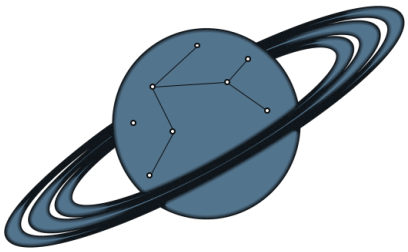


IMPERIAL COLLEGE LONDON

THEORETICAL SYSTEMS BIOLOGY GROUP

MSC BIOINFORMATICS AND THEORETICAL SYSTEMS BIOLOGY

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

USER MANUAL

Laura de Arroyo Garcia (L.A.G.), YunSoo Kim (Y.K), Ana-Isabella Tanase (A.T), Virginia Fairclough, Dr John W. Pinney

March 22, 2016

Executive Summary

- **PLANET (ProportionaLity Analysis NEtworks Tool) is a web-server that generates weighted differential networks for metagenomic data displaying altered dependencies between pairs of operational taxonomic units (OTUs).**
- **The methodology underlying this tool relies on the computation of proportionality values between pairs of OTUs in two different conditions, which is a measure of the linear relationship between two variables.**
- **Two back-end pipelines providing specific advantages are available to tailor to specific user requirements.**
- **The web-server is supported by a dynamic front-end built with a user-centric approach to promote an intuitive use of the analysis pipeline.**
- **PLANET provides extensive network visualisation and analysis features that enhance user experience and highlight biologically significant results.**

Acknowledgments

Our team would like to thank our supervisor, Dr John Pinney, for his valuable insights and for providing us with the opportunity to work on this project.

We would also like to express our gratitude to Virginia Fairclough for supporting us 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

IBD - Inflammatory Bowel Disease

ICD - Inflammatory Crohn's Disease

ID - Identifier

JSON - JavaScript Object Notation

NaN - Not a Number

OTU - Operational Taxonomic Unit

PDF - Portable Document Format

PLANET - ProportionaLity Analysis of NEtworks Tool

PNG - Portable Network Graphics

RGB - Red Green Blue

rRNA - Ribosomal Ribonucleic Acid

SVG - Scalable Vector Graphics

URL - Uniform Resource Locator

UUID - Universally Unique Identifier

Contents

1 Background (A.T)

1.1 Metagenomics

Metagenomics is defined as the study of microbial genetic material extracted from environmental studies. The development of this field stemmed from the observation that whilst our understanding of microorganism communities is mainly based on studies of samples put in culture, the vast majority of microbial species has never been cultured in a laboratory (*Handelsman, 2004*). The key advantage of metagenomics thus resides in the fact that it is a culture-independent method for analysing a microbial community. Advances in sequencing technologies and in particular the advent of next-generation sequencing have tremendously facilitated the exploration of the diversity within these communities (*Bragg and Tyson, 2014*). A commonly employed tool to generate a phylogenetic profile of a microbial sample is the sequencing of 16S ribosomal Ribonucleic Acid (16S rRNA) (*Woese and Fox, 1977, Zoetendal et al., 2008*). This non-coding RNA is a component of the 30S small subunit of prokaryotic ribosomes and is present in all prokaryotic living organisms. Its structure of interspersed, conserved and variable regions makes it an ideal phylogenetic marker of microbial taxa (*Rajendhran and Gunasekaran, 2011*). Probes can hybridise to the conserved regions for amplification, whilst the hypervariable regions in the loops mutate over time and can be used to identify species. The obtained sequences can be clustered by similarity into operational taxonomic units (OTUs), which are often defined at a 97% identity threshold for the 16S rRNA sequences (*Chen et al., 2013, Kuczynski et al., 2012*).

1.2 Relevance

The community of microorganisms living within a given environment is termed microbiota, whilst the collection of their genetic material is the microbiome. More specifically, the human

microbiota refers to the communities of microorganisms living in association with the human body. Each body site (skin, gut or oral cavity for example) hosts its own specific microbial community (*Turnbaugh et al., 2007*). They are all incredibly diverse, including bacteria, phage, viruses or archaea. These microorganisms live symbiotically with their host and are involved in the maintenance of health by interacting with their environment and with each other (*Consortium et al., 2012*). In the human gut, they can contribute to energy harvesting by breaking down polysaccharides that we are otherwise unable to digest, or promote the development of the immune system (*Sekirov et al., 2010*). Disruptions of the healthy microbiota (dysbiosis) can similarly lead to the development or aggravation of health conditions. Sokol *et al.* showed that inflammatory bowel disease (IBD) patients had an unbalanced intestinal microbiota characterised by a fungal community with altered composition, and additionally highlighted disease-specific inter-kingdom network alterations (*Sokol et al., 2016*). In order to understand the role of the microbiota in eubyosis and dysbiosis, we must investigate the symbiotic dependencies between the different constituents of these complex communities.

1.3 Measures of Dependency

16S rRNA profiling studies can be used to quantify the relative abundance of each OTU within a sample, which is determined by the number of reads in a given cluster (*Bikel et al., 2015*). A commonly employed measure to assess the level of pairwise dependencies between variables in biology is correlation (*Almudevar et al., 2006*). However, Lovell *et al.* highlighted that the use of correlation as a measure of pairwise association in relative data is inappropriate, as two variables being correlated in the relative data does not imply correlation in the absolute data that gave rise to it (*Lovell et al., 2015*). Furthermore, one important limitation of 16S rRNA phylogenetic profiling is the introduction of bias by the choice of both primers and hypervariable regions (*Brooks et al., 2015*), which can either enrich or select against certain OTUs. This fea-

ture can lead to further alterations in the correlation values obtained from relative abundances, as correlation is not subcompositionally coherent (the correlation value is dependent on the constituents included in the analysis) (*Lovell et al., 2015*).

The authors of the study propose a measure of goodness of fit to proportionality as an effective alternative to correlation. The directed proportionality values for a pair of variables (A,B) are calculated as follows:

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

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

Proportionality overcomes the problems described above by ensuring that variables that are proportional in the relative data are also proportional in the underlying absolute data.

2 Overview of PLANET (L.A.G.)

In this user manual, we present PLANET (ProportionaLity Analysis NEtworks Tool), a freely available web server that explores inter-taxa dependency changes between pairs of OTUs in two metagenomic datasets, controls and cases.

PLANET incorporates the aforementioned concept of proportionality in its methodology to highlight symbiotic relationships that are significantly altered in dysbiosis (microbial imbalance) when compared to healthy controls. Thus, it can be employed as a hypothesis generator to predict altered symbioses that might have an impact in the stability of complex microbial communities.

This piece of software is compliant with all major web browser technologies, and can be accessed from https://msc.bc.ic.ac.uk:20164. PLANET is composed of three major components:

- An eye-friendly, front-end web interface, containing a user input page for uploading files, overview of the software and tutorial pages, a network retrieval tab and a contact page.

- A network visualisation component, which retrieves a weighted differential network for result visualisation after proportionality calculation.

- A back-end execution component, composed of our core methodology (implemented in Python) and a Flask server providing a suitable web development environment for execution.

The primary tool within PLANET is called Titan. This exploratory instrument retrieves significant changes in proportionality measures through the provision of two metagenomic datasets carrying abundance readings. Input files should be in either *.biom* or *.txt* format (refer to Section 6.1 for further details). The methodology behind the generation of weighted differential networks is based on a repeated process of sampling and proportionality calculation (ϕ) for all possible OTU pairs, which allows testing for a significant change in dependency between the two datasets (*Fairclough et al., 2016*). The confidence in this change is provided as a certainty score. A summary of the implemented methodology is depicted in Figure 1 (also used in Laura de Arroyo's individual report), and will be discussed in more detail in Section 5 of this manual.

Prior to input file submission, the user is given the choice to run a default analysis or a customised analysis. The default analysis runs on 2,500 iterations, an alpha value of 0.05 and genus as its selected taxonomic level. However, these characteristics may be changed by altering the optional arguments provided on the input page settings. The user may also prefer to run an analysis using Pearson correlation instead of proportionality as the measure of inter-taxa dependency. Although this is not advised for relative abundance data (Section 1.3), this option is also provided by Titan for comparison purposes.

Titan can run on seven different taxonomy ranks (genus, species, family, order, class, phy-

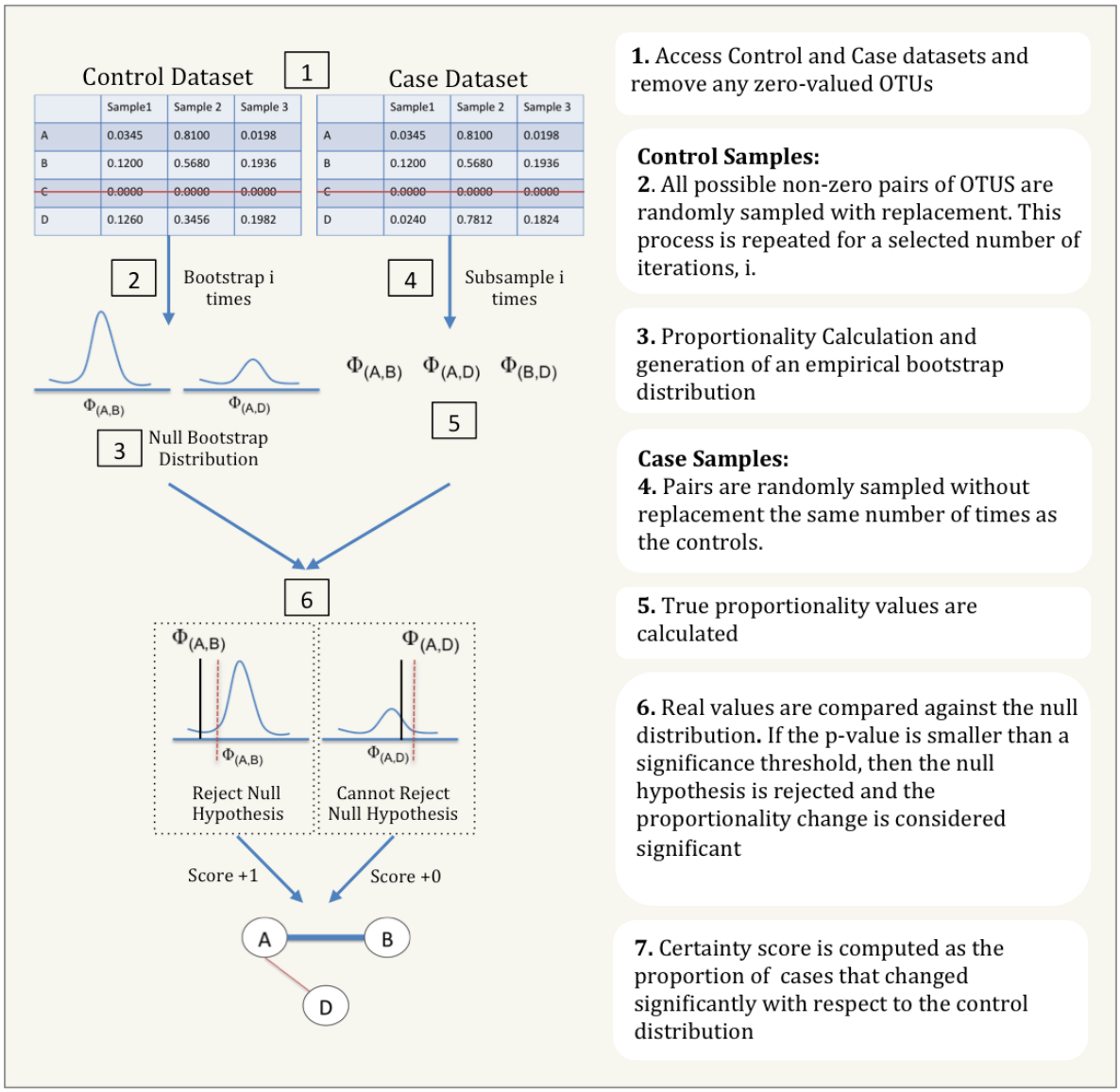


Figure 1: **Summarised Methodology (Produced by L.A.G.)**. A graphical representation of the procedure is shown on the left-hand side, with summarised descriptions of the different stages of the pipeline on the right. The two tables (1) represent the original, metagenomic datasets carrying abundance readings. The curves in blue (2 and 6) represent the Bootstrap ϕ distributions, produced after a number of iterations, i. The significance threshold is represented in red (6), and the true ϕ values (derived from proportionality calculation for case samples) are shown as black vertical lines (6). These exemplify when the null hypothesis would be or would not be rejected. Finally, an example of a simple differential network is shown. The blue edge indicates that there is a positive difference between the true ϕ value and the median of the bootstrap ϕ distribution (known as Δ value), which means that they are less proportional. A red edge indicates a negative difference and therefore higher proportionality. The thickness of the edge represents the confidence score, which is the proportion of iterations for which the difference in proportionality was shown to be significant.

lum and kingdom) and can yield both undirected or directed networks. Once files are uploaded, the user is redirected to a self-refreshing loading page and the output appears once the job is over. The data analysis process can take from a few minutes up to several hours, depending on the number of sampling iterations required, server load and the size of the input files. Network visualisation offers a permanent legend, a filter slider and search tool box. Furthermore, the user can download any files produced during the analysis, such as temporary and intermediate files, a log file with a summary of the run settings, and a JSON (JavaScript Object Notation) file, which is used by our pipeline to render the final network.

PLANET offers a set of additional features aimed at enhancing user experience across our web-page, the most relevant being an emailing functionality which eases result retrieval, a contact page, and a comprehensive tutorial on how to optimise PLANET's functionality.

3 Front-end Structure

3.1 Website Design (A.T)

The PLANET website was built on the Bootflat template (*Flathemes, 2016*), which combines functionalities of the Bootstrap framework with principles of flat design. These features allow the website to responsively adapt its layout to any type of supporting device, as shown in Figure 2. Examples of the dynamic nature of the layout include the format of the input containers, which can arrange horizontally or vertically, and the navigation bar that either extends or collapses, according to screen size. The front-end is currently composed of the ten following HTML (HyperText Markup Language) pages:

- A front-page (Figure 4, A)
- An overview page detailing the purpose and methodology of the Titan pipeline (Figure 5)

Figure 2: **The Web Server Layout Dynamically Adapts to Different Devices.** From left to right, the Figure shows the layout of the input-page on a tablet, mobile phone, desktop and laptop. The inputs align horizontally on the desktop and laptop, and vertically on the phone and tablet. Produced by A.T.

- A tutorial explaining how to use the PLANET tools (Figure 6)
- A Titan input page to launch an analysis (Figure 7, A)
- A network retrieval input page (Figure 7, B)
- A contact page (Figure 4, B)
- A loading page to display whilst a task is running
- Three result pages displaying either directed or undirected networks generated by a Titan run, or networks generated from a previous run of the server (Figure 9).

The PLANET logo (Figure 3) is a reminder of the server's acronym and its central purpose, which is the generation of networks.The website's theme relies on four colours, each with a particular purpose in the overall layout. The bright blue is used to highlight key elements,

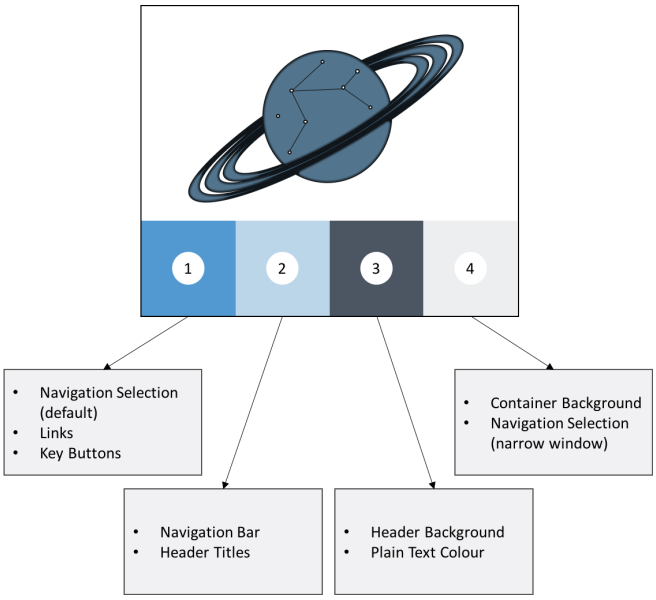


Figure 3: **PLANET Logo and Front-End Theme.** The logo of the PLANET website is shown at the centre of the Figure. The four theme colours of the website are shown below the logo. The grey boxes provide a description of the main purpose of each colour in the overall layout. Produced by A.T.

such as links, main buttons, or figure components. Light blue is used as the navigation bar background and font-colour of header titles. The main text is written in dark grey, which is also the background colour of the page header. Finally, the central containers on each page and the collapsed navigation bar have a light grey background. The personalisation of the website was performed by adding a customised CSS (Cascading Style Sheets) file with the desired features. The additional CSS provides new styling elements and overwrites the basic features provided by the Bootflat template.

All pages of the website share a number of common features. The navigation bar is fixed at the top of the viewport and scrolls with the content of the page whilst remaining discrete due to its partial transparency. Each page contains a main header with a title and a short description of the contents. Text and images are centered on the viewport within squared containers.

The front-page is an aesthetic introduction to the PLANET web-server. From this, the user can directly start a Titan analysis by clicking on the central logo or visit any other page through

the top navigation bar (excluding the results and loading pages) (Figure 4, A). The page also has a dynamic network attached to the mouse cursor. To understand the purpose of the web-server, the user can refer to the overview page (Figure 5), which provides a short description of the Titan pipeline and outlines the underlying methodology.

The tutorial page provides a detailed understanding of the steps and options required to run a Titan analysis or retrieve an existing network (Figure 6). The first section explains the type of studies that PLANET is best suited for. The second section focuses on the features of the input page whilst making a distinction between required and optional arguments for clarity purposes. The options for each input are explained in detail and recommendations for the optimal combination of parameters are also provided. The final section details the features of the network visualisation output. Images of the website and examples of outputs illustrate a typical user experience with the web-server.

The main input page to launch the Titan analysis is subdivided in two sections by hiding the optional parameters. The top container displays the upload buttons allowing the user to select files corresponding to the control and case datasets (Figure 7,A). The inputs offer the possibility of selecting multiple files corresponding to different samples of a same condition. If the user inputs files in text format, they need to specify the structure of the tab-delimited table (Refer to section 6.1 for details) by indicating whether OTUs represent rows or columns (Figure 5,C3). An email address can be provided for the user to receive a notification with a link to the output page once the job is over.

The user can display additional options by clicking on the “more options” link. This hide/show feature was implemented using JavaScript. The user can select the taxonomic rank to perform the analysis on. The default is set on “Genus” and can be changed to species, family, order, class, phylum or kingdom. Whilst the default analysis pipeline returns a network with undirected edges (Figure 16), the user has the option to generate a directed network instead

(Figure 17). Two sliders are provided for the numerical inputs: number of iterations (with a default of 2500) and significance level for the statistical test (default $\alpha = 0.05$). As two back-end scripts are available for Titan, each offering specific advantages (Refer to section 6.2 for details), the user can choose whether they want intermediate data (p-values, ϕ values and Δ values) to be returned or whether they are exclusively interested in the network features. The temporary files generated by the pipeline, such as filtered tables (Refer to section 6.1.1 for details), can also be retrieved. The server provides the option of using Pearson Correlation instead of Proportionality as the measure of inter-OTU dependency, although we recommend choosing Proportionality to avoid false positives (Refer to Section 5 for more details).

When the user clicks on the"Generate Network" Button at the bottom of the page, the Titan analysis tool is launched and the user is redirected to a loading page whilst the job is running. Each analysis has its own run-identifier (ID) which is used to generate a specific URL for the loading page. This page informs the user that they can access the output of the analysis by either bookmarking the loading page or by accessing the link sent to the provided email address once the job is over. The loading page self-refreshes every 10 seconds. When the analysis is complete, the loading page redirects the user to the result page with the same URL.

In addition to the Titan analysis pipeline that generates networks from metagenomic datasets, the website offers a tool that allows users to retrieve network outputs from a JSON file (Figure 7, B), which can be downloaded from a previous Titan analysis or be independently generated, as long as it complies to PLANET's format (Refer to Section 6.3 for details). This tool (discussed further in Section 4.6) requires two inputs to be provided by the user:

- A file in JSON format (Figure 22)
- Whether the file contains data from a directed or an undirected network.

To visualise a typical output produced by the network retrieval tool, the user can tick the "use

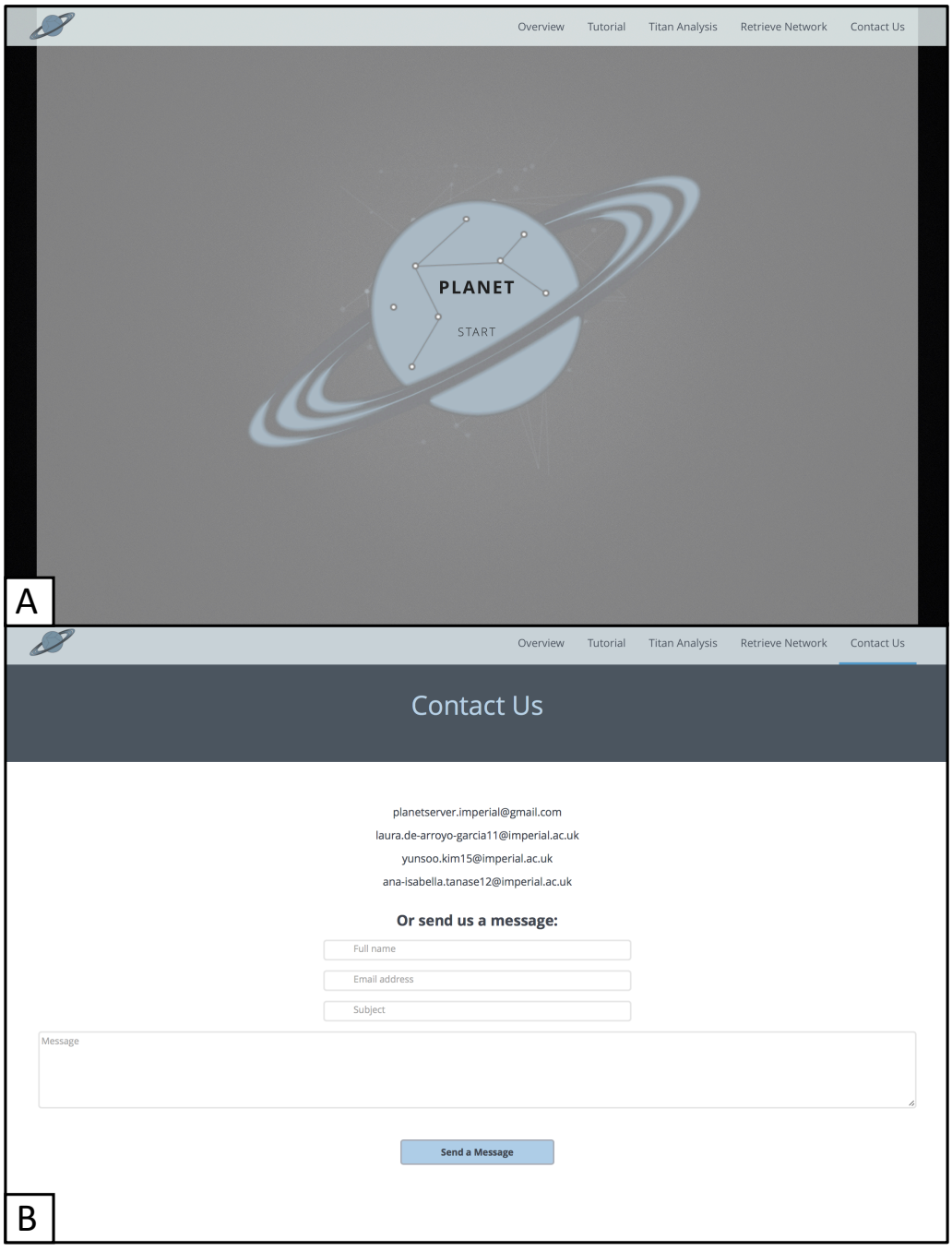


Figure 4: **PLANET Home Page and Contact Page.** A: Home page of the website. Users can directly access the analysis input-page by clicking on the central PLANET logo; B: contact page. Users can send a message to the PLANET developers by filling in the boxes provided on the page. Produced by A.T.

Figure 5: **Overview Page of the PLANET Website.** The light blue navigation bar is shown at the top. The page header is shown below. The main text is centred on the page within light grey containers. Produced by A.T.

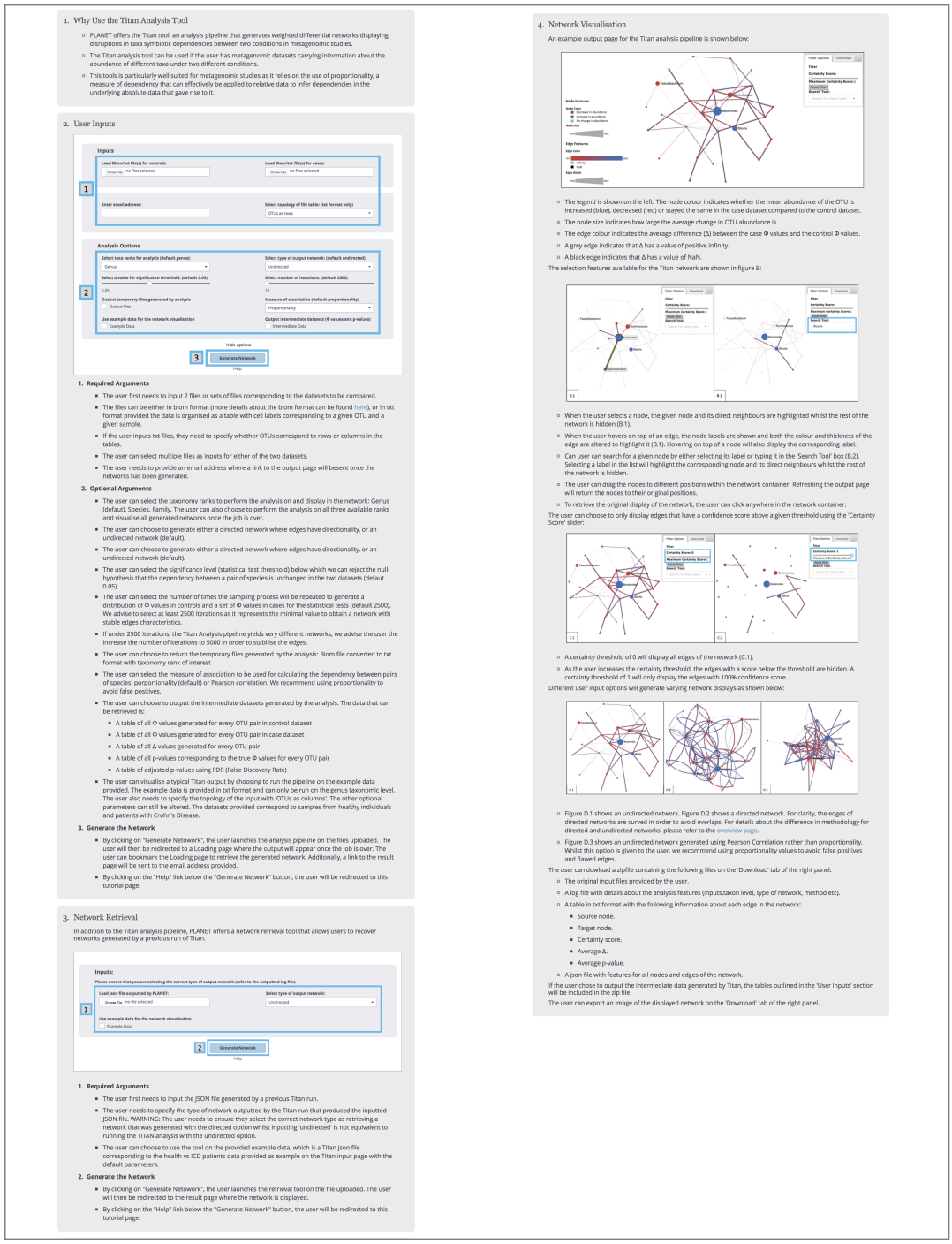


Figure 6: **PLANET Tutorial Page.** The figure shows the main page content of the tutorial. The tutorial is divided into 4 sections:"Why use the Titan Analysis Tool","User Inputs","Network Retrieval" and "Network Visualisation". Produced by A.T.

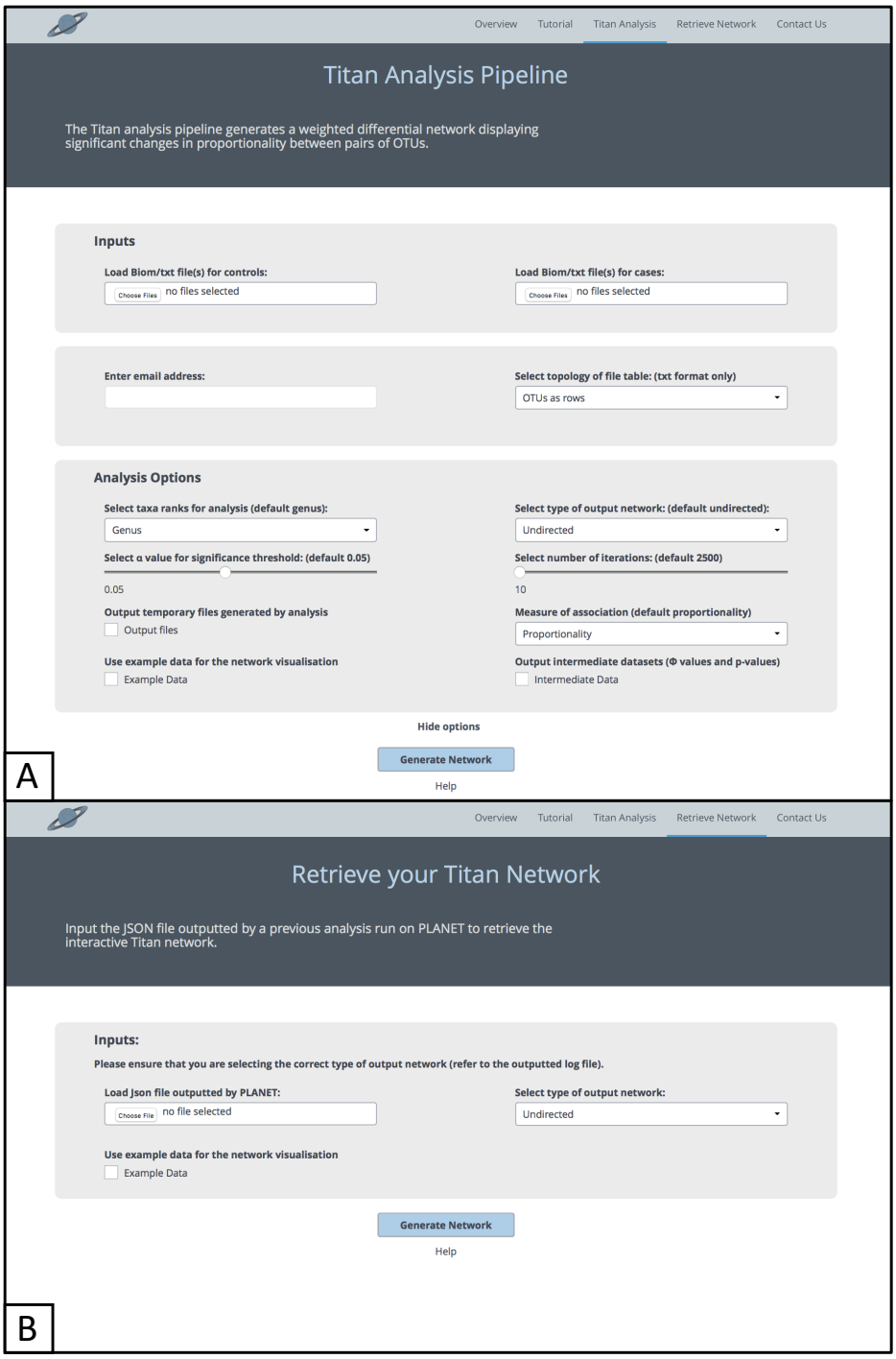


Figure 7: **PLANET Input Pages.** A: Input-page for the Titan Analysis Pipeline. The optional parameters are shown but can be hidden by the user by clicking on "Hide options"; B: Network-retrieval page. Produced by A.T.

example data for network visualisation" box and launch the task. The example data provided corresponds to an undirected weighted differential network displaying altered dependencies between healthy individuals and patients with inflammatory bowel disease (IBD) (*Morgan et al., 2012*).

Finally, the user can get in touch with the development team of the PLANET web-server through the contact page (Figure 4, B). Two options are available; the user can either send an email to one of the specified email addresses, or fill out the provided form, which will be forwarded to all PLANET developers. The user should ensure that all the required fields are filled, including email, full name and subject.

3.2 Pop-up flags (Y.K)

PLANET raises pop-up flags to alert the user regarding their requests. Concerning the contact page, when the user clicks on the "Send a Message" Button, the page refreshes with a pop-up appearing: "Your message was sent successfully. Thank you for your contribution. We will contact you as soon as possible.", for the user to know whether their message was delivered successfully to the PLANET server email (Figure 8, A).

The input page also generates error pop-up flags for an incorrect email structure. If the user provides an email address in the wrong format, a pop-up message appears next to the email input box so as to provide additional information regarding the correct structure. The user can follow these instructions to fix the error. If the user did not provide an '@' sign for the email address, the pop-up message suggests the user to do so (Figure 8, B). If the user did not provide the email domain (i.e. the address succeeding the '@' sign), the pop-up message informs the user that the email domain is missing (Figure 8, C). These pop-ups guarantee that the email address provided by the user is always in the right format. It also informs the user where the error may be regarding the email structure.

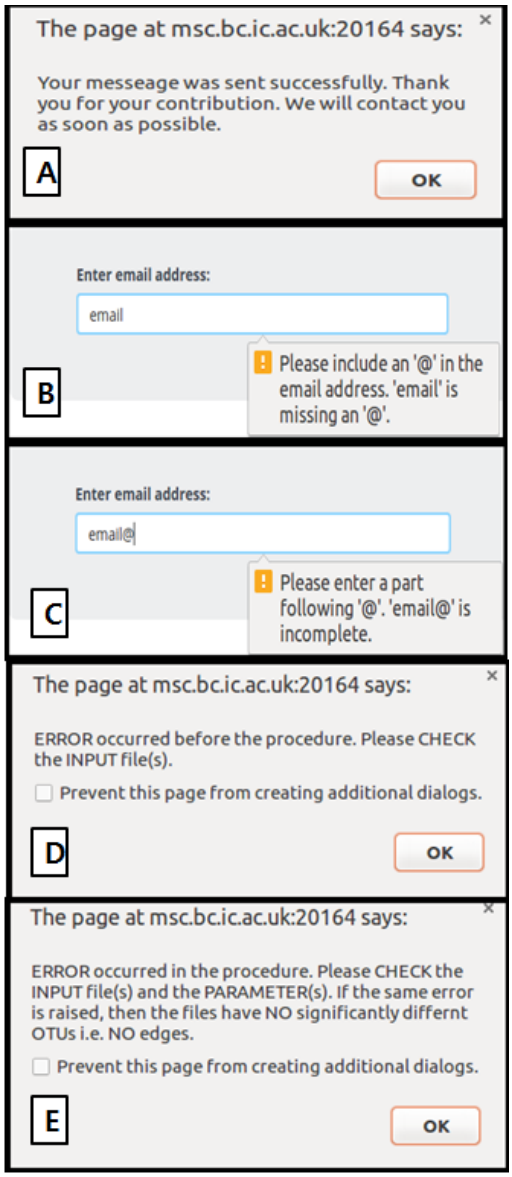


Figure 8: **PLANET Pop-ups.** This figure shows all the pop-ups used in PLANET. PANEL A - The pop up message from the contact page after the submission of a message. The page provides a pop-up with a message alertin the user that the submission has been completed. PANEL B and C show how the pop ups are used to notify the user for the incorrect email input. Panel B - The pop-up message which informs the user that the @ sign is missing. Panel C - The pop-up message when the user did not provide email domain for the address. Panel D and E show the error pop-ups from the result page. The page provides a pop-up with a message, which informs the user about the source of the error occurred during the analysis. Panel D - Error raised before proportionality calculation. The error occurs when the user does not provide a set of input files, or a right input file format or extension. Panel E - Error raised when the differential network does not contain any edges. The causes of this error are more complex than those of the first error. The input files may have no edges because they do not contain all the necessary information for the analysis, or the input files actually do not have any OTU interactions with significant changes between the two data sets. Produced by Y.K.

Pop-up boxes are used in the result page to provide simple descriptions of any errors that occurred during the Titan analysis. Currently two types of error are detected. Firstly, when the input files are in the wrong format or missing (Figure 8, D). The user receives the following message: "ERROR occurred before the procedure. Please CHECK the INPUT file(s)". The second type of error is raised when the differential network has no edges (Figure 8, E). This error is more complex than the first one. Although all cases occur when the differential network does not have any edges, it is hard to define what the actual source of the error may be. One case is when input files do not have any significantly altered OTU dependencies. In this situation, input files are in the correct format and carry the appropriate information, but the analysis cannot retrieve any pairs of OTUs with significant dependency changes in cases compared to controls. Alternatively, an error is raised when the input files are parsed without carrying all the needed information for the analysis. The user receives the following message: "ERROR occurred in the procedure. Please CHECK the INPUT file(s) and the PARAMETER(s)". Thus, if this error is raised, we recommend running the analysis with a larger number of iterations. If the same error is raised again, then the datasets have no significant changes in OTU dependencies i.e. no edges. Please refer to Section 6.5 for additional information on how the result page can detect errors that occurred in the analysis pipeline.

4 Output Interface (L.A.G.)

The output interface (otherwise known as network rendering or visualisation page) integrates several web-browser components, including CSS, HTML, JavaScript and three JavaScript libraries, Sigma.js, D3.js and jQuery (*Jacomy and Plique, 2015*), (*Bostock, 2015*), (*Methvin et al., 2016*).

This web page is consistent with the rest of the website, also built on the Bootflat template.

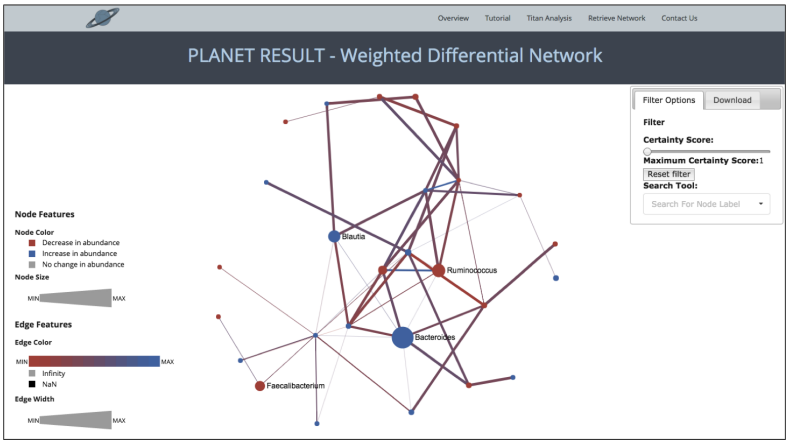


Figure 9: **Network rendering page**. A screenshot of the network rendering page was taken after running an analysis using an example IBD dataset. The output page has an extensible navigation bar at the top, and offers a centered network container, permanent legend on the left-hand side and a collapsible side panel containing a filter, reset button, search box, and download options. The network is generated by displaying OTUs as nodes and proportionality changes between pairs of OTUs as edges. **Node Colour -** Blue nodes indicate an increase in mean abundance in cases with respect to controls, whereas red nodes represent a decrease in mean abundance. **Node Size -** Magnitude of change in mean abundance, whether it is an increase or decrease. **Edge Colour -** Within a red-to-blue gradient, edge color indicates the difference between the true ϕ value from cases and the median of the ϕ bootstrap distribution. **Edge Width -** Representation of the certainty score, the proportion of iterations that presented significant changes in proportionality in cases. Produced by L.A.G.

It offers a clean, intuitive, easy to navigate structure, with collapsible navigation bar, graph container, legend and side panel (Figure 9). The web-based network visualisation tool is easy to use, interactive and supportive of dynamic behaviours, real-time, scalable (caters to datasets of varying complexity and dimensions), and compatible across web browsers and a range of electronic devices.

4.1 Sigma.js Implementation

Network retrieval was built using Sigma.js. Sigma is a specialised, graph-drawing JavaScript library. It facilitates publishing networks on the browser and also allows for network exploration and manipulation on the web environment. It renders the graph through the front-end of the web application by parsing the data generated through the back-end (*Jacomy and Plique, 2015*).

Getting started with Sigma is very straight-forward. This module is composed of two constituents, a graph model and a controller. The graph model is responsible for data collection and manipulation, whilst the controller offers a means for the rendering process and function implementation. For development purposes, the package must be fully installed locally. On the other hand, network rendering and external file parsing only require a local server (*Jacomy and Plique, 2015*).

After importing Sigma to the HTML template, we must have an HTML element displaying the Sigma container. Sigma must also be instantiated for the graph rendering process to be initialised. Sigma's rendering engine is very flexible and thus it can be instantiated in various ways (*Jacomy and Plique, 2015*). Any functions that update or alter network content, layout or spatialisation (such as filtering or clustering) must be defined within the Sigma instantiation and called from the script.

Sigma has several features that makes it a powerful package to work with. It offers a wide array of network-specific built-in functions, easing the implementation process. For instance, it automatically adapts the size and position of the elements it adds into the container accordingly, upon changes on container width and height. It also provides a means for zooming into and away from the graph, node and edge clicking and hovering, amongst others. It also has a flexible, pluggable architecture, by offering an extendable system that easily integrates plug-ins and renderers for additional functionality. This means that any additional features can be easily added in and customised (*Jacomy and Plique, 2015*).

All built-in functions and additional plug-ins used for constructing the final PLANET output page will be described in more detail in later sections of this manual.

4.2 Network Rendering

The ultimate goal of network retrieval is to provide a means to communicate valuable information to the user. The final graph reveals OTUs as nodes, and their corresponding, significant proportionality changes as edges, which link nodes that show changes in pair-wise dependencies. Node and edge styles (colour and size) are also fundamental to convey additional information.

Node colour, implemented using RGB (Red Green Blue) values, represents either an increase (blue) or decrease (red) in mean abundance for a specific OTU in the case samples with respect to control samples. If mean abundance remains the same in the two conditions, then the OTU is displayed as a grey node. Node size (ranging from 4px-20px) corresponds to the magnitude of that abundance change, whether it is a decrease or increase in mean abundance.

Concerning edges, the colour falls within a red-to-blue gradient, indicating the average difference between the case ϕ values and the median of the control bootstrap distribution, a value known as Δ. A grey edge represents a Δ value of infinity, whereas a black edge represents a value of 'not a number' (NaN) (Refer to Section 6.3 for further details). Edge width (ranging from 0.25px to 5px) represents the certainty score (from 0 to 1), which is the proportion of times the change in ϕ was deemed significant out of the total number of run iterations, i.e an indicator of the level of certainty on a proportionality change. This means that the thicker the edge, the higher the confidence in the proportionality change is. Refer to Figure 9 for a visual representation of the aforementioned features.

These features connect the results collected from the methodology to a network retrieval environment in the front-end. Sigma builds the output graph by parsing data contained in a JSON file, printed after the processing task is over in the back-end. JSON files are generated in Python using a manually curated JSON parser, which translates results produced by our analysis pipeline into comprenhensive data that Sigma can render in the form of a differential network. The JSON parser achieves the following:

- Mean abundance changes are calculated using Python and translated into RGB (Red Green Blue) values in order to indicate an increase or decrease in mean abundance using blue or red nodes, respectively.

- Magnitude of abundance change is calculated and translated into node size, with a minimum of 4px and a maximum of 20px.

- A red-to-blue RGB gradient represents Δ values. Δ readings are normalised and converted to RGB values, used to represent edge colour.

- Certainty scores are translated into edge width readings, with a minimum of 0.25px and a maximum of 5px.

To fulfill Sigma's requirements for successful graph display, nodes are also complemented with x and y coordinates, generated using NetworkX following a "Spring" layout (*Schult and Swart, 2008*). OTU names were included as node IDs and labels, and both size and colour were specified using the JSONparser described above. For additional information about the JSON file format, please refer to Section 6.3.

Sigma provides a JSON-specific renderer that acquires and provides structure to external JSON data. When this plug-in is loaded, nodes are incorporated into the container using their node IDs (OTU names) and coordinates. Edges are added in using source (origin node) and target (destination node) information, also provided in the JSON file. Node and edge style information is also implemented.

4.3 Network Features

A distinction is made between Sigma built-in functions and add-ons used for customised functionality. Built-in functions can be easily implemented by importing packages onto the HTML

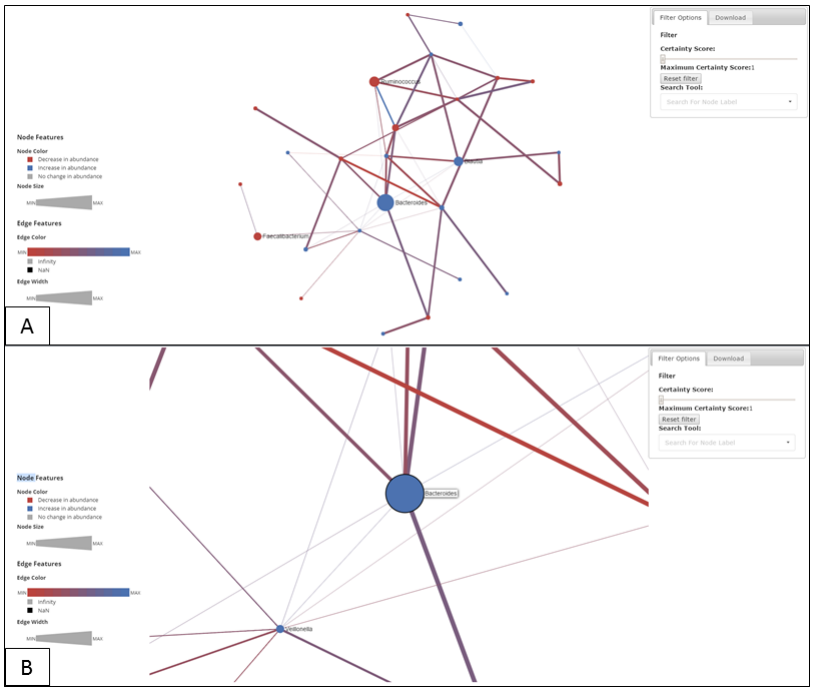


Figure 10: **Zoom Function**. The network rendering page allows the user to zoom into and away from the graph. A - Screenshot from the original output page, B - Zoomed in image of the same network. Produced by L.A.G.

page during development. Add-ons, however, may require the loading of a plug-in or renderer event, and additional function declarations for implementation and functioning. These add-ons comprise the set of additional features that were developed by our group to provide the user an extra level interactivity with the data.

4.3.1 Sigma built-in functions

- **Zoom:** Sigma.js provides the possibility to zoom into or away from the network when double clicking the network container (Figure 10, A and B).

- **Node Hovering:** Sigma distinguishes between "Default Node colour/Size" and "Node Hover colour/Size". When the user hovers over a node, node style is updated from default to its "hovered" style (which is customisable), making the node stand out. Its node label

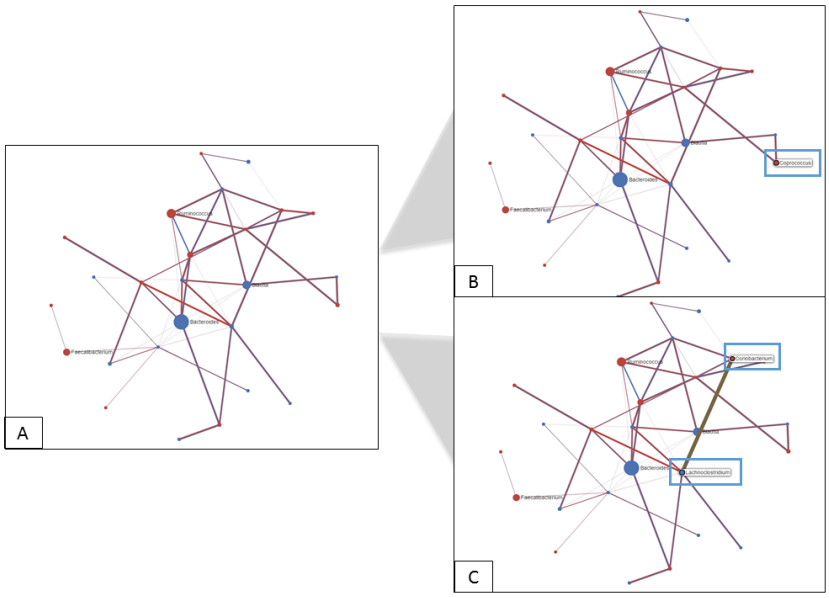


Figure 11: **Node Hovering Function.** Node hovering allows the user to dynamically detect nodes and edges by changing their current layout styles from their default settings. A - Original output image, B - Hovered nodes acquire a black borderline and a permanent label displaying the node's ID (OTU label). C - Hovered edges change color and width. Furthermore, source and target nodes acquire a black borderline and become attached to their node labels. Produced by L.A.G.

is also accessed upon hovering (Figure 11, A and B). When the user moves the cursor away, node style is refreshed and reverted to default.

- **Edge Hovering:** Similarly, when the user hovers over a edge, the node labels of its source and target nodes are displayed, and both nodes and the corresponding edge are displayed in their hovered styles (Figure 11, A and C). Hovering away from the label refreshes the script and renders these components back to their default settings.

4.3.2 Network Add-ons

- **Drag Nodes:** This function gives the user an opportunity to interact with the network, and move nodes and edges around during exploration (Figure 12, A and B). Nodes are initially added into the graph container by accessing a set of x and y coordinates for each node,

contained in the JSON file parsed by Sigma. The "Drag Nodes" plug-in allows for these coordinates to be altered and updated accordingly, and refreshed back to default. This feature manipulates the network frame and does not apply any changes to the original coordinates contained in the file.

- **Node-neighbours function:** This function facilitates retrieval of a node and its direct neighbours, in such a way that when the user selects a given node, the node as well as its direct neighbours and connecting edges become highlighted, whilst the rest of the network is dimmed (Figure 12, A and C).

- **Edge Filtering:** This function was built using an already existing Sigma plug-in as a template, and carefully modified to target edge width. Edge width represents certainty score, and the user is given the option to filter away edges according to their desired confidence threshold. Certainty scores range from 0 to 1, and as the certainty score increases, only those edges over the set threshold will be highlighted, whilst the remainder will be hidden. This function is also linked to a reset button that retrieves the original network display (Figure 13).

- **Node Searching:** A search function allows the user to search for a specific OTU by typing its name in a search box, or to retrieve a list of all significant OTUs present in the final output. When the user selects the name of a specific constituent, it triggers a node search by node label. If found, the node in question is highlighted and the "Node-neighbours" function is called, featuring the selected node's direct neighbours and dimming the remainder of the graph (Figure 12, C).

- **Network Display Options:** Differential networks generated from proportionality calculations can be displayed as directed or undirected graphs. This option is provided on the input page prior to running the analysis, as it determines the back-end methodology to be

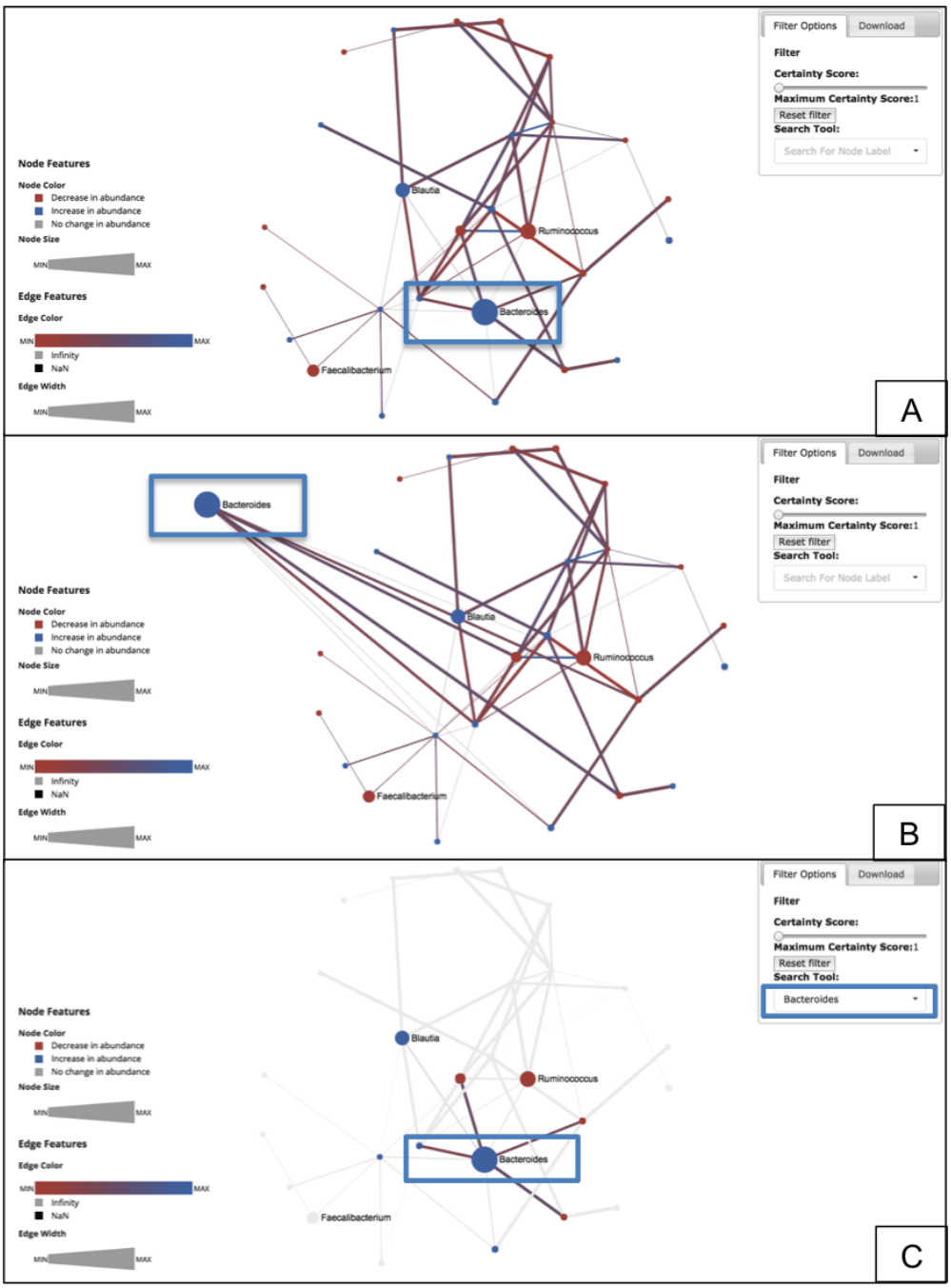


Figure 12: **"Drag Nodes" and "Node-neighbours" Functions**. The user can drag nodes around the graph container and update their positions, and retrieve a node and its direct neighbours by double clicking on a specific OTU. A - Original Image. B - A node is selected and dragged away from its original position. The new position updates the node's current coordinates. C - A node is clicked and its direct neighbours and corresponding edges are highlighted as the remainder of the graph is dimmed. This function is also initialised when the user searches for a specific node (see side panel). Produced by L.A.G.

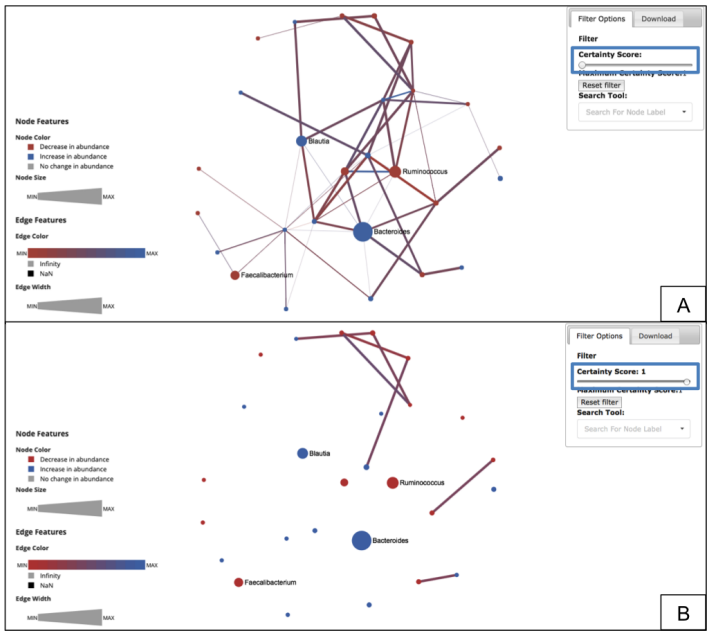


Figure 13: **Filter Function**. The user can filter edges according to their width, which corresponds to their certainty score. Certainty score ranges between 0 and 1, to represent the weighted nature of the differential networks. A - Network output set at a certainty threshold of 0. B - Network output set at a certainty threshold of 1. Produced by L.A.G.

applied. This is described in detail in Section 5 of this manual. Selecting the "directed" option retrieves a network that displays curved edges (to prevent edge overlap) with an arrowhead. Undirected graphs have edges displayed as straight lines. PLANET also offers the user the possibility to run a directed or undirected Pearson Correlation analysis for sole comparison purposes. However, when working with abundance data, it is highly recommendable to run a proportionality analysis instead (refer to Section 1.3). A visual representation of these three types of data display is shown in Figure 14.

4.4 Legend

A comprehensive legend is permanently provided on the bottom left corner, and includes a concise summary of both edge and node colour, as well as edge and node size information

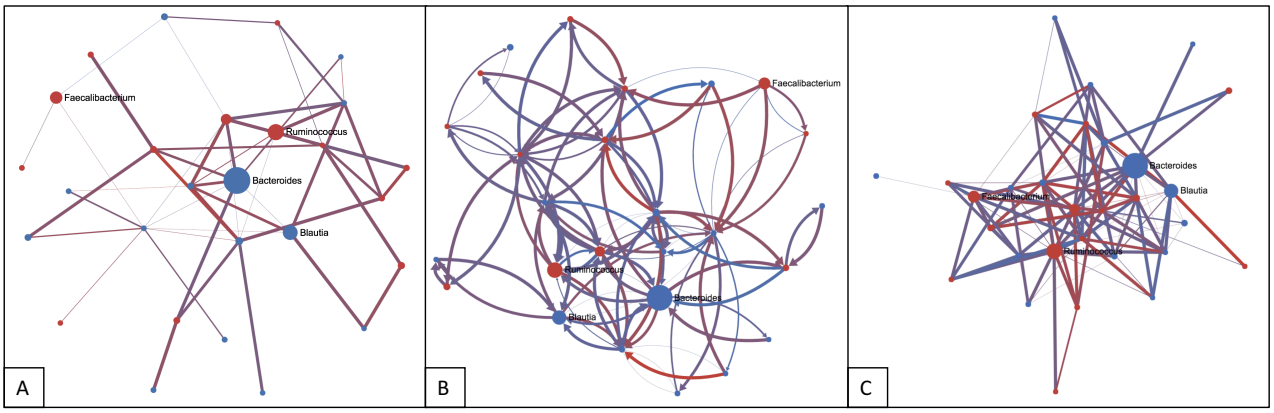


Figure 14: **Network Displays.** Our pipeline can generate both directed or undirected networks, both for proportionality studies and Pearson correlation. Directed and undirected networks differ in the methodology applied to generate the networks (refer to Section 5 for additional information). A - Undirected network, straight lines and no arrowhead . B - Directed network, curved edges and arrowheads. C - Undirected correlation network, straight lines and no arrowhead. Produced by L.A.G./A.T

(Figure 15, A).

This component of the output page was built using D3.js. D3 is a data-driven JavaScript package, used for building any kind of graphical visualisation framework. It transforms data in the web document using HTML, CSS and Scalable Vector Graphics (SVG) components through the use of selectors and operators. Selectors use CSS tags to grab elements from the HTML page. Once a tagged element is found and selected, D3 operators manipulate such elements by appending or adding attributes, such as properties, styles or HTML content (*Bostock, 2015*).

The legend was built by first defining a legend-specific div element on the HTML page and then appending and styling shapes onto the legend element. Node and edge width graphics were built using SVG paths, and the gradient graphic was built upon an existing template provided by D3.

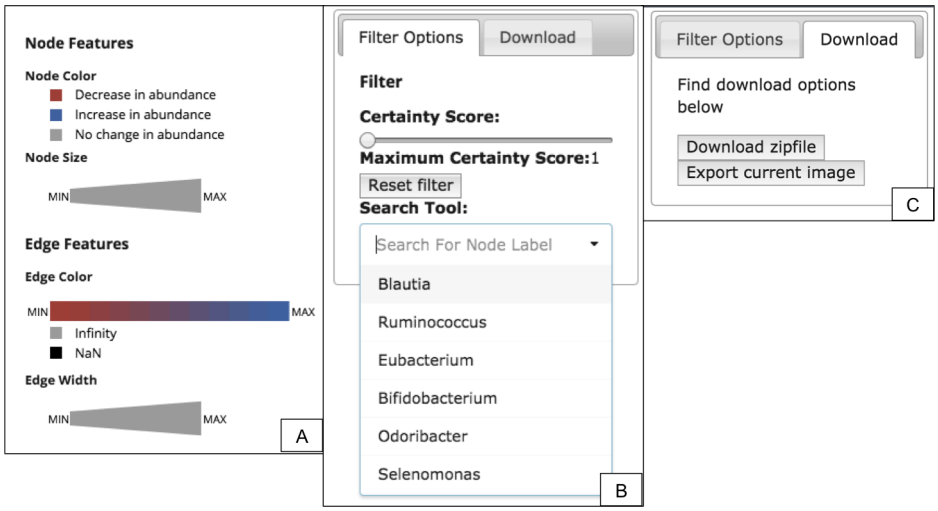


Figure 15: **Legend and Side Panel** The output page is composed of a permanent legend and a collapsible side panel. A - Legend containing a comprehensive summary of the networks' features. These are node colour, node size, edge colour and edge width. B - Side panel, which contains a certainty score filter, reset button to reset the network filter back to default, and a search box to facilitate node search and retrieval. The "search tool" box is composed of a search box and a customisable drop-down list containing all node labels (OTUs) for a specific run. C - Side panel, download options include downloading a zip archive containing a text output file, a log file with a summary of the run settings, a JSON file and the original files submitted by the user. Furthermore, "export current image" allows downloading a PNG image of the graph container. Produced by L.A.G.

4.5 Side Panel

The output page was supplemented with a collapsible side panel, implemented using jQuery (Figure 15, B and C) (*Methvin et al., 2016*). This side panel includes, on its first tab, the filter slider, reset button, and "Search Tool" box (Figure 15, B). It also offers a set of download options (Figure 15, C).

The functionality of the filter slider is described in Section 4.3.2. A reset button is also bound to it, allowing the user to refresh the container and return nodes and edges back to default (Figure 13).

The "Search Tool" box was built using jQuery and includes both an on-select search box and a drop-down menu (Figure 15, B). The drop-down function is bound to Sigma in such a way that it accesses all node labels passed on from the JSON file. The drop down menu is

thus customised for each independent network rendering event. This search box is bound to the Node searching function described above.

Download options are provided for the user to gain access to all files generated throughout data analysis (Figure 15, C). Users are provided with a zip archive containing a text file, a JSON file, the original files provided by the user for analysis, and a log file containing summarised run features. The zip archive may also contain temporary files and intermediate files, if the user requested these prior to the analysis. Additional information regarding the content of these files can be found in Section 6.3. These files can be used to retrieve the network output through PLANET's "Retrieve Network" tab (discussed next), or to run further analysis using additional network manipulation and analysis softwares, such as Cytoscape (*Shannon et al., 2003*) and others. Please refer to Laura de Arroyo's personal report for additional information on downstream, stand-alone network visualisation and manipulation softwares.

In addition to this, the "Download Current Image" button allows the user to download a PNG (Portable Network Graphics) image of the network container (Figure 15, C). This function was implemented using the "Snapshot" renderer event provided by Sigma. The user is free to save the original network layout or to make changes to the graph (zoom in or out, drag nodes, select or filter a specific set of nodes) and save these updates as an image.

4.6 Retrieving Networks

As previously outlined in Section 3.1, PLANET offers an additional tool, in conjunction to the Titan analysis pipeline: "Retrieve your Network". This feature allows the user to retrieve any network output from a JSON file. This file may have been generated after running a metagenomic analysis using Titan, or it may be an independently generated JSON file. If the user wishes to input their own JSON file, they should ensure that the latter is structured in the correct format (refer to Section 6.3 for additional information on the JSON output format).

For logistics purposes, users are also required to specify whether the JSON file data corresponds to a directed or undirected network. Directed networks contain twice the number of edges in a JSON file (two edges per interaction), whereas network retrieval for an undirected graph generates only one link per OTU pair. Furthermore, directed and undirected graphs are displayed with different layout settings (discussed in section 4.3.2). As a result, if a directed network is retrieved as an undirected graph, edges will overlap and results will be be misleading.

It is important to bear in mind that Network Retrieval is only a network visualisation tool and is not the equivalent to, or a replacement for the Titan pipeline. This is because the methodology to retrieve directed vs. undirected networks differs in confidence score computation. If the user runs an initial analysis as either directed or undirected, and wishes to explore the opposite option, Titan should be run again with the updated settings. Details about the methodology and how it is applied in these two scenarios is described in detail in Section 5.

5 Detailed Methodology (A.T)

In the previous section, features and functions of the network rendering page are described in detail. The data analysis methodology generating the outputted graphs (implemented in Python) relies on a resampling process, which allows the computation of certainty scores for the network edges.

This section covers this methodology in detail, explaining how both undirected and directed differential networks are retrieved. The methodology generating undirected differential networks is provided below and illustrated in Figure 16:

- OTUs for which all samples have abundance values of zero are initially discarded from the analysis pipeline.
- For each OTU pair present in both datasets:

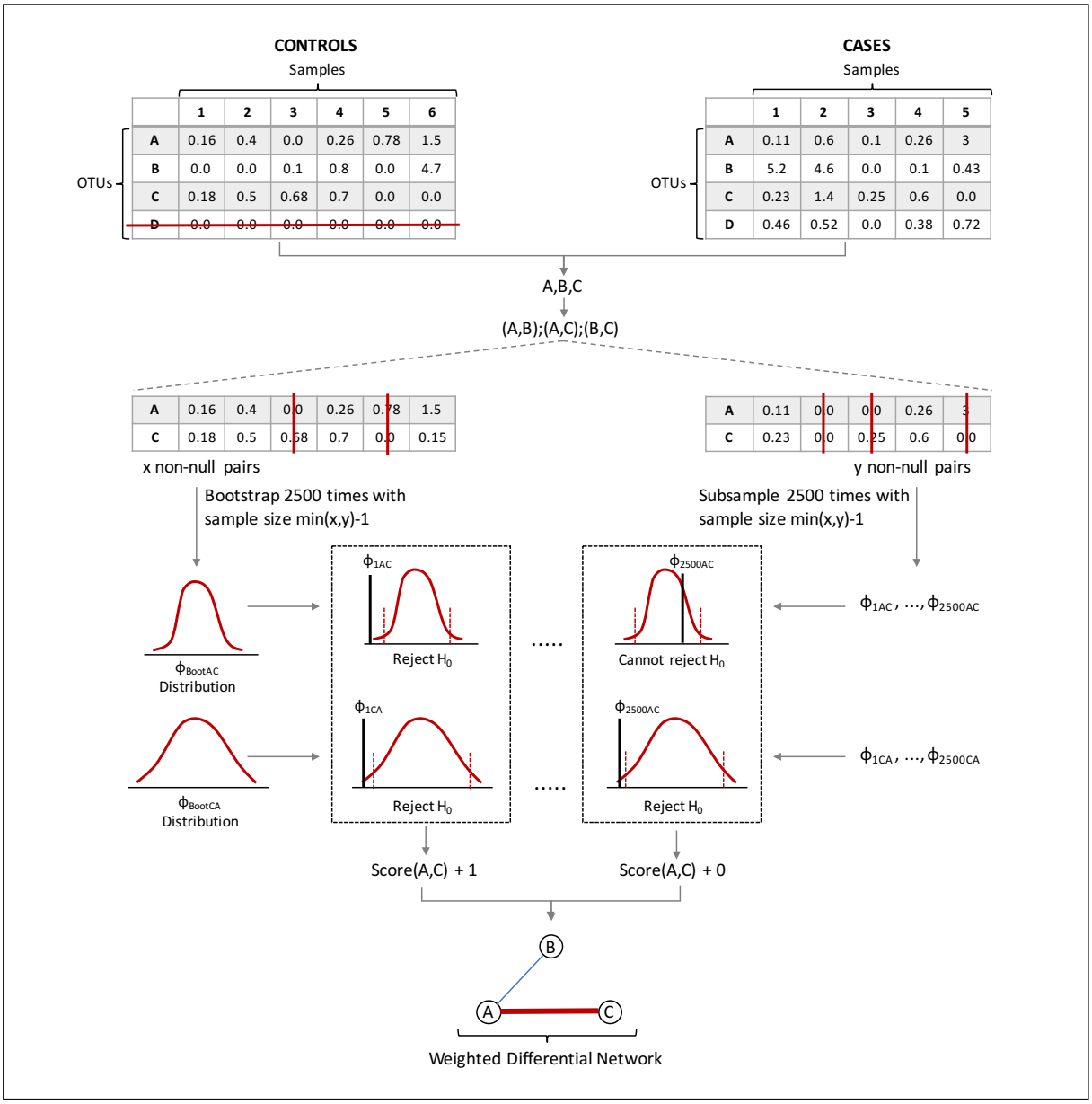


Figure 16: **Summary of the Comparative Analysis Pipeline for Undirected Networks.** The control and case input tables with samples as columns and OTUs as rows are shown at the top of the Figure. OTUs crossed in red have all-zero abundances and are discarded. The analysis below is performed for every OTU pair ((A,B), (A,C) and (B,C)) but is only illustrated for (A,C). Samples crossed in red have abundance values of zero and are culled. The bootstrapping and ϕ calculation process on the control set is repeated 2500 times to generate the directed null distributions shown as red curves. The critical points of the distributions are shown as red dotted lines. The subsampling and ϕ calculation process on the case set is repeated 2500 times to generate the directed true ϕ values shown on the right. If the true ϕ value is outside the zone defined by the critical points, the null hypothesis (H_0) is rejected. The null hypothesis needs to be rejected for both directed ϕ values of an iteration for the edge score to be incremented. Nodes of the network represent OTUs. The thicker the edge, the higher the certainty score. Blue edges have a negative Δ, whilst red edges have a positive Δ. Produced by A.T.

1. The minimum number (n) of non-zero samples between the control and case datasets is computed.
2. Samples for which either or both abundance values are zero-valued are culled from the control and case datasets.
3. The remaining non-zero samples in the control set are randomly sampled with replacement to generate a bootstrapped set of length (n-1) for the OTU pair.
4. Due to the asymmetry of the proportionality measure, two directed proportionality values ($\phi_{boot(A,B)}$ and $\phi_{boot(B,A)}$) are computed for the bootstrapped set.
5. The non-zero samples in the case dataset are randomly sampled without replacement to generate a subsample of length (n-1) for the OTU pair.
6. The directed proportionality values ($\phi_{true(A,B)}$ and $\phi_{true(B,A)}$) are computed for the subsample.

- Steps 3-4 are repeated 2500 times to create a bootstrap null distribution. Steps 5-6 are repeated 2500 times, and each true ϕ value generated is compared to the null distribution for computation of the certainty score.

- A p-value, corrected for multiple-testing using False Discovery Rate (FDR) (*Benjamini and Hochberg, 1995*) is calculated for each true ϕ value. For undirected networks, if both directed p-values of a given iteration are smaller than the significance threshold ($\alpha = 0.05$), the change in proportionality is considered significant and the edge score is incremented. For directed networks, each directed ϕ value is individually compared to its null distribution (Figure 17).

- The final edge certainty score is the proportion of subsampling iterations for which the change in ϕ was shown to be significant.

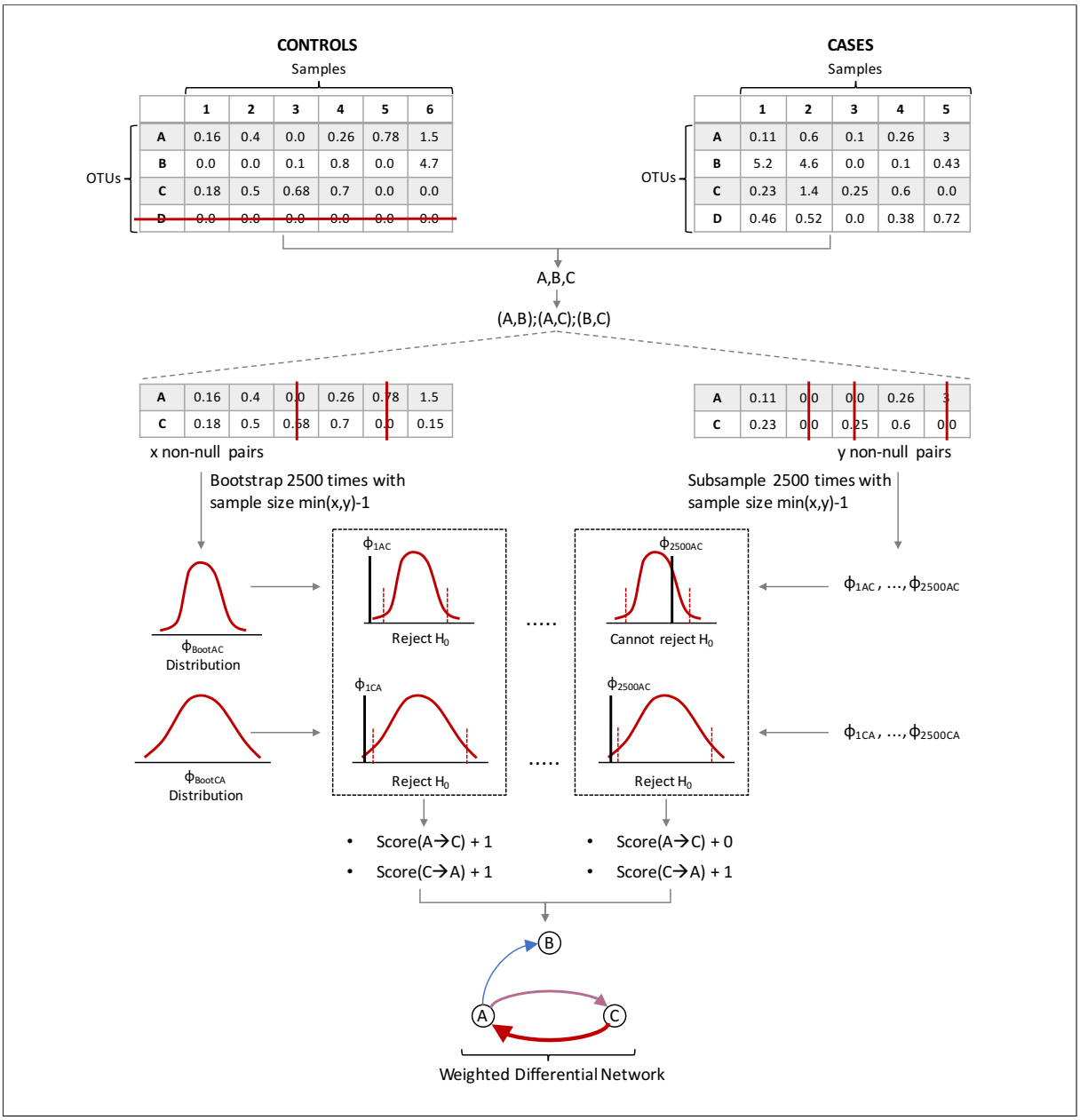


Figure 17: **Summary of the Comparative Analysis Pipeline for Directed Networks.** The control and case input tables with samples as columns and OTUs as rows are shown at the top of the Figure. OTUs crossed in red have all-zero abundances and are discarded. The analysis below is performed for every OTU pair (A,B),(A,C) and (B,C) but is only illustrated for (A,C). Samples crossed in red have abundances of 0 and are culled. The bootstrapping and ϕ calculation process on the control set is repeated 2500 times to generate the directed null distributions shown as red curves.The critical points of the distributions are shown as red dotted lines The subsampling and ϕ calculation process on the case set is repeated 2500 times to generate the directed true ϕ values shown on the right. If the directed true ϕ value is outside the zone defined by the critical points, the null hypothesis (H_0) is rejected. Nodes of the network represent OTUs. The thicker the edge, the higher the certainty score. The edge colour-gradient indicates the value of Δ: blue edges have a negative Δ, whilst red edges have a positive Δ. Produced by A.T.

- In addition to the certainty score, the average change in ϕ between the two datasets is also computed by taking the average difference (Δ) between the true ϕ values and the median of the null distribution. If the Δ value is negative, the true ϕ values in the case dataset are on average smaller than the median of the null distribution from the control set, and therefore the two OTUs are more proportional in the case dataset than the control dataset (the closer ϕ is to 0, the stronger the proportionality). If the Δ value is positive, the OTUs are less proportional in the case dataset than the control dataset.

Users are given the option to measure dependencies between OTU pairs using Pearson's correlation instead of proportionality for comparison purposes. Two variables (x,y) are said to be correlated if they fit the following equation:

$$y = mx + c \tag{3}$$

Where m is the slope of the line of y plotted against x, and c is the y-intercept. In comparison, (x,y) are said to be proportional if they fit the following equation:

$$y = mx \tag{4}$$

Where m is the slope of the line. Therefore, proportionality is a more stringent measure than correlation, as variables that are proportional are also necessary correlated, but the opposite does not hold. By using correlation, the user might gain some edges that represent true correlated OTUs that are not proportional. However, for the reasons outlined in Section 1.3, using correlation will yield a large amount of false-positive edges that are not correlated in the underlying absolute data. Figure 14 illustrates how using Person's correlation will output a denser network than correlation. This effect outweighs the benefit gained from displaying correlated edges, as

true positives and false-positives cannot be differentiated in the networks. The users are thus advised to select proportionality as the measure of inter-taxa dependency.

6 Back-End Structure

6.1 Inputs (Y.K)

In order to implement the methodology discussed in the previous section, Titan can accept two file extensions as inputs, ".biom" and ".txt.". We developed parsers for each file formats. Titan has two BIOM file parsers and one text file parser which are discussed in the following subsections. Also, Titan allows the user to input multiple files to represent a common dataset. Example input data are presented here as well to help the user understand what PLANET expects (Figures 18 and 19). We recommend the users to check if their input files follow the accepted formats and carry the right information. Also, the users should also check the parameters they chose for the analysis as some parameters are related to parsing the data, and if the wrong parameters are given, the input files can produce an error during the pipeline even though they contain the right information.

6.1.1 BIOM File Parser (Y.K)

".biom" is the file extension for the Biological Observation Matrix (BIOM) format and is Titan's default input file format (*McDonald et al., 2012*). The BIOM file is often used in biological studies to store numerical data such as relative abundance data (*McDonald et al., 2012*). It can contain descriptions of the biological samples used in the studies and any additional information. Thus, many tools for metagenomics, such as Qiime, use the BIOM file format to store taxonomic assignments of OTUs and their corresponding abundance data (*Kuczynski et al., 2012*).

We decided to use the BIOM file format as the default input file format because of its wide usage in metagenomics. Details of the BIOM format can be found at http://biom-format.org/.

The BIOM file format has multiple versions, and each of them has different ways of storing the data, either in JSON or HDF5 formats. The Titan pipeline accepts all BIOM file formats (Fig 20). However, the analysis can still raise an error if the inputted BIOM files do not contain the right information, such as taxonomic assignments. Also, the BIOM files with only the binary data (i.e. 0 and 1 as the abundance data) will cause an error in the pipeline because the proportionality method involves calculation of logarithm, and logarithm of zero is not defined. Thus, zeros are removed from the calculation, which leaves the data to be just one value, 1, and makes the analysis not meaningful because variance is always 0. In fact, text file inputs with the similar properties (i.e. binary abundance data and no taxonomic descriptions) can produce the same errors. Please check the content of the input files before the submission. Refer to Figure 20 for the accepted BIOM file formats. Detail of the accepted text file format is described in the Section.

Titan converts the input BIOM files into a human readable table format. The conversion filters OTUs at a specific taxon level that the users chose for their analysis, and then extracts the abundance data of the remaining OTUs. The converted table is used for the proportionality calculation between pairs of OTUs.

Titan currently uses two BIOM file parsers; while one is more memory efficient because it writes and uses temporary files, the second one is more time efficient. Both parsers generate the same output, which is a table containing filtered OTUs and corresponding abundance data. They both use the BIOM conversion tool from the biom-format python package (*McDonald et al., 2012*).

Concerning the parser that generates temporary files, the BIOM conversion package is used as a command line operation. The “biom convert” function from the package is used to convert

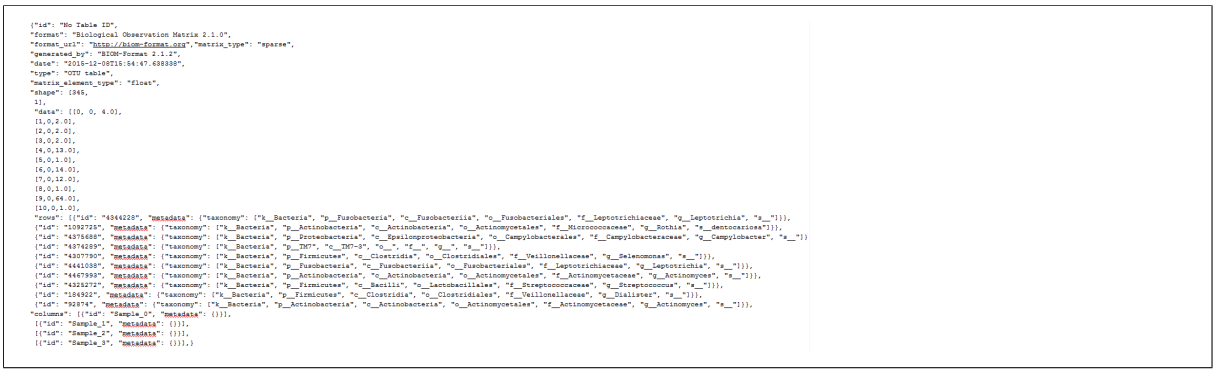


Figure 18: **PLANET Example BIOM file.** The example file shows the structure of the accepted BIOM file as well as its information. The example data is in JSON format. Thus, it contains information regarding the file itself such as BIOM file version. The necessary fields of information for Titan analysis are the data dictionary, which contains abundance data, and the rows and the columns dictionaries. The data dictionary should have lists, where each list represents a data point. The first index of the list represents a row index (OTU ID) and the second index is a column index (Sample ID) for the data. The third index of the list is the relative abundance. Rows dictionary contains OTU id and its corresponding taxonomic assignment for each row. The column dictionary has sample IDs and their metadata. The metadata for sample IDs is not necessary so it can be null (not provided) as shown in this figure. HDF5 version of BIOM file contains the same data but in different structure; two data types, observation and sample, are necessary. It is not shown in this figure because the content of the raw HDF5 file in text is not a human readable format. The content of it can only be accessed using a parser designed for it. Produced by Y.K.

table, which is then loaded to Python for taxon filtering. The filtered table is then saved as a temporary file, which is reloaded to Python for combining duplicate OTUs and removing samples with only zeros. The users can download the taxon filtered table of their input BIOM files, if they selected this parser. The taxon filtered table contains the OTU IDs replaced by their taxonomic assignments at a specific taxon level. Thus, the table can provide the number of OTUs from the input BIOM files with their taxonomic assignments defined at the taxon level.

On the other hand, the parser with no temporary files uses the BIOM conversion package as a Python module. The BIOM file is loaded directly to Python and taxon filtering is done using the metadata of the loaded BIOM file. Then, duplicate OTUs and samples with only zeros are handled in the same way that those are handled by the parser with temporary files .

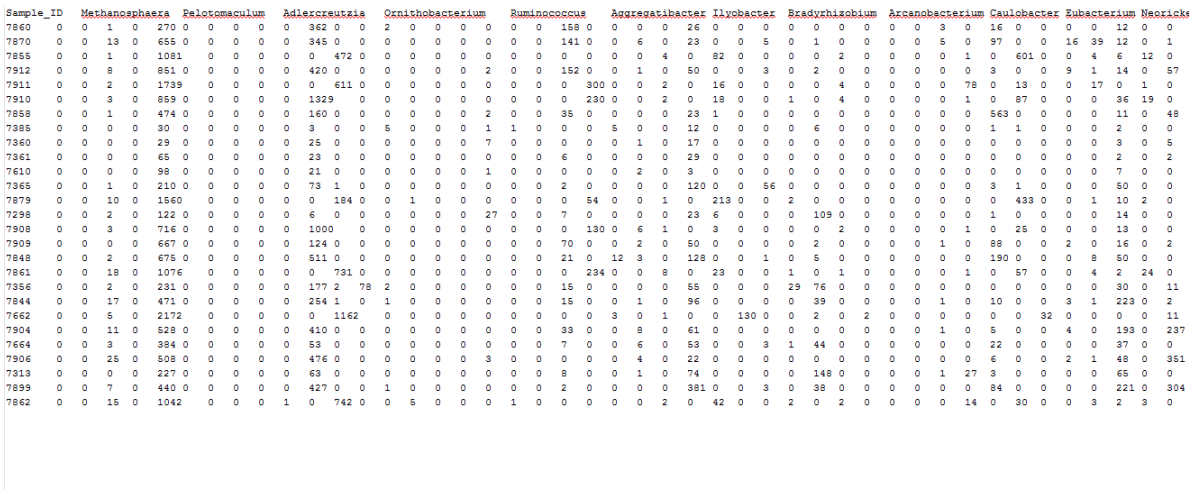
Both parsers can accept multiple input files. All the input files are parsed individually. Then, the panda data frame outputs of the parsers are merged into a table to represent one set of data.

The merged data frames are saved as human readable table files, if the parser with temporary files is selected for the analysis. Concerning the parser with temporary files, one temporary table is generated per each input file. Thus, for multiple input files, the parser with temporary files results in a much longer analysis than the parser without temporary files.

In theory, the parser with temporary files is appropriate for larger input files because the parser without temporary files keeps the original BIOM files in memory. However, the parsers do not contribute hugely to the overall efficiency of the pipeline, because the overall efficiency is mainly dependent on the proportionality calculation. Thus, the users should select the parser with temporary files if and only if they are interested in downloading the temporary files. Thus, Titan has the parser without temporary files as the default parser. For multiple input files, if the users are interested in acquiring merged tables of the input files, then they should select the parser with temporary files.

6.1.2 Text File Parser (A.T/Y.K.)

Some applications that calculate abundance data from 16S rRNA fasta files output a text file containing a table of abundance data (*Kuczynski et al., 2012*). Thus, for the users with human readable table text files, PLANET incorporates a parser for the text file table format. The input text file should have a tab-delimited table with OTUs as rows or columns, and their abundance data for multiple samples (Figure 19). The text file parser implements the last few components of the BIOM file parser with temporary files. The input text file is treated as a taxon filtered table, which is then loaded to a python environment for combining the duplicate OTUs and removing samples with all-zero abundances. The user needs to specify the structure of the table on the input page by indicating whether OTUs represent rows or columns. For tables with OTUs as rows, the Pandas "read_csv" function (*McKinney, 2010*) is applied to the inputs in order to read the files into data frames. For tables with OTUs as columns, the data frame generated from



Sample_ID	Methanosphaera	Pelotomaculum	Adlercreutzia	Ornithobacterium	Ruminococcus	Aggregatibacter	Ilyobacter	Bradyrhizobium	Arcanobacterium	Caulobacter	Eubacterium	Necricks
7860	0 0 1 0 270	0 0 0 0	0 362 0 0	2 0 0 0 0	0 0 158	0 0 0 26	0 0 0	0 0 0 0	0 0 3 0	16 0 0	0 0 12	0 0
7870	0 0 13 0 655	0 0 0 0	0 345 0 0	0 0 0 0 0	0 0 141	0 6 0 23	0 0 5	0 1 0 0	0 0 5 0	97 0 0	16 39 12	0 1
7855	0 0 1 0 1081	0 0 0 0	0 472 0	0 0 0 0 0	0 0 0	0 4 0 82	0 0 0	0 2 0	0 0 0 1	0 601 0	0 4 6	12 0
7912	0 0 8 0 851	0 0 0 0	0 420 0	0 0 0 2	0 0 152	0 0 1	0 50 0	0 3 0	2 0 0	0 0 0	0 3 0	0 9	1 14	0 57
7911	0 0 2 0 1739	0 0 0 0	0 611 0	0 0 0 0 0	0 0 300	0 0 2 0	16 0 0	0 0 4	0 0 0 78	0 13 0	0 17	0 1
7910	0 0 3 0 859	0 0 0 0	0 1329 0	0 0 0 0 0	0 0 230	0 0 2 0	18 0 0	1 0 4	0 0 0 1	0 87 0	0 0 36	19 0
7858	0 0 1 0 474	0 0 0 0	0 160 0 0	0 0 0 2	0 0 35	0 0 0 23	1 0 0	0 0 0	0 0 0 0	563 0 0	0 0 11	0 48
7385	0 0 0 0 30	0 0 0 0	0 3 0 0	5 0 0 0 1	1 0 0 0	5 0 0 12	0 0 0	0 6 0	0 0 0 0	1 1 0	0 0 2	0 0
7360	0 0 0 0 29	0 0 0 0	0 25 0 0	0 0 0 7	0 0 0 0	0 1 0 17	0 0 0	0 0 0	0 0 0 0	0 0 0	0 0 3	0 5
7361	0 0 0 0 65	0 0 0 0	0 23 0 0	0 0 0 0 0	0 0 6	0 0 0 0	29 0 0	0 0 0	0 0 0 0	0 0 0	0 0 2	0 2
7610	0 0 0 0 98	0 0 0 0	0 21 0 0	0 0 0 1	0 0 0 0	0 2 0 3	0 0 0	0 0 0	0 0 0 0	0 0 0	0 0 7	0 0
7365	0 0 1 0 210	0 0 0 0	0 73 1 0	0 0 0 0 0	0 0 2	0 0 0 0	120 0 0	56 0 0 0	0 0 0 0	3 1 0	0 0 0	50 0 0
7879	0 0 10 0 1560	0 0 0 0	0 0 184 0	0 1 0 0 0	0 0 0	54 0 0 1	0 213 0	0 2 0	0 0 0 0	0 0 0	433 0	0 1	10 2	0
7298	0 0 2 0 122	0 0 0 0	0 6 0 0	0 0 0 0 27	0 0 7	0 0 0 0	23 6 0	0 0 109	0 0 0 0	0 0 1	0 0 0	0 14	0 0
7908	0 0 3 0 716	0 0 0 0	0 1000 0	0 0 0 0 0	0 0 0	130 0 6	1 0 3	0 0 0	0 2 0	0 0 0	1 0	25 0	0 0	13 0	0
7909	0 0 0 0 667	0 0 0 0	0 124 0 0	0 0 0 0 0	0 0 70	0 0 2 0	50 0 0	0 0 2	0 0 0 1	0 88 0	0 2 0	16 0	2
7848	0 0 2 0 675	0 0 0 0	0 511 0 0	0 0 0 0 0	0 0 21	0 12 3 0	128 0 0	1 0 5	0 0 0 0	190 0 0	0 8	50 0	0
7861	0 0 18 0 1076	0 0 0 0	0 0 731 0	0 0 0 0 0	0 0 0	234 0 0	8 0 23	0 0 1	0 1 0 1	0 0 0	1 0	57 0	0 4	2 24	0
7356	0 0 2 0 231	0 0 0 0	0 177 2 78	2 0 0 0 0	0 0 15	0 0 0 0	55 0 0 0	29 76 0	0 0 0 0	0 0 0 0	0 0 30	0 11
7844	0 0 17 0 471	0 0 0 0	0 254 1 0	1 0 0 0 0	0 0 15	0 0 1 0	96 0 0 0	0 39 0	0 0 0 1	0 10 0	0 3 1	223 0	2
7662	0 0 5 0 2172	0 0 0 0	0 0 1162	0 0 0 0 0	0 0 0	0 3 0 1	0 0 130	0 0 2	0 2 0 0	0 0 0	32 0 0	0 0	11
7904	0 0 11 0 528	0 0 0 0	0 410 0 0	0 0 0 0 0	0 0 33	0 0 8 0	61 0 0 0	0 0 0 0	0 0 1 0	5 0 0	4 0	193 0	237
7664	0 0 3 0 384	0 0 0 0	0 53 0 0	0 0 0 0 0	0 0 7	0 0 6 0	53 0 0	3 1 44	0 0 0 0	0 0 22	0 0 0	0 37	0 0
7906	0 0 25 0 508	0 0 0 0	0 476 0 0	0 0 0 3	0 0 0	0 4 0 22	0 0 0	0 0 0 0	0 0 0 6	0 0 2	1 48	0 351
7313	0 0 0 0 227	0 0 0 0	0 63 0 0	0 0 0 0 0	0 0 8	0 0 1 0	74 0 0 0	0 148 0	0 0 0 1	27 3 0	0 0 0	65 0	0
7899	0 0 7 0 440	0 0 0 0	0 427 0 0	1 0 0 0 0	0 0 2	0 0 0 0	381 0 0 3	0 38 0	0 0 0 0	84 0 0	0 0 221	0 304
7862	0 0 15 0 1042	0 0 0 1	0 742 0	0 5 0 0 0	1 0 0	0 0 0 2	0 42 0	0 2 0 2	0 0 0 14	0 30 0	0 3 2	3 0

Figure 19: **PLANET Example text file.** This figure is an example of the accepted text file. The text file is a tab- delimited file with OTU ids as columns. Titan also accepts a text file with OTU ids as rows. The text file has labels as the first column and the first row. Other than the labels, the table is filled with relative abundance data. Produced by Y.K.

the input files needs to be transposed before processing by the Titan pipeline. Multiple text input files are handled in the same way that the BIOM text input files are (Refer to the BIOM File Parser section).

6.2 Method Implementation (A.T/Y.K)

Two python scripts were developed for performing the Titan data analysis. Each presents its own advantages, in relation to the desired outputs. Both scripts output a JSON file required for network visualisation, as well as a table in text format containing features of nodes and edges of the network. One of the scripts (Algorithm 2, see below) works on all OTU pairs for each step of the calculation, and stores the intermediate data generated by the pipeline, such as ϕ values, Δ values and p-values. In contrast, Algorithm 1 completes similar sized jobs at a higher speed because it works on all the steps of the calculation for a pair of OTUs and repeats the calculation for the remaining pairs. However, it does not return the intermediate data. The capacity for data retrieval in Algorithm 2 implies that the pipeline has more expensive memory requirements,

Algorithm 1 Certainty Scoring (Y.K)

procedure PHICALCULATION($BIOMcontrol$,$BIOMcase$, $samplingcount$, $alpha$)
    $OTUset = intersection(OTUsfromBIOMcontrol, OTUsfromBIOMcase)$
    $OTUsetTemp = OTUset$
    for each OTU i in $OTUset$ **do**
        $OTUsetTemp$ =remove i from $OTUsetTemp$
        for each OTU j in $OTUsetTemp$ **do**
            initialise lists $Phidistribution$ and $TruePhi$
            while $count$ less than $samplingcount$ **do**
                $bootstrapped = bootstrap(i$ and j from $BIOMcontrol)$
                $subsampled = subsample(i$ and j from $BIOMcase)$
                if Proportionality is chosen for the analysis **then**
                    Calculates proportionality for $bootstrapped$ and $subsampled$
                else
                    Calcualtes correlation for $bootstrapped$ and $subsampled$
                end if
                stores the results to $Phidistribution$ and $TruePhi$ accordingly
                $count = count + 1$
            end while
            Calculation of p-values using $TruePhi$ and $Phidistribution$ with $alpha$
            Calculation of delta using $TruePhi$ and $median(Phidistribution)$
            FDR correction of p-values using $alpha$
            $certainty =$ number of p-adjusted less than $alpha$
        end for
    end for
    return average delta, average p-adjusted, certainty score for each pair of OTUs
end procedure

Algorithm 2 Certainty Scoring (A.T)

procedure CERTAINTYSCORING($ControlData$,$CaseData$,$SamplingCount$,α)
    $OTUset$ = intersection(OTUs from $ControlData$, OTUs from $CaseData$)
    $Pairs$ = combinations($OTUset$)
    Initialise list $NonZero$
    for each Pair (x, y) in $Pairs$ **do**
        $MinZero$ = min(non-zero samples $ControlData$, non-zero samples $CaseData$)
        Stores $MinZero$ to $NonZero$
    end for
    $Dataframe_{Boot}$,$Dataframe_{True}$ = initialise empty dataframes
    for each Pair (x, y) in $Pairs$ **do**
        $x_{Control}, x_{Case}, y_{Control}, y_{Case}$ = remove samples with zero values
        for i in range(0,SamplingCount) **do**
            bootstrapped = bootstrap($i_{Control}$,$j_{Control}$, size=$NonZero$(i,j)-1)
            subsampled = subsample(i_{Case},j_{Case}, size=$NonZero$(i,j)-1)
            if Proportionality is True **then**
                ϕ_{Boot}= proportionality($bootstrapped$)
                ϕ_{True}= proportionality($subsampled$)
            else
                ϕ_{Boot}= correlation($bootstrapped$)
                ϕ_{True}= correlation($subsampled$)
            end if
            Stores ϕ_{Boot} and ϕ_{True} to $Dataframe_{Boot}$ and $Dataframe_{True}$
        end for
    end for
    $Dataframe_{Pvalues}$,$Dataframe_{\Delta}$ = initialise empty dataframes
    for each Pair (x, y) in $Pairs$ **do**
        for Every ϕ_{True} **do**
            Calculates p-value using distribution and median of ϕ_{boot} for (x,y)
            Δ= ϕ_{True}- median(ϕ_{boot} distribution)
            Stores p-value and Δ to $Dataframe_{Pvalues}$ and $Dataframe_{\Delta}$
        end for
    end for
    $Dataframe_{CorrPvalues}$ = FDR correction of p-values
    $DataFrame_{Certainty}$ = initialise empty dataframe
    for each Pair (x, y) in $Pairs$ **do**
        Certainty = proportion of corrected p-values less than α
        Store Certainty, avg(Δ), avg(p-value) to $DataFrame_{Certainty}$
    end for
    return $Dataframe_{Boot/True}, Dataframe_{Pvalues/CorrPvalues}, Dataframe_{\Delta}, Certainty$
end procedure

Figure 20. Shared Package Dependencies	
Package Name	**Application**
os	Generates a zip file of all the outputs from the analysis
sys and *arg*	Package the script as a command line tool for a possible distribution
re	Assigns appropriate names to the output files
json	Returns a JSON file for network visualisation
NetworkX	Generates x and y coordinates using a spring layout
Numpy	Carries out proportionality calculation and other minor mathematical calculations
pandas	Imports filtered BIOM files as data frames and handles large data structures produced by the second algorithm
statsmodels.sandbox	Used for False Discovery Rate (FDR) correction of p-values

Figure 20: **Python Module Dependencies.** These modules were used for both pipelines (described in our previous section). Produced by Y.K/L.A.G.

and thus that the runtime is longer. By having both pipelines available through the web-server, the user can control this trade-off between speed and data retrieval.

Both scripts have similar, shared Python package dependencies: *os, sys, argparse, re, json*, NetworkX (*Schult and Swart, 2008*), pandas (*McKinney, 2010*), Numpy (*Van Der Walt et al., 2011*), and *statsmodels.sandbox* (*Seabold and Perktold, 2010*). These modules and their applications are described in Figure 20.

6.3 Outputs (A.T/Y.K)

The Titan pipeline for both scripts outputs a text file, a JSON file and a log file. These are returned as a zip file that is made available for the user to download from the result page. Input files are also included in the zip archive, as it facilitates identification of what data sets were initially submitted and their corresponding results. Temporary filtered tables can also be

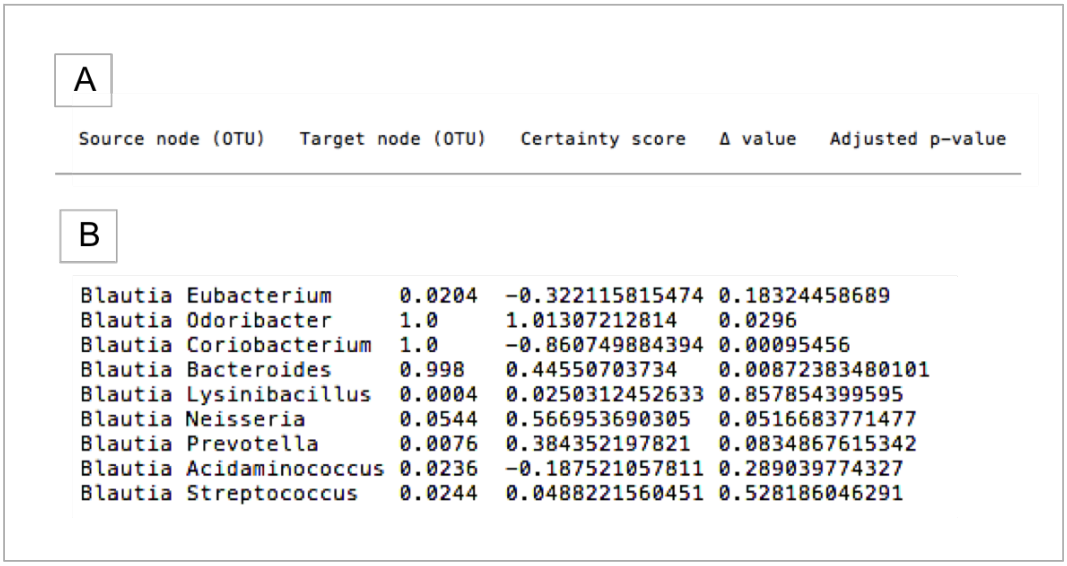


Figure 21: **Main Output Text File Format**. A - Simulated output text file format, displaying edge information: target and source nodes, certainty score, Δ values and Adjusted average p-value. B - Example text format from a test run using the IBD example data. Produced by L.A.G.

produced, if the parser for temporary file retrieval is selected by the user before running the analysis. Merged tables can be outputted if the user provided multiple input files and selected the parser with temporary files. If the user indicated that they wish to retrieve the intermediate data on the input-page, the corresponding files will be included in the zip file

The text file has a tab-delimited format, with one edge per line containing relevant information about a specific pair of OTUs. From left to right, the columns of the file are as shown in Figure 21 (A), and below (B) is a screenshot of an example text file.

For the undirected analysis, Δ values are the average Δ values of the two directed edges. The text file output is designed to be used for further analysis with alternative modelling and data manipulation platforms, such as Cytoscape. Cytoscape allows for powerful statistical analysis and more complex operations, such as clustering or topological analysis, which provide additional information about the network and could also be used for modeling (*Shannon et al., 2003*).

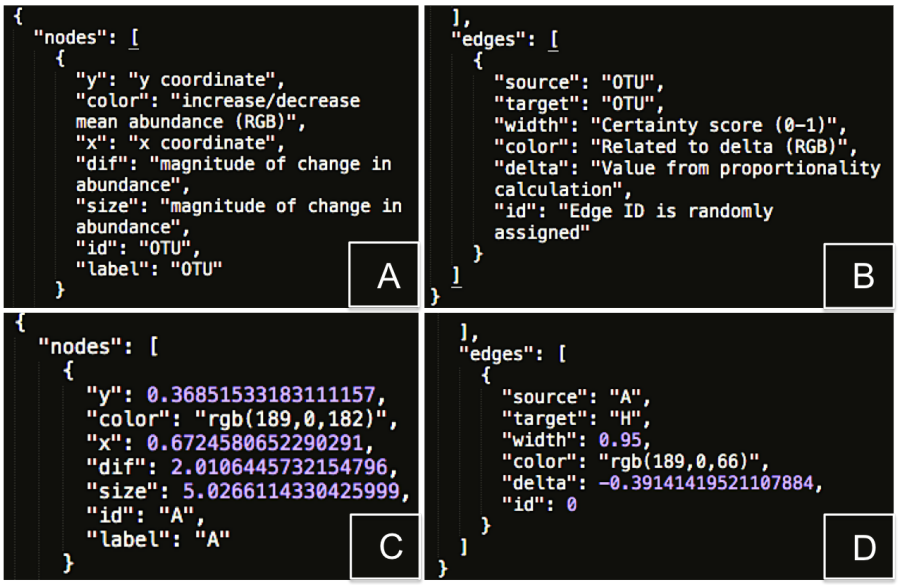


Figure 22: **JSON file format**. A - Simulated JSON format for node (A) and edge (B) data. It displays the information included in the output JSON file after parsing the results through our manually curated JSON parser. It can also be a key for any users providing their own JSON files to render a graph through our "Retrieve Network" function. B - Simulated JSON format for node (C) and edge (D) data - example JSON file. Produced by A.T/L.A.G.

The JSON file output is used for the network visualisation with Sigma on the PLANET website. The JSON file is divided in nodes and edges. Fields for nodes and edges are listed in Figure 22 (A and B), in no particular order, as a simulated JSON file. An example is provided below (Figure 22, C and D). For edge colors, Δ values are used to assign RGB values. There are some cases where Δ values can be infinity or NaN. Infinity Δ values come from infinity true ϕ values, which are the results of the proportionality equation. The proportionality equation uses variance of logarithmic functions. When the denominator variance function yields a zero value, then the ϕ value is infinity. An NaN ϕ value is the result of a logarithmic of zero. As all zeros are removed before the calculation, Δ values calculated by the proportionality method should not have any NaN values. However, correlation can yield NaN values because when all the values in data sets A and B are the same, the two data sets are not correlated, thus the correlation function returns NaN (We used "numpy.corrcoef" for the calculation of correlation).

Email	at3912@imperial.ac.uk
Control_file(s)	['static/uploads/dc1de982-da9d-4e26-aeb6-0bc486c1f28b/genus_healthy_kraken_mpa_table_26Mar15.txt']
Case_file(s)	['static/uploads/dc1de982-da9d-4e26-aeb6-0bc486c1f28b/genus_iCD_kraken_mpa_table_26Mar15.txt']
Taxon	Genus
Numsampling	2500
Alpha	0.05
Directed	False
Temp	False
Inversion	True
Method	Proportionality
Intermediate Data	True

Figure 23: **Log File Output for the Example IBD Data.** Field names are shown in bold in the left column. The parameters that were provided for the run are shown in the right column. Produced by A.T.

For each run of the analysis, a log file is generated with all the information regarding the run. The log file provides a general description of the analysis (Figure 23). It has the email address, input files path, and parameters that the user selected for the run. Each line contains one item in the following order:

- 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

A

		Iterations				
Source (OTU)	Target (OTU)	Value 1	Value 2	Value 3	Value 4	Value 5

B

Source	Target	0	1	2	3	4
Klebsiella	Blautia	-0.1712	-0.2930	-0.2539	-0.1712	-0.1770
Klebsiella	Ruminococcus	-0.4733	-0.4077	-0.5482	-0.4274	-0.5254
Klebsiella	Bacteroides	-1.5504	-1.1307	-2.2257	-1.2901	-1.0900
Klebsiella	Lachnoclostridium	-0.8501	-0.6870	-0.8501	-0.5950	-0.6654
Blautia	Klebsiella	1.7398	1.0795	1.2076	1.7398	1.4219
Blautia	Roseburia	1.4123	1.5072	1.5518	1.5518	1.5072

Figure 24: **Intermediate Outputs Text file Format**. A - Text Format Structure . B - Example of an output: Δ values. Produced by A.T

- 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

If the user indicated that they wish to retrieve the intermediate data, the zip file will additionally include the following files in text format:

- A table with the bootstrap ϕ values generated at each iteration for all OTU pairs
- A table with the true ϕ values generated at each iteration for all OTU pairs
- A table with the Δ values generated at each iteration for all OTU pairs
- A table with the p-values corresponding to each true ϕ value for all OTU pairs

- A table with the p-values corrected for multiple testing using FDR (*Benjamini and Hochberg, 1995*)

The intermediate data files outlined above have a common structure (Figure 24). Each row corresponds to a directed pair of OTUs, as proportionality is an asymmetrical measure and (A,B) might have different ϕ values than (B,A), where A and B are OTUs. The first two columns correspond to the source OTU and the target OTU, whilst the next columns correspond to individual iterations. Therefore, a Titan analysis run with the default parameters will yield intermediate data files with 2502 columns (2500 columns corresponding to all iterations). These files are in a tab-delimited table format, and can therefore be easily processed in Excel or other spreadsheets for further analysis.

6.4 Flask Implementation (Y.K)

Three servers are used for PLANET: Flask, Celery, and Redis (*Grinberg, 2014*); (*Rocco and Helmers, 2016*); (*Sanfilippo, 2016*). Flask is a framework, developed in Python, with various built-in functions, such as a server developer and debugger. Celery is a task queue, which runs jobs (functions) within a server instance. Thus, the jobs can be run as background tasks. Redis is a server that stores many different types of data in memory. It is often used as a database and a message broker, which is delivering messages between two interfaces or applications. All the servers are configured in Python to be connected and integrated with the proportionality calculation. The proporportionality calculation is imported as a module and the function is loaded to Celery.

Flask was implemented to create a server for the front-end. Thus, all the HTML pages used for PLANET follow a Jinja2 template format as required by Flask. Flask integrates all the user interface (the front-end), so the CSSs and JavaScripts are implemented correctly on HTMLs in

a Flask environment. Flask connects the user interface with the functionality of the website. It receives input files and parameters from the input page and assigns random universally unique identifiers (UUID) for each analysis when the user clicks "Generate Network" Button in the input page. The UUID is used to create a directory within a pre-defined uploading directory. This directory is used to store all the outputs of the analysis. Then, input files are uploaded to the analysis directory. Flask also generates a log file in the analysis directory. Then, Flask sends a job to the Celery server with parameters and paths for the uploaded files. When Celery receives a job from Flask, it assigns its worker to the job for ϕ calculation.

Celery uses the Redis server to store the status of the result and also as a message broker for sending and receiving messages between Flask and Celery. Currently, 50 analyses can be run simultaneously. Celery also handles emailing, depending on the job status. Flask module for emailing is used, and Celery functions are used to attach emailing functionality to its workers. When the email address is provided, it sends an email to the user with the URL for the network visualisation once the requested job is completed successfully. For the cases of errors raised during ϕ calculation (errors at different stages of the pipeline - parsers, edge generation, and more. See next section), Celery sends an email to the provided email as an error notification. The error notification has a URL that includes a description of the error raised during the calculation (Refer to the next Section 6.5 for details).

In addition, Flask connects the front-end with Celery, so loading pages and result pages have an access to the Celery job status. Thus, upon the completion of the analysis, Flask redirects the user from the loading page to the result page while maintaining the same URL. Also, the result page can flag error messages with a description of the error raised if the analysis was not completed successfully (See next section for more detail). Upon a request for downloading files, Flask provides the files as a response. Concerning the contact page, Flask handles the request by sending an email to the PLANET server email with all the information provided. For network

retrieval, Flask receives the input JSON file and the parameter for network visualisation, and it redirects the user to the result page for network retrieval.

6.5 Error Handling (Y.K)

During the Titan analysis, errors may be raised. Currently, we are aware of two cases. One type occurs when the input files are either in the wrong format or missing. The second type of error is raised when the network has no edges. As the Titan analysis is run as a background task within the Celery server, we assigned a universally unique identifier (UUID) as a job identifier (ID) for each run of analysis. The UUID is used to keep a track of the analysis. Celery has built-in functions which allow accessing the status of a task using its ID. In Celery, the status of a task is either running or completed. When the job is completed, it just means that the task is no longer running. The error state of the analysis can be accessed after the job is completed. Celery can send a message regarding errors occurred during the pipeline to Flask, which allows the result page to detect an error.

The result page can differentiate the type of errors because the two error types are raised from different parts of the analysis. The first error is raised during parsing the input files. The second error is raised during retrieving edge attributes from the differential network generated by the calculation. Thus, the result page can alert an appropriate message for an error if occurred during the pipeline (see Section 3.2).

7 Further Considerations

In this manual, we introduce the first version of the PLANET web server. Although the core of the software already exists, future work could be directed both towards the improvement of existing functions, and the implementation of additional features.

7.1 TSV parser (Y.K)

The European Bioinformatics Institute (EBI) Metagenomics database uses *.tsv* as one of their taxonomic abundance data formats. This format is very different from *.txt* or *biom*. Thus, a TSV parser can be developed so that users can use TSV abundance files for our analysis. The TSV format has sample id, metadata, and abundance values per line. Thus, we can have a python script that reads the input file line by line to parse the information and generate a dataframe.

7.2 Code Efficiency (Y.K)

A special focus is put on the efficiency of the code. In the future, the code should be parallelised and improved for an increase in efficiency, making it possible for the server to process more and larger files at a faster speed than it currently does. One approach that we are considering is using graphical processing unit (GPU) for the analysis pipeline calculation. The Theano package could be applied to implement these changes (*Bergstra et al., 2010*). Theano creates a graph instance of the function that needs to be computed. This graph instantiation allows the code to be run on GPU. Although it is complex to implement, it has been known that calculations can be much faster with GPU (*Bastien et al., 2016*).

7.3 Additional Network Retrieval Functions (L.A.G.)

Currently, the user is able to extract important information by visually exploring the output, interacting with and manipulating it. We are exploring the possibility of providing additional information to the user when clicking or searching for a specific node. Upon node selection, a temporary, interactive panel will be made visible and display information associated to an OTU in question, such as average p-value reported, certainty score and Δ readings.

Alternative network visualisation environments provide the option to compute network sum-

mary statistics, and explore more complex aspects of a network such as clustering coefficients, centrality measures, analysis of community structures and graph topology (*Shannon et al., 2003*) (*Bastian et al., 2009*) (Refer to Laura de Arroyo's individual report for additional information). Currently, our software functions as a primary exploratory device, and the user is encouraged to use the files we provide to carry out their own downstream analysis. However, in the future, providing a means for assessing summary statistics would be an interesting space that PLANET could tap into. Furthermore, alternative network manipulation environments emphasise the idea of hierarchical network structures (*Shannon et al., 2003*) (*Bastian et al., 2009*). Based on this concept, we elaborated the possibility to provide a so called "Category filter", where the user would be provided the ability to filter nodes in relation to their taxonomic level (species, genus, family, order, class, phylum, kingdom) (*Shannon et al., 2003*) (*Jacomy and Plique, 2015*). By having this feature available to the user, it would prevent having to run separate analyses to explore hierarchical taxonomic relationships.

7.4 Additional Download Options (L.A.G.)

Currently, the user is free to download a zip file containing all files printed during analysis, but at present, no image is being provided to them via compressed files, neither is it shared via their email address. This means, the only way to visualise the graph output generated by our pipeline is by going on the web browser. Our next step would be to implement the possibility to automatically download of an output image into the user-specific zip folder. Additional functions would also include "exporting to the Cloud","exporting to Dropbox", or a "Share your Results" box, enabling the user to send results to a third party (*Poirot et al., 2003*).

7.5 One File Analysis (A.T)

In the future, the PLANET toolkit will be complemented by an additional analysis pipeline, which displays the true proportionality measures between pairs of OTUs in a single dataset. Whilst a simplified version of this tool has already been developed, the sparse nature of metagenomic data raised complexities such as an important number of false positives, identified by a disproportionate proportion of edges with a near zero proportionality value. Metagenomic datasets are very zero-rich, and the process of culling samples with abundance values of zero might leave few non-zero samples for the analysis. The remaining samples are also likely to have abundance values of one, thus leading to the calculation of a ϕ value of zero. A possible solution to limit flawed edge values would be to generate a null distribution of ϕ values for every OTU pair and assess the significance of the true ϕ value by performing a statistical test producing a p-value. For further details about this pipeline and the progress achieved so far, please refer to Isabella Tanase's individual report.

8 Concluding Remarks

In this user manual, we introduce the PLANET web-server. This software's user interface and back-end execution cluster have been designed and implemented using Python and web-browser frameworks, based on the methodology developed by Fairclough *et al.*.

PLANET's main purpose is to provide a computation of both mean abundance and proportionality changes between pairs of OTUs, and to display results visually through a freely available web interface. Whilst this manual mainly discusses the application of PLANET for studies of the human microbiome, the Titan analysis tool can also be employed to elucidate changes in symbiotic dependencies in a range of environmental samples corresponding to altered conditions, such as different locations or growth media amongst others. A crucial purpose

of PLANET is therefore making the methodology developed by Fairclough *et al.* accessible to the wider Metagenomics community.

References

Almudevar et al., 2006. Almudevar, A., Klebanov, L. B., Qiu, X., Salzman, P., and Yakovlev, A. Y. (2006). Utility of correlation measures in analysis of gene expression. *NeuroRx*, 3(3):384–395.

Bastian et al., 2009. Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *Proceedings of the Third Internation ICWSM Conference*.

Bastien et al., 2016. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2016). Using the gpu - theano. http://deeplearning.net/software/theano/tutorial/using_gpu.html. Accessed: 20-03-2016.

Benjamini and Hochberg, 1995. Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

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.

Bikel et al., 2015. Bikel, S., Valdez-Lara, A., Cornejo-Granados, F., Rico, K., Canizales-Quinteros, S., Soberón, X., Del Pozo-Yauner, L., and Ochoa-Leyva, A. (2015). Combining metagenomics, metatranscriptomics and viromics to explore novel microbial interactions: towards a systems-level understanding of human microbiome. *Computational and structural biotechnology journal*, 13:390–401.

Bostock, 2015. Bostock, T. (2015). D3.js: Data driven documents. https://d3js.org/, Accessed: 18-03-2016.

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

Brooks et al., 2015. Brooks, J. P., Edwards, D. J., Harwich, M. D., Rivera, M. C., Fettweis, J. M., Serrano, M. G., Reris, R. A., Sheth, N. U., Huang, B., Girerd, P., et al. (2015). The truth about metagenomics: quantifying and counteracting bias in 16s rrna studies. *BMC microbiology*, 15(1):66.

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.

Consortium et al., 2012. Consortium, H. M. P. et al. (2012). Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207–214.

Fairclough et al., 2016. Fairclough, V., Sim, A., and Pinney, J. W. (2016). Using proportionality measures to analyse changes in inter-genus dependencies within the human microbiota. *Unpublished*, held at Imperial College London.

Flathemes, 2016. Flathemes (2016). Bootflat. http://bootflat.github.io/index.html. Accessed: 20-03-2016.

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.

Jacomy and Plique, 2015. Jacomy, A. and Plique, G. (2015). Sigma.js. http://sigmajs.org/, Accessed: 18-03-2016.

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.

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.

Methvin et al., 2016. Methvin, D., Borchers, K., Sontag, A., Anne-Gaelle, C., Sherov, M., Sexton, A., Smith, A., Schulhof, G., Schiemann, D., Peachey, M., and Resig, J. (2016). jquery: The write less, do more, javascript library. http://https://jquery.com/, Accessed: 18-03-2016.

Morgan et al., 2012. Morgan, X. C., Tickle, T. L., Sokol, H., Gevers, D., Devaney, K. L., Ward, D. V., Reyes, J. A., Shah, S. A., LeLeiko, N., Snapper, S. B., et al. (2012). Dysfunction of the intestinal microbiome in inflammatory bowel disease and treatment. *Genome Biol*, 13(9):R79.

Poirot et al., 2003. Poirot, O., O'Toole, E., and Notredame, C. (2003). Tcoffee@ igs: a web server for computing, evaluating and combining multiple sequence alignments. *Nucleic acids research*, 31(13):3503–3506.

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, 2016. Rocco, M. and Helmers, J. H. (2016). Celery: Distributed task queue. http://www.celeryproject.org/. Accessed: 20-03-2016.

Sanfilippo, 2016. Sanfilippo, S. (2016). redis. http://redis.io/. Accessed: 20-03-2016.

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.

Sekirov et al., 2010. Sekirov, I., Russell, S. L., Antunes, L. C. M., and Finlay, B. B. (2010). Gut microbiota in health and disease. *Physiological Reviews*, 90(3):859–904.

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.

Sokol et al., 2016. Sokol, H., Leducq, V., Aschard, H., Pham, H.-P., Jegou, S., Landman, C., Cohen, D., Liguori, G., Bourrier, A., Nion-Larmurier, I., et al. (2016). Fungal microbiota dysbiosis in ibd. *Gut*, pages gutjnl–2015.

Turnbaugh et al., 2007. Turnbaugh, P. J., Ley, R. E., Hamady, M., Fraser-Liggett, C., Knight, R., and Gordon, J. I. (2007). The human microbiome project: exploring the microbial part of ourselves in a changing world. *Nature*, 449(7164):804.

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.

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.