is to create a web server for the front-end and to integrate the front-end with the functionality of the server, which is the calculation of changes in dependencies using proportionality for generating a differential network. All components of PLANET back-end are written or implemented in Python; input file parsers and proportionality calculation method for dependencies between OTUs are developed in Python language; Flask, which is a microframework developed in Python, is implemented as a Python module for the development of a web server; Celery, which allows the proportionality measure calculation to be a background task, is also imported as a Python module; Redis server is used as a storage for the Celery task outputs and a message broker, connecting Flask and Celery, and its functional assignments are done in a Python environment. Of the two main components, I consider my major contributions to the web server to be the back-end. Thus, here I describe the back-end in detail.


Titan Analysis Pipeline
PLANET
A
B
1
2
3
4
C
Control
Case
For a pair of OTUs (A and B)
Bootstrap and calculate phi for number of iterations
Subsample and calculate true phi for number of iterations
Calculates P-Value
$\Phi_{(A,B)}$
$\Phi_{(A,B)}$
$\Phi_{(A,B)}$
Repeats the calculation for all the pairs
D

OTUs	Sample 1	Sample 2	Sample 3
A	1.0	2.0	0.0
B	2.0	3.0	0.0
C	0.0	10.0	1.0

OTUs	Sample 1	Sample 2	Sample 3
A	2.0	21.0	0.0
B	1.0	30.0	0.0
C	3.0	14.0	0.0

Figure 1: **Overview of PLANET Analysis Pipeline.** This figure provides a general overview of the PLANET Pipeline PANEL A - the home page where the central logo can be clicked to start the analysis. PANEL B - the input page for the analysis. Box 1 shows the boxes for input files. Box 2 is for email address which the user can provide for an update of the status of the analysis. Box 3 is for the structure of the text input files. Box 4 represents the optional parameters. The user does not have to specify these parameters as the default values can be used. PANEL C - a summary of the methodology to generate a differential network using proportionality measures. PANEL D - a visual representation of the differential network at the front-end

3 Back-End Structure

3.1 Input File Parsers

PLANET accepts two input file formats and has different parsers for the input file formats (Figure 1). One accepted input file format is the Biological Observation Matrix (BIOM) file. It contains the descriptions of biological samples, i.e. OTUs or taxonomic assignments and their abundance data (*McDonald et al., 2012*). It is widely used in comparative omics including metagenomics (*McDonald et al., 2012*). The BIOM format is widely used for metagenomic data because it can be easily constructed using 16s rRNA sequence data (*Kuczynski et al., 2012*). In fact, the European Bioinformatics Institute (EBI) metagenomics database uses the BIOM file format as a taxonomic abundance data format (*Mitchell et al., 2015*). Thus, we decided that PLANET would use BIOM file format as the default input file format.

However, various versions of the BIOM file format have different ways of storing the data (JSON and HDF5), which complicated the initial approach of directly calculating proportionality from the raw (input) BIOM files. Thus, I designed a pipeline to use a table as a common intermediate data format which the proportionality calculation works on. I developed two parsers for the conversion from the BIOM file format to a table format. The parsers filter OTUs at a taxon level and then extract abundance data of the remaining OTUs. The converted table is used for the proportionality measure calculation between pairs of OTUs.

The reason that I implemented two parsers for the BIOM file format was to test two different approaches. One of the parsers writes temporary files and uses them for the analysis. The parser uses the BIOM Python package as a command line tool; “biom convert” tool is used to convert the BIOM into a human readable table in a text format with the last column containing the taxonomic assignment for each row (*McDonald et al., 2012*). The converted table has columns as samples and rows as OTU IDs. Then, the table is imported into a Python environment.

The rows of the table are filtered for a specific taxon level, which the users choose when they run the analysis. PLANET has the default taxon level as Genus. In fact, all taxon levels below Domain are available (Kingdom, Phylum, Class, Order, Family, Genus, Species). Domain is not included because many public BIOM files did not have any taxonomic assignments at Domain level. The highest taxonomic level that those BIOM files had was Kingdom. For prokaryotes, there is no need to have taxonomic assignments at Domain level because Domain and Kingdom have the same labels, either Archaea or Bacteria. The taxon filtered table is exported as a temporary file. The temporary file is imported to Python as a pandas data frame by pandas package "read CSV" function (*McKinney, 2010*). The package dependencies are summarized in the following list:

- pandas - used to combine duplicate OTUs and remove samples with only zeros
- re - used to assign the file extension of the temporary files
- os - used to run the BIOM command line tool

Another parser does not write temporary files. It imports the BIOM file into a Python environment using the BIOM Python package. Then, the BIOM Python package is used to filter OTU IDs based on a taxonomic level. The filtered IDs and corresponding abundance data are used to create a pandas data frame. The duplicate OTUs and samples with only zeros are filtered in the same way as in the other parser. This parser uses the following packages:

- pandas - used to combine duplicate OTUs and remove samples with only zeros
- biom.parse - used to load biom files into Python and filter OTUs

The difference in the approach resulted in the difference in efficiency. The parser without temporary files is time efficient by a factor of 30 to 50 because it does not use time on creating and loading temporary files. The other parser is memory efficient because it uses temporary files to converts the BIOM file into a table format. In theory, the parser with temporary files works better with larger input files, while the other parser is more appropriate for smaller files.

Still, both parsers generate the same output, which is a converted table containing the filtered OTUs and corresponding abundance data. Also, although their memory and time efficiencies are different, they do not affect the overall efficiency of the analysis notably because most of the time and memory are used up by the differential proportionality calculation. Thus, I recommend using the parser with temporary files only if interested in acquiring temporary files. Otherwise, the parser without temporary files should be selected. In fact, the parser without temporary files is the default parser for the analysis.

The initial concern with various BIOM file versions is solved by the parsers. All BIOM file versions can be used for the analysis as long as they have the right file extension, which is ".biom." However, the pipeline can still raise an error due to the input BIOM files, if they do not contain the right information such as no taxonomic assignments. Also, if they have binary data i.e. 0 and 1 for abundance data, then the result would not carry meaningful information because proportionality equation involves calculation of variance, and variance of zeros and ones are not interesting (but this does not mean that the analysis does not complete).

PLANET also accepts a text file format. The accepted text file is a tab-delimited table with OTUs as either rows or columns. The text file parser is almost a duplicate of the last part of the BIOM file parser with temporary files. The text file is treated as the taxon filtered temporary table file. It is imported to a Python environment as a pandas data frame using the pandas "read CSV" function. If OTUs are on columns, the data frame is transposed. Then, duplicate OTUs and samples with only zeros are filtered. Thus, the text file parser uses just the pandas Python package. We implemented a text file parser because some applications calculating abundance data from 16s rRNA fasta files convert their output BIOM files into human readable text files

(*Kuczynski et al., 2012*).

Furthermore, I implemented a file format detector so that the user does not have to specify whether they inputted a BIOM file or a text file. However, the user has to specify whether the OTU IDs are on rows or columns. The default is OTU IDs as rows.

When public data sets were downloaded for testing purposes, data from the EBI metagenomics database had only one sample per BIOM file. In other words, the parsers for PLANET needed a capability of accepting multiple abundance data files for one set of data. Thus, parsers were modified so that they can accept multiple files. The multiple files were parsed individually, and then the final multiple panda data frames were merged into one data frame for each set of data. The merged data frames are exported as a human readable table format if the user wanted to have temporary files. The front-end was changed accordingly to allow the users to select multiple files as input data.

The EBI metagenomics database also uses another data format for taxonomy abundance data, which is the TSV file format. The TSV file is different from the BIOM and text file input files. Thus, in the future it would be beneficial to design a parser for TSV file format. The TSV file has, for each line, OTU id, metadata, and abundance values delimited by tabs. Thus, parsing the information line by line using Python can be implemented to generate a data frame that the proportionality function can use.

3.2 Method Implementation

The measure of goodness of fit to proportionality is calculated using the following equations:

$$\phi_{(A,B)} = \frac{var(log(A/B))}{var(logA)} \tag{1}$$

$$\phi_{(B,A)} = \frac{var(log(B/A))}{var(logB)} \tag{2}$$

where A and B are arrays of abundance data for OTU A and OTU B.

Algorithm 1 Certainty Scoring

```
1:  procedure PHICALCULATION($BIOMcontrol$,$BIOMcase$, $samplingcount$, $alpha$)
2:      $OTUset = intersection(OTUsfromBIOMcontrol, OTUsfromBIOMcase)$
3:      $OTUsetTemp = OTUset$
4:      for each OTU $i$ in $OTUset$ do
5:          $OTUsetTemp$ =remove $i$ from $OTUsetTemp$
6:          for each OTU $j$ in $OTUsetTemp$ do
7:              initialise lists $Phidistribution$ and $TruePhi$
8:              while $count$ less than $samplingcount$ do
9:                  $bootstrapped = bootstrap(i$ and $j$ from $BIOMcontrol)$
10:                 $subsampled = subsample(i$ and $j$ from $BIOMcase)$
11:                 Calculates proportionality for $bootstrapped$ and $subsampled$
12:                 stores the results to $Phidistribution$ and $TruePhi$ accordingly
13:                 $count = count + 1$
14:             end while
15:             Calculation of p-values using $TruePhi$ and $Phidistribution$ with $alpha$
16:             Calculation of delta using $TruePhi$ and $median(Phidistribution)$
17:             FDR correction of p-values using $alpha$
18:             $certainty =$ number of p-adjusted less than $alpha$
19:         end for
20:     end for
21:     return average delta, average p-adjusted, certainty score for each pair of OTUs
22: end procedure
```

PLANET uses proportionality as a measure of dependencies between OTUs. The proportionality calculation method has two sets of abundance data frames as inputs and other input parameters, which are the number of sampling, alpha for significance level, and a boolean value for whether the edges are directed or undirected.

The calculation of proportionality is done in pairs of OTUs i.e. species A and B. For a pair of OTUs, samples (columns) with zeros are removed because the proportionality function does not accept zeros. Each set of resulting data frames should have at least three samples. We need at least three samples for each set of data frames because we are sampling n-1 samples from the data frames where n represents the smaller number of samples between the two sets and the proportionality equation has variance, which requires at least two items.

Then, one set of data is used to bootstrap (i.e. sample with replacement) n-1 samples. The other set of data is used to subsample (i.e. sample without replacement) n-1 samples. The subsampling is used to calculate a set of individual true proportionality values and all the bootstrapped samples represent a phi distribution. As the two possible combinations (i.e. AB and BA) for the two OTUs yields different values for proportionality, the phi value calculation is done for the two combinations. For the proportionality values from bootstrapped samples, infinity and NaN values are exempt from making the phi distribution. Subsampling and bootstrapping are repeated for the number of iterations as provided by the user. The bootstrapped phi distribution is used to calculate the p-value of each true phi-value. The resulting p-values are then corrected using the false discovery rate (FDR). The statsmodel.sandbox Python package is used for FDR correction (*Seabold and Perktold, 2010*). The corrected p-value is then compared to the threshold value. At this point, we have two sets of p-adjusted values, p-adjustedAB and p-adjustedBA. Two delta values (i.e. deltaAB and deltaBA) are calculated by taking the difference of the true phi-value and the median of the bootstrapped phi distribution.

The number of significant p-adjusted values out of the number of sampling iterations (known as the certainty score) is then used as an edge weight (width) of the differential network. Edge colors are assigned according to delta values. If the user selected the differential network to be directed, then the two certainty scores and two delta values for each pair of OTUs are used to assign the corresponding edge attributes of the two directed edges. Otherwise, averages of the values are used for edge attributes of the one undirected edge between a pair of OTUs.

Initially, networkx was used for network visualization. However, as the output network was not interactive, networkx was not the solution for our website. Still, when PLANET method is distributed as a Python package, and researchers use it as a command line tool, networkx can be used for network visualization. In fact, I implemented argparse and sys Python packages to allow the method to run on the command line.

The method has two outputs. One of them is a text file containing a tab-delimited table. Each line of the table has information from one edge. The fields of the text file (from left to right) are source node (OTU), target node (OTU), certainty score (from 0 to 1), delta value, and average adjusted p-value. The major reason for including such a text file as an output of the method is to allow users to run further statistical or network analysis using Cytoscape or any other tools they use. Cytoscape provides various levels of analysis from basic ones, such as degree distribution, to complex analyses such as clustering nodes based on statistical analyses (*Shannon et al., 2003*). The further analyses would provide additional information which can be used to design a possible model for a further experiment. Although PLANET does not currently provide the further analysis because the aim of the website is generating a differential network, in the future, basic statistical analysis or even clustering can be implemented using the networkx Python module.

Another output is the JSON file, which is used by sigma.js for network visualization. The JSON file is a dictionary of dictionaries; it has two dictionaries, nodes and edges. Then, nodes have 7 sub-dictionaries, and edges have 6 sub-dictionaries.

- Nodes

    1. Id = OTU
    2. Label = OTU
    3. x-coordinate
    4. y-coordinate
    5. Change in abundance
    6. Size (magnitude of the change in abundance)

7. Color (direction of the change in abundance) in RGB

- Edges

1. Source OTU
2. Target OTU
3. Id (Edge Id, which is randomly assigned)
4. Delta value from proportionality calculation
5. Width = Certainty score (0 to 1) from proportionality calculation
6. Color (Related to delta value) in RGB

Initially, the JSON file did not have node colors and sizes. However, we wanted the node attributes to provide some information about the input data. Thus, node size shows the magnitude of the change in abundance for an OTU between the two input data sets, and node color shows the direction of the change; a bigger node means bigger change. Blue represents an increase in abundance in the case data set with respect to the control, and red represents a decrease in abundance. Grey represents no change in abundance.

For edges, colors are in a gradient of red to blue. The colors are assigned according to the delta ratio (delta value normalized to values from 0 to 1). Red represents delta ratios closer to 0 and blue represents delta ratios closer to 1. NaN and infinity values are exempt from the calculation of delta ratio because they would create errors. However, they still have colors assigned: grey for infinity and black for NaN.

The proportionality script has the following Python package dependencies: os, sys, argparse, re, json, networkx, pandas, numpy, and statsmodels.sandbox NetworkX (*Schult and Swart, 2008*); (*McKinney, 2010*); (*Van Der Walt et al., 2011*); (*Seabold and Perktold, 2010*). Their usages are brifely summarized in Table 1.

Package	Usage
os	generates zip file for all the outputs from the analysis
sys and argparse	allow the script to be run as a command line tool for a possible distribution
re	assigns appropriate file extensions for the outputs
json	exports JSON data
networkx	applies spring layout to assign x and y coordinates for nodes
pandas	works with data frames
numpy	has many mathematical functions which are used for calculation
statsmodels.sandbox	FDR correction of p-values The efficiency of the method can be improved.

Table 1: **A Table for Python Pacakage Dependencies.** The table summarizes the python packages used in the pipeline, and how they were implemented in the pipeline.

In the future, it could be beneficial to implement a Python package called Theano (*Bergstra et al., 2010*). It allows Python functions to have graph instances and be run on Graphic Processing Unit (GPU). It is known that computation on GPU is much faster than that on CPU (***?***).

3.3 Servers Implementation (Y.K)

As other functions were in Python, for compatibility and easier integration of all the scripts, Flask was chosen for server development because it is a micro framework in Python for website development (*Grinberg, 2014*). Flask uses the Jinja2 template format, so all the HTMLs are in Jinja2 template format. Initial attempts of implementing Flask were not so easy because it has a strict restriction on file referencing and project directory structures. Therefore, the Cascading Style Sheets (CSS) and JavaScripts had to be referenced by using one of the builtin functions, not by the ordinary way of providing file paths.

Flask is mainly used to integrate all the HTMLs and connect them with the method developed to generate a differential network. Input files and parameters from the input page are sent to the Flask server which generates a random universally unique identifier (UUID) for the run. Also, it generates a log file with all the input parameters provided in the order of following:

- Email address for sending the results
- Input file (control) 1 path
- Input file 2 (case) path
- Taxon level at which the analysis is run
- Number of iterations for the sampling process
- α value used for threshold of p-value significance
- Output file path (which includes the run ID)
- Whether the network generated is directed or undirected
- Whether the temporary BIOM file parser was used
- For inputs in text format (if provided), whether the table was inverted to have OTUs as rows
- Dependency measure employed in the pipeline (either proportionality or correlation)
- Whether the intermediate data is returned to the user

The files are uploaded for the analysis. Previously, proportionality calculation function was imported as a module, and the function was used directly in the Flask instance. This approach limited redirecting the user to the loading page as the function had to be called before redirecting. Thus, the Celery server is implemented which allows the method to be run as a background task (*Rocco and Helmers,*). Whenever the function is called, rather than running and waiting for it to complete, Flask sends the work to the Celery server. Also, I use a Redis server as a storage of Celery results and a message broker between Flask and Celery servers for keeping track of the analysis (*Sanfilippo,*). As each analysis has unique UUID, by using its id, its analysis status can be tracked. Thus, Celery can handle emailing upon error or completion (if an email address was provided). It sends the URL with an appropriate message. As Flask is connected to the front-end and Celery server, the result page can get the description of an error if raised. This allows the page to have an error flag pop up with a simple description of what the error is and the predicted source of error. Flask also sends downloading files as a response to a request. Flask also handles incoming emails from the contact page. It sends an email to the PLANET server email with all the fields that user provided.

4 OTHER CONTRIBUTIONS

My other (smaller) contributions are summarized briefly in Table 2.

Contribution	Explanation
Front-page	designed it. I implemented the random network generation following the mouse cursor. I implemented logo in the center as a button to Titan analysis with the special hovering effect.
Titan input page	I implemented multiple input selection.
Network retrieval	implemented example data for the run checkbox. I used the result page to generate network retrieval specific result HTML.
Loading page	developed a javascript which reloads the page every 10 seconds and basic layout of it.
Contact page	introduced the template for the contact page.
Result page	developed neighbors display, edge filtering, and node searching, a dropdown search panel, download options (Refer to the manaul Section 4.3.2). I integrated undirected and directed result pages into one HTML.
Pop ups	developed them for the Contact page, the result page, and the input page (Refer to the manual Section 3.2).

Table 2: **A Table for Other Contributions.** The table summarizes my smaller contributions and provides a little descriptions for what I did.

5 Conclusion

My contributions to PLANET are mainly focused on the back-end.The back-end receives the input files from the front-end, and uses the appropriate parser to calculate changes in the dependencies between pairs of OTUs. The resulting differential network is then displayed at the front-end. Thus, it connects all the user interfaces (the front-end) designed by my colleagues with the functionality of the website, generating a differential network using the proportionality calculation.

References

Bergstra et al., 2010. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

Bragg and Tyson, 2014. Bragg, L. and Tyson, G. W. (2014). Metagenomics using next-generation sequencing. *Environmental Microbiology: Methods and Protocols*, pages 183–201.

Chen et al., 2013. Chen, W., Zhang, C. K., Cheng, Y., Zhang, S., and Zhao, H. (2013). A comparison of methods for clustering 16s rrna sequences into otus. *PloS one*, 8(8):e70837.

Greenblum et al., 2011. Greenblum, S., Turnbaugh, P. J., and Borenstein, E. (2011). Metagenomic systems biology of the human gut microbiome reveals topological shifts associated with obesity and inflammatory bowel disease. *Proceedings of the National Academy of Sciences*, 109(2):594–599.

Grinberg, 2014. Grinberg, M. (2014). *Flask Web Development: Developing Web Applications with Python.* " O'Reilly Media, Inc.".

Handelsman, 2004. Handelsman, J. (2004). Metagenomics: application of genomics to uncultured microorganisms. *Microbiology and molecular biology reviews*, 68(4):669–685.

Kuczynski et al., 2012. Kuczynski, J., Stombaugh, J., Walters, W. A., González, A., Caporaso, J. G., and Knight, R. (2012). Using qiime to analyze 16s rrna gene sequences from microbial communities. *Current protocols in microbiology*, pages 1E–5.

Langille et al., 2013. Langille, M. G. I., Zaneveld, J., Caporaso, J. G., Mcdonald, D., Knights, D., Reyes, J. A., Clemente, J. C., Burkepile, D. E., Thurber, R. L. V., Knight, R., and et al.

(2013). Predictive functional profiling of microbial communities using 16s rrna marker gene sequences. *Nat Biotechnol Nature Biotechnology*, 31(9):814–821.

Lovell et al., 2015. Lovell, D., Pawlowsky-Glahn, V., Egozcue, J. J., Marguerat, S., and Bähler, J. (2015). Proportionality: a valid alternative to correlation for relative data. *PLoS Comput Biol*, 11(3):e1004075.

McDonald et al., 2012. McDonald, D., Clemente, J. C., Kuczynski, J., Rideout, J. R., Stombaugh, J., Wendel, D., Wilke, A., Huse, S., Hufnagle, J., Meyer, F., et al. (2012). The biological observation matrix (biom) format or: how i learned to stop worrying and love the ome-ome. *GigaScience*, 1(1):7.

McKinney, 2010. McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Mitchell et al., 2015. Mitchell, A., Bucchini, F., Cochrane, G., Denise, H., Hoopen, P. T., Fraser, M., Pesseat, S., Potter, S., Scheremetjew, M., Sterk, P., and et al. (2015). Ebi metagenomics in 2016 - an expanding and evolving resource for the analysis and archiving of metagenomic data. *Nucleic Acids Res Nucleic Acids Research*, 44(D1).

Rajendhran and Gunasekaran, 2011. Rajendhran, J. and Gunasekaran, P. (2011). Microbial phylogeny and diversity: small subunit ribosomal rna sequence analysis and beyond. *Microbiological research*, 166(2):99–110.

Rocco and Helmers, . Rocco, M. and Helmers, J. H. Celery: Distributed task queue.

Sanfilippo, . Sanfilippo, S. redis.

Schult and Swart, 2008. Schult, D. A. and Swart, P. (2008). Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, volume 2008, pages 11–16.

Seabold and Perktold, 2010. Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, pages 57–61.

Shannon et al., 2003. 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 research*, 13(11):2498–2504.

Van Der Walt et al., 2011. Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Walker et al., 2015. Walker, A. W., Martin, J. C., Scott, P., Parkhill, J., Flint, H. J., and Scott, K. P. (2015). 16s rrna gene-based profiling of the human infant gut microbiota is strongly influenced by sample processing and pcr primer choice. *Microbiome*, 3(1).

Woese and Fox, 1977. Woese, C. R. and Fox, G. E. (1977). Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proceedings of the National Academy of Sciences*, 74(11):5088–5090.

Zoetendal et al., 2008. Zoetendal, E., Rajilić-Stojanović, M., and De Vos, W. (2008). High-throughput diversity and functionality analysis of the gastrointestinal tract microbiota. *Gut*, 57(11):1605–1615.