Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politécnica de Valencia

Including metagenomic analysis in Blast2GO

End-of-degree Project

Grado en Ingeniería Informática

Author: José Antonio Pérez García

Tutor: Juan Miguel García Gómez

External tutor: David Seide

July 2018

ABSTRACT

Bioinformatics is a discipline that is increasingly prevalent nowadays, its objective is the management and treatment, at a computational and statistical level, of biological or medical data. Within this area is metagenomics, a new field of study dedicated to the global analysis of genomic sequences of different microorganisms present in the same community or environment. Although it is a new field, it has many applications, so it is attracting the attention of more and more researchers and, consequently, more and more tools are being developed that are capable of carrying out this type of analysis. Faced with this situation, the company Biobam Bioinformatics is interested in introducing this service in Blast2GO and thus providing this bioinformatics platform with a taxonomic classification analysis. One of the most important features in this type of pipelines is the representation of the results obtained after the analysis. For this reason, a study of all the tools that allow an adequate representation of this type of results has been carried out, as they are already done in other tools that have the same objective such as MEGAN or MG-RAST. The study concluded that the Krona metagenomic data visualization tool is the best option. Therefore, for the integration of this tool in Blast2GO, a new plug-in has been created for the platform. It consists of an algorithm that collects the metagenomic results of the classification and transforms them into a taxonomic tree following the format Krona needs to represent it. As a result, Blast2GO provides its users a graphical visualization of metagenomic data, allowing them to exploit the functionalities of the Krona tool.

Key Words : Graphic visualization, metagenomics, bioinformatics, Blast2GO.

RESUM

La bioinformàtica és una disciplina cada vegada més present en l'actualitat que el seu objectiu és la gestió i el tractament, a un nivell computacional i estadístic, de dades biològiques o mèdiques. Dins d'aquesta área es troba la metagenòmica, un nou camp d'estudi dedicat a l'anàlisi global de seqüències genòmiques de diferents microorganismes presents en una mateixa comunitat o ambient. Malgrat ser un nou camp té bastants aplicacions, per aquest motiu cada vegada està captant l'atenció d'un major nombre d'investigadors i, en conseqüència, s'estan desenvolupant cada vegada més eines capaces de realitzar aquesta anàlisi. Davant aquesta situació, l'empresa Biobam Bioinformatics està interessada a introduir aquest servei en Blast2GO i així dotar, a aquesta plataforma bioinformàtica, d'una anàlisi de classificació taxonòmica. Una de les característiques més importants en aquest tipus de pipelines és la representació dels resultats obtinguts després de l'anàlisi. Per això s'ha realitzat un estudi de totes les eines que permeten una representació adequada d'aquest tipus de resultats, com ja es realitzen en altres eines que tenen el mateix objectiu com MEGAN o MG-RAST. Arran d'aquest estudi es va arribar a la conclusió que l'eina de visualització de dades metagenòmiques Krona era la millor opció. Per tant, per a la integració d'aquesta eina en Blast2GO s'ha realitzat un nou plug-in en la plataforma. Aquest consta d'un algorisme en el qual es recullen els resultats metagenòmiques de la classificació i els transforma en un arbre taxonòmic seguint el format que necessita Krona per poder representar-ho. Com a resultat, Blast2GO proporciona als seus usuaris una visualització gràfica de dades metagenòmiques, permetent l'explotació de les funcionalitats de l'eina Krona.

Paraules clau : visualització gràfica, metagenòmica, bioinformàtica, Blast2GO.

RESUMEN

La bioinformática es una disciplina cada vez más presente en la actualidad cuyo objetivo es la gestión y el tratamiento, a un nivel computacional y estadístico, de datos biológicos o médicos. Dentro de esta área se encuentra la metagenómica, un nuevo campo de estudio dedicado al análisis global de secuencias genómicas de diferentes microorganismos presentes en una misma comunidad o ambiente. A pesar de ser un nuevo campo tiene bastantes aplicaciones, de ahí que cada vez este captando la atención de un mayor número de investigadores y, en consecuencia, se están desarrollando cada vez más herramientas capaces de realizar dicho análisis. Ante esta situación, la empresa Biobam Bioinformatics está interesada en introducir ese servicio en Blast2GO y así dotar, a esta plataforma bioinformática, de un análisis de clasificación taxonómica. Una de las características más importantes en este tipo de pipelines es la representación de los resultados obtenidos tras el análisis. Por eso se ha realizado un estudio de todas las herramientas que permiten una representación adecuada de este tipo de resultados, como ya se realizan en otras herramientas que tienen el mismo objetivo como MEGAN o MG-RAST. A raíz de ese estudio se llegó a la conclusión de que la herramienta de visualización de datos metagenómicos Krona era la mejor opción. Por lo tanto, para la integración de dicha herramienta en de Blast2GO se ha realizado un nuevo plug-in en la plataforma. Este consta de un algoritmo en el que se recogen los resultados metagenómicos de la clasificación y los transforma en un árbol taxonómico siguiendo el formato que necesita Krona para poder representarlo. Como resultado, Blast2GO proporciona a sus usuarios una visualización gráfica de datos metagenómicos, permitiendo explotar las funcionalidades de la herramienta Krona.

Palabras clave : Visualización gráfica, metagenómica, bioinformática, Blast2GO.

CONTENTS

1. INTRODUCTION

1.1 Motivation

This project is included in the field of bioinformatics, which is a science that arises from the need to analyze a large amount of biological data, which also have a certain degree of complexity. For this purpose, software solutions are applied, with a statistical mathematical basis, to maximize the extraction of information. It can be said that bioinformatics is multidisciplinary, since it unites computational and biological science to solve a wide variety of biological problems.

One of these software solutions is Blast2GO, a bioinformatics platform developed by Biobam, which, with an easy-to-use interface for users, presents a series of tools that focus on functional annotation and analysis of genomic data sets with the aim of obtaining as much relevant information as possible, allowing high-performance services to be used in a personalized manner to develop bioinformatics tasks.

Figure 1. *Logo of the company Biobam and of its software platform Blast2GO.*

Biobam wants to incorporate a new type of bioinformatics analysis into the Blast2GO software, more specifically, one of metagenomics. This new field of study within the discipline of bioinformatics is becoming increasingly important. It allows the extraction, sequencing and exploitation of information form the metagenome of a community of microorganisms, in a global manner, that are found in their natural environment, avoiding the individual cultivation of each one, thus facilitating the study that is going to be carried out of a specific environment. In addition, the number of projects in this field has increased, which makes it necessary to develop more bioinformatics solutions for the analysis of these data, that is why Biobam wants to include a metagenomic tool in its platform.

For example, a pharmacist is developing a gastrointestinal drug and, in order to see how their new drug is affecting the intestinal flora and/or how it is being absorbed by the body, a software tool that can provide them with a metagenomic pipeline where they can obtain the answers to these questions in a simple and easy way. Therefore, they takes three samples of the intestinal mucosa of the test subject, one before, one two hours after taking the medication and another after eight hours when it should have been completely absorbed. Analyze these samples using the software tool and see the results of the taxonomic classification of the microorganisms in a good visualization tool in which you can see the differences between the samples with the naked eye, they obtains essential results for the progress of the drug development.

It can be seen the need to implement a metagenomic tool with good data visualization within Blast2GO. This field of bioinformatics is experiencing a boom in the number of studies being carried out on this subject, making it interesting to integrate a software solution of this type. Therefore, what will be done throughout this project is the implementation of a software solution for the taxonomic treatment of metagenomic data to then be able to represent them in a graphical and easy way to understand, by those users of Blast2GO who want to perform a metagenomic analysis. The graphical representation of the results is one of the most important parts of the metagenomic pipeline, since a correct visualization of the data and the results helps the researcher who is dealing with them to analyze them in a fast and easy way, and thus makes them able to reach much more relevant conclusions and with a more powerful and credible demonstration.

1.2 State of the art

In the last decade, metagenomics has emerged as a new science that belongs to the genomic sciences. As mentioned above, it is responsible for the extraction, cloning, sequencing and analysis of the genome of a sample from an entire community of organisms. All this is done for the whole sample at once without differentiating between organism or cultivating them, avoiding the difficulties encountered in the laboratory in the cultivation of certain microorganisms. The central objective of this new field of study is to gain knowledge of the diversity of cultivable and non-cultivable microorganisms in the wide variety of communities and/or ecological environments [1]. Because many different genes can be discover and studied within the community. Also, the need and intense search of biotechnology to identify new

enzymes and biomolecules with industrial application has allowed this field to develop in such a way that, in a very short time, it has obtained a great popularity in research. For example, studies are underway to explore bacterial diversity in marine aquatic environments, such as the sea itself, coastal lagoons and hypersaline environments resulting from the evaporation of seawater. To sequence a large number of microorganisms from media not impacted by humans. In this way, it is possible to study and store the DNA of these microorganisms in DNA libraries where all the genetic information is stored, constituting a genetic reserve of great interest, for example Genbank of NCBI.

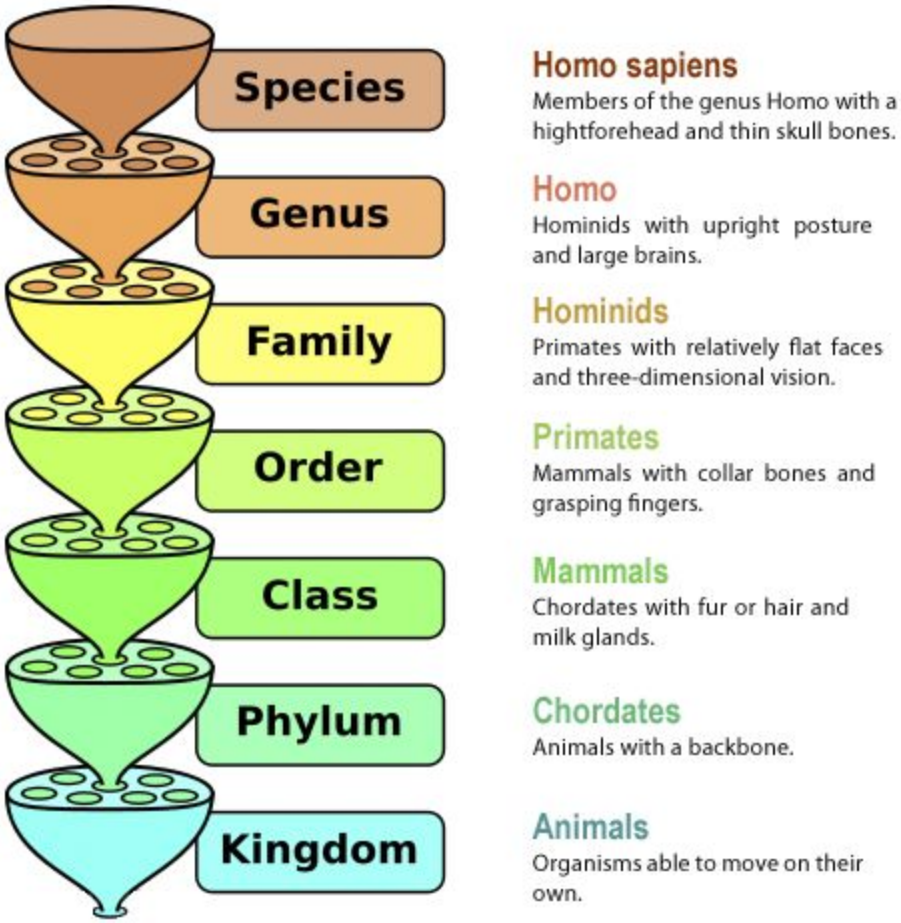


Figure 2. *Partial diagram of the different ranks of the taxonomic lineage of Homo Sapiens.*

For the realization of metagenomic classification, phylogeny is used. Phylogeny is a field within Biology that studies and seeks the origin and development of the various species. It establishes the genetic and evolutionary relationships between species, using their similarities in matters such as DNA, morphology, embryology, DNA molecules, among others [2]. A

phylogenetic tree of life is made to represent these evolutionary relationships between species. In addition, this tree shows the taxonomy of the species. Taxonomy is a classification science that is appliedonce the phylogenetic tree has been made. As can be seen in Figure 2, this structure is hierarchical and has seven important levels or ranks which are Kingdom, Phylum, Class, Order, Family, Genus and Species, the latter being the most specific. Taxonomy uses taxonomic units known as taxon or taxa (in plural), which are one or more related organisms considered as a group.

The breakthrough in high-throughput genomic technologies, as well as the improvement in next-generation sequencing (NGS), has led to a significant rise in metagenomics in science and to great prospects and advances in microbial ecology studies and in access to the genetic content of entire communities of organisms [3, 4]. Currently there are several approaches to sequencing metagenomic samples, the two most popular being the amplicon sequencing of the 16S ribosomal, which takes advantage of the region's distinction between different species to sequence that area, and to make a taxonomic classification of the sample, and the Shotgun sequencing, which sequences the entire genome of the sample completely. There are quite a few differences between them that can determine the use of one approach or another, although that decision is often linked to the study. The 16S rRNA is good for studies where a huge number of samples is analyzed to see the differences between them. 16S is also faster and more profitable. The problem with this approach is that it can only identify bacteria, it cannot identify viruses or fungi. In contrast, random shotgun metagenomics sequencing allows the simultaneous study of all types of organisms. It is even possible to discover new bacterial genes and genomes. It offers a higher resolution, allowing a more specific taxonomic and functional anotation. But it is more costly than the 16S rRNA [5, 6].

For the analysis of a metagenome, which is a set of microbial genes present in a given environment or ecosystem, different approaches can be followed. A simplified example of the procedure is represented in Figure 3. After the extraction of the sample(s) and the sequencing step, a metagenomic classification is performed. For this, there are several technologies that can be used. For example MEGAN[7], which is an open source analysis package that introduces a pipeline with an interactive visualization of the results. In 2006, It became one of the first metagenomic data analysis tools. Firstly, the program takes the reads of the samples and compares them with known sequences from a database, using the BLAST algorithm. Then, it associates a taxa to each read with the LCA (Lowest Common Ancestor) algorithm and shows

the results in a chart. Another well-known technology is Kraken. It is a method that quickly identifies all the readings of a metagenomic sample. These reads are k-mers which are small sections of k-length from the DNA sequence of the sample. These readings are confronted with a database, constructed by multiple genomes, for the taxonomic classification of metagenome based on matches. Kraken also uses the LCA algorithm to find a k-mer that has been paired with two or more taxa [8].

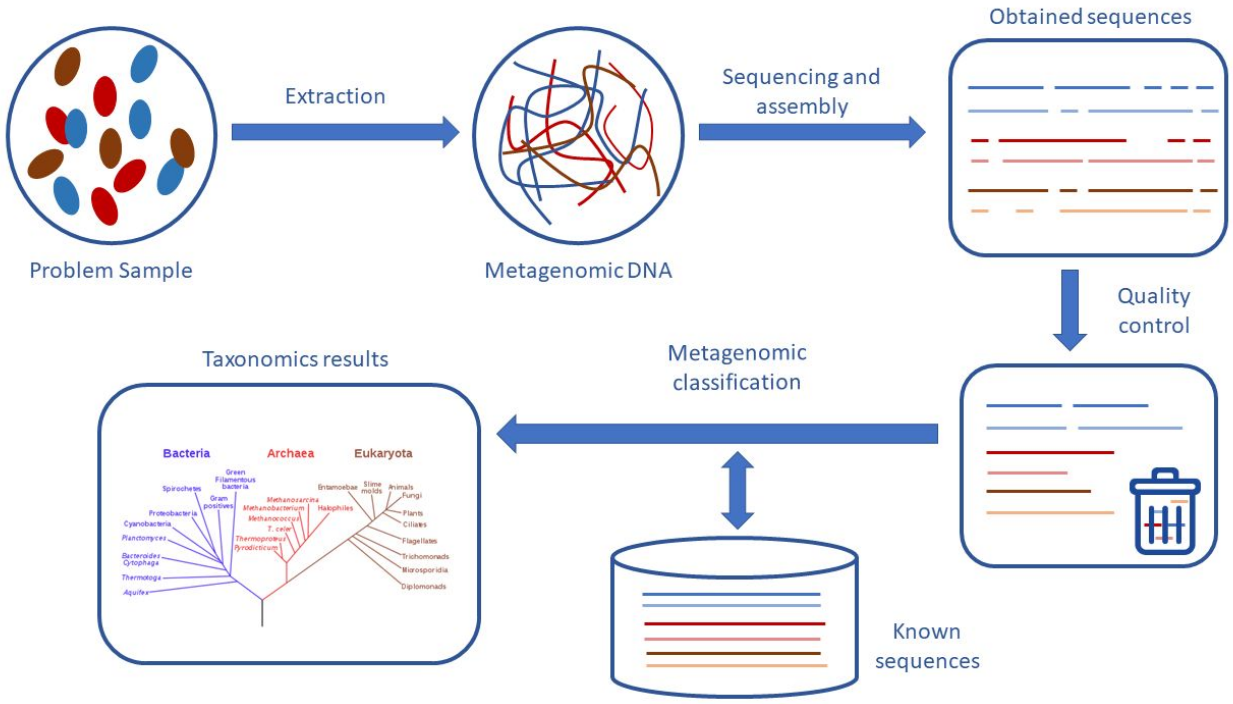


Figure 3. *Simplified example of the procedure of a metagenomic analysis.*

The last step of the metagenomic analysis pipelines is the representation of the results. Nowadays, various metagenomic pipelines which add a graphical visualization in addition to their analysis result exist. This is important because they were taken as a model to follow and understand what a representation of metagenomic data can look like, what kind of information from the results may be shown and how it could be implemented and integrated in Blast2GO.

1.3 Objectives

The main objective of this project is to integrate a good and proper form to graphically represent the results of the taxonomic classification into the Blast2GO platform. This tool will be added as part of the new metagenomic workflow that Biobam is willing to add to its software's service. This objective can be separated in various sub-objectives which guide this project:

1. Understand how a metagenomic analysis can be done.
2. Find a visualization tool which can show as much information as possible in a user-friendly way to obtain meaningful insights to conclude the metagenomic investigations in an easy way.
3. Adapt the solution to the plug-in structure of the Blast2GO platform.
4. Create a compatible data structure to facilitate all the necessary data to draw the chart.
5. Implement an algorithm to adapt the result of the taxonomic classification of the metagenomic pipeline to the new data structure and to finally create the graph.

2. ANALYSIS

The objective of the Taxonomic chart project is to represent a taxonomic classification result of a metagenomic analysis inside the Blast2Go platform. It is necessary to clarify in detail which will be the input of the algorithm and what is the expected output.

2.1. Specifications of the input

At the beginning of the pipeline the reads of the metagenome are uploaded in FASTA or FASTQ format. Both file formats are a standard to represent nucleotide sequences of the DNA using a single-letter code. The FASTQ files also contain a quality score. The metagenome is analyzed and classified taxonomically by Kraken using a cloud service. The final result is a TSV file in which each value of the row is separated by tabs. The values of the file are the name of the read, the taxonomy in which it was classified and an average score that assigns a confidence on the classification of the read. This file is parsed into a Blast2Go object which name is KrakenObject. This object is the input of the Taxonomy chart project. Moreover, the KrakenObject is displayed in an organized table in the Blast2GO platform.

KrakenObject holds the following data:

- **KrakenData**. It is Google Guava table data structure. It is a simple dictionary with which it can be obtained the count of a taxonomy, which is the number of reads associated to a taxonomy, with the ID of this taxonomy and the name of a sample.
- **RankedData**. Same as above but with cumulative counts per rank. This cumulative count is obtained in the following way, if for example a sequence is classified at the species level, its count is added to the parent level, and so on until reaching the level of superkingdom. This data is the one represented in the table which is shown in the Figure 4.
- **KrakenScores**. Another Google Guava table with which can be obtained the relative average confidence score of the reads associated to a rank until the level of species, with the ID of the taxonomy and the name of the sample of the metagenome analyzed.

- **Output**. The raw data parsed from the TSV file which is the result of the Kraken analysis. This attribute is a dictionary which key is the name of a sample of the metagenome analyzed and the value is the information of its classification. The KrakenData and the KrakenScores use this information to fill their tables.
- **HtmlReport**. HTML which appears in a tab of Blast2GO once the analysis is done. It shows statistics of the analysis, such as the percentage of the reads classified, statistics of the taxonomy as the percentage of the classification in each rank, the parameters of the analysis and the distribution of the top ten hits taxonomy per rank.

Actually, the algorithm is fed with the KrakenData and the KrakenScores attributes of the KrakenObject. It uses the first attribute to create the taxonomy tree and to compute the count of each taxon. the second is used to add the average score to each rank of the taxonomy.

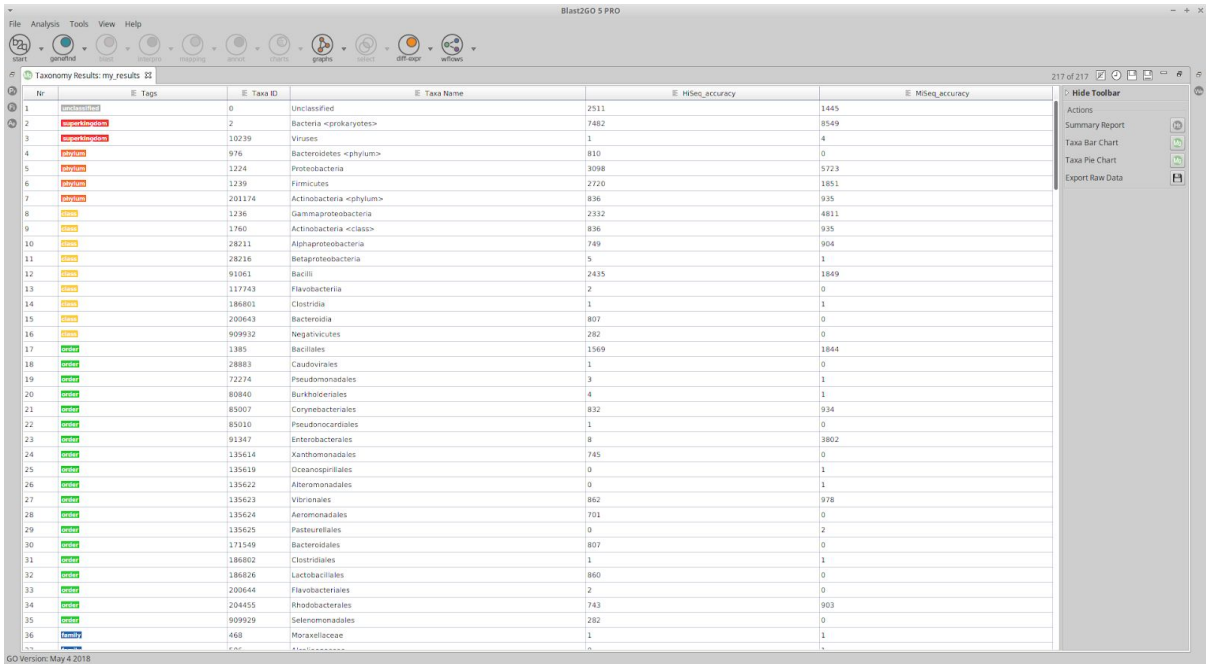


Figure 4. *Present table which represents the data of the KrakenObject.*

2.2. Specifications of the output

The expected output of the algorithm is a graphical method capable of representing the results of a metagenomic analysis. It should be done in a way that is easy to understand and uncomplicated to use for the user. In addition, it should help to achieve meaningful insights and

therefore have the ability to be exported to PDF or PNG format so that this graphical solution can be added to a paper of a scientific nature. The user should also have the possibility to visualize the results of several metagenome samples at the same time in a convenient way to allow a direct inter sample comparison. The chosen visualization tool should display the complete lineage in a single graph. In order to see a more global information of the hierarchy of a sample at a glance. Moreover, if it is going to be added into a report, this characteristic makes that if it can be seen the whole lineage just one chart is going to be attached into the report. On the contrary, it should be attached one chart per rank.

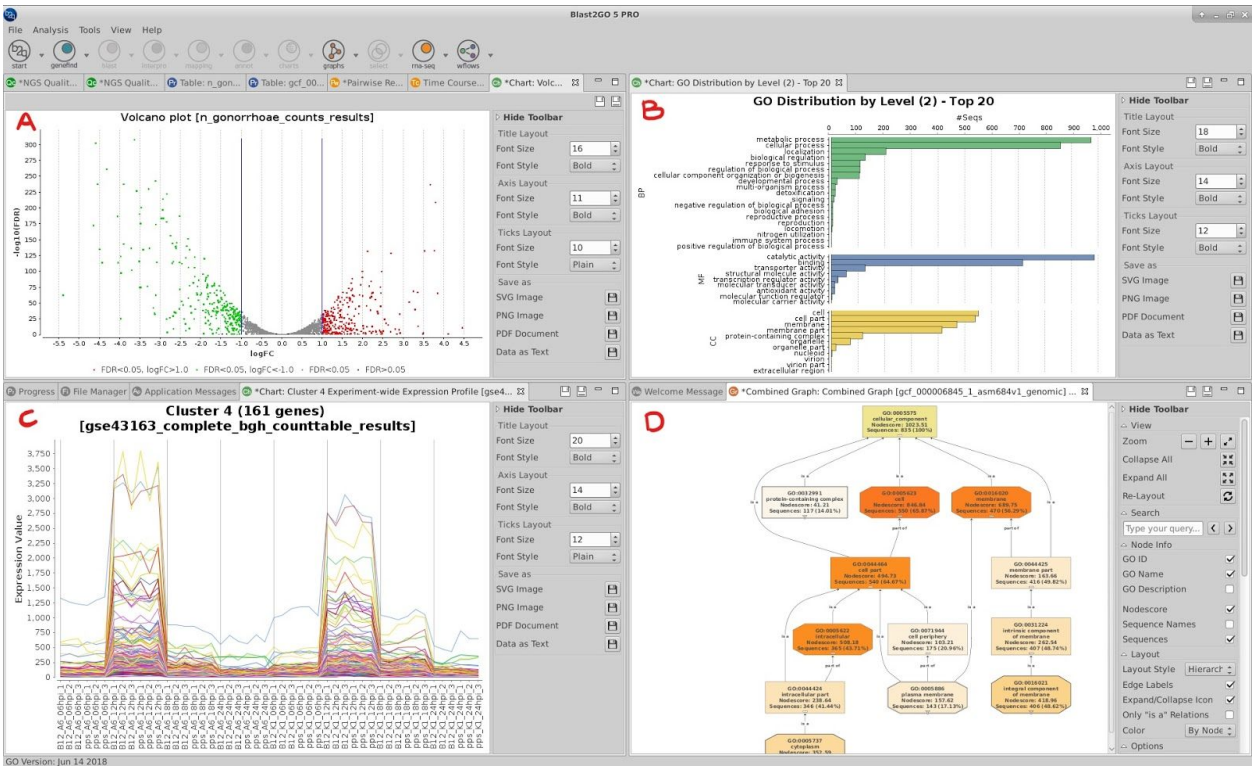


***Figure 5**. Group of some charts offered in the Blast2GO platform. A) Volcano Chart. B) Bar Chart. C) Linear chart. D) Combined Graph.*

In order to strengthen the visual identity of its product, Biobam has created a unique visual guideline for its charts in Linux, MacOS or Windows. Therefore, another thing that the graphic output of the algorithm should take into account is to have an interface equal or similar to the other charts of the platform. That helps to integrate this tool better.

As shown in Figure 5 the chart viewer is made up by two panels and a navigation bar at the top of each tab. The left one is where the chart itself represents graphically the data, it takes

up the biggest space of the viewer. The background color is white to see the chart more clearly. On the right side, there is the panel with all the settings and options to modify and control all the data which is being represented. In contrast to the actual chart, the background of the side panel is light grey. This sidepanel is divided into groups of options that with similar functionality. For example, in Figure 5.C the grey panel has four groups. The groups are delimited by a darker line and all the letters are black. It is convenient to follow the specifications of the interface, because it helps to guide the user and makes the tool more intuitive.

3. METHODOLOGY AND DISCUSSION

The First thing before the development of a software project is to have a clear idea of what technologies and tools should be used for its proper implementation. This chapter will lay out the study and selection of the usage of the technologies and tools which have been used for the implementation.

3.1. Graphic Representation Tool

As explained before, a very important part of a metagenomic pipeline is the visualization of its results. This objective can be achieved by just choosing a proper graphic tool able to represent this information in a clear way to make it easier to obtain a significant understanding of the study. Even though Blast2GO has a large variety of charts and graphs, neither of them could represent metagenomic data properly. Therefore, a study has been made to determine which visualization tool should be used. There are a lot of different options to choose, and each one can give distinct relevant information of the data they represent.

3.1.1. MEGAN

The graphical representation tool chosen by MEGAN is the node-link tree (Figure 6 and Figure 7). This chart has alike a tree shape, and it is made up of nodes and edges. A node can have two different types of shapes,depending on if the tree is for the representation of one or various samples. If the tree only shows one sample, the nodes have a circle shape which diameter is directly proportional to the number of reads (Figure 7). The more reads the node has the bigger the diameter. In the example of Figure 7 it can be seen that there are more reads of Proteobacteria than any other phylum because it has the biggest circle. If the tree represents more than one sample at a time, the node has a rectangle shape divided in squares equal to the number of samples which are being compared. It can be seen in Figure 6 that, as there are twelve datasets, the rectangles are made up by twelve squares, one by sample. Each square has a part of its area colored depending on the number of reads the sample has in this node, one different color for each sample. Independently of its shape, each node represent one variant in a rank of the whole taxonomy of the sample. For example, in the Mcaldum's node (Figure 6) it can be seen that the sample corresponding with the color blue has more reads

associated in this node than the others, because the colored area is bigger. The other part of the chart are the edges, a node can have one or more edges. The edges connects nodes between them building a fatherhood relation. The nodes without child nodes are the leaf nodes. The node of the left part of the edge is the father of the right node. This relation spreads from left of the tree to the right creating levels in the taxonomy, each level is a rank and the deeper the more specific. That is the reason why the root is in the left side in Figure 6 and Figure 7.

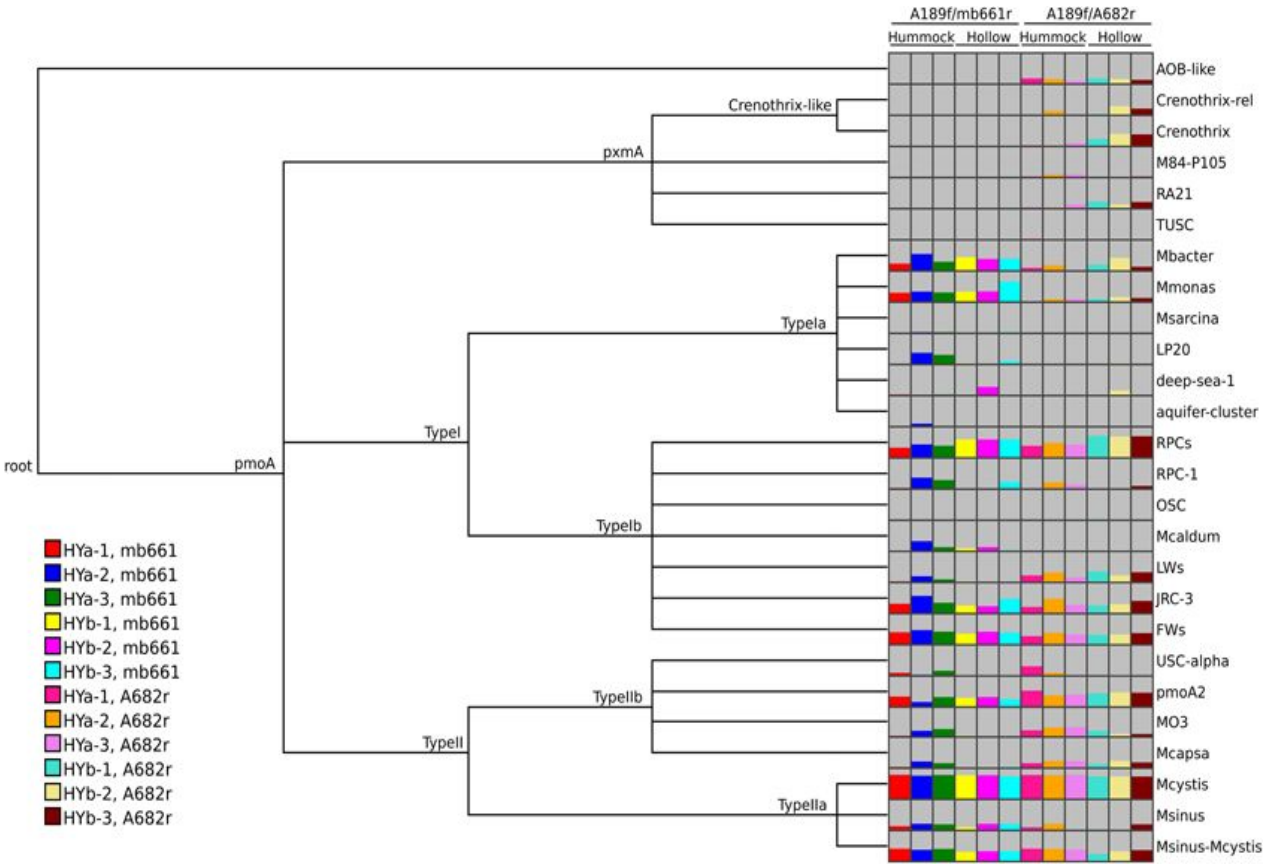


Figure 6. *(Image from Dumont MG, Lüke C, Deng Y and Frenzel P (2014) [14] Figure 5) Example of a representation of a multiple sample node-link tree in MEGAN.*

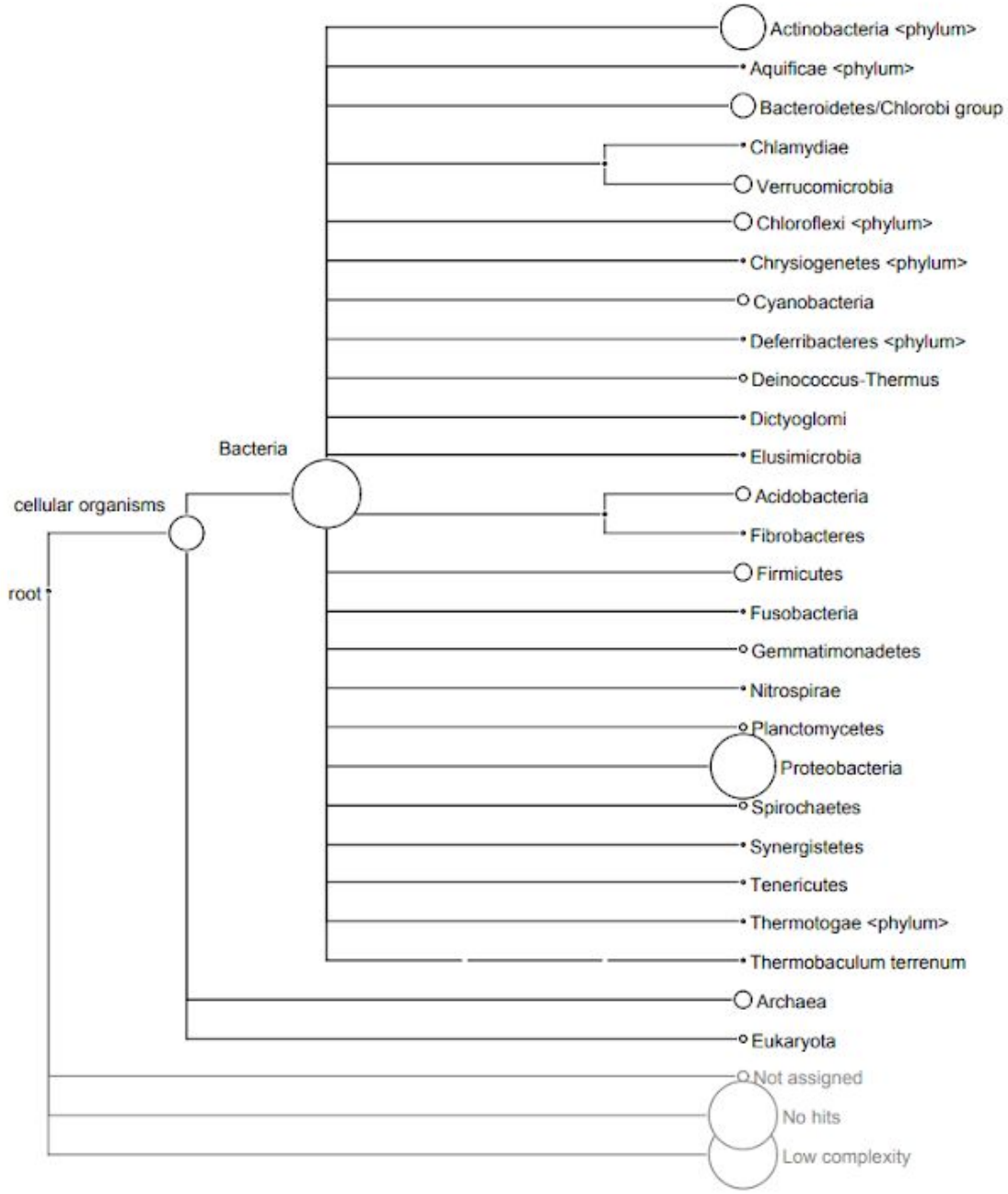


Figure 7. *Example of a representation of a single sample node-link tree in MEGAN with a data set which only reaches to the phylum rank.*

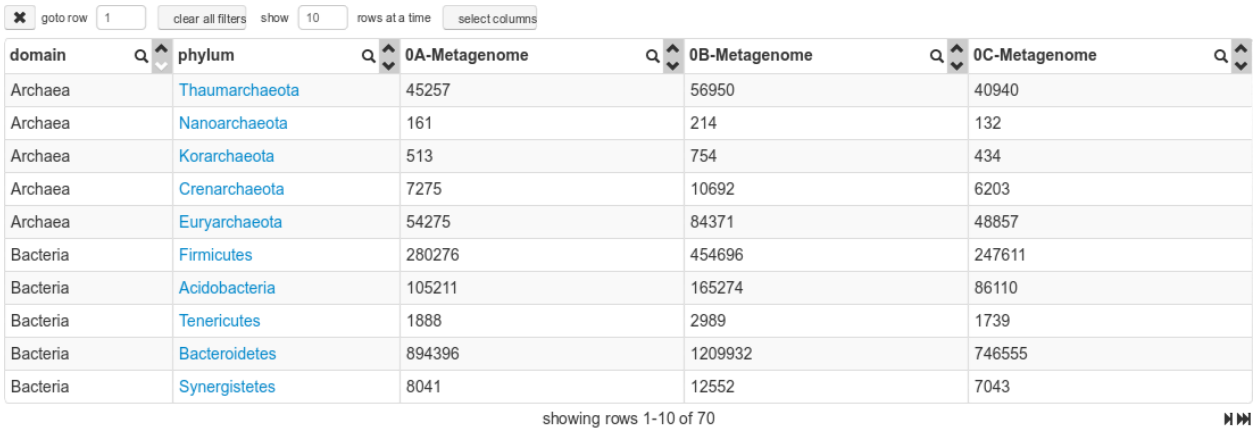
3.1.2. MG-RAST

This web application service has a large group of different chart visualizations to visualize the results of the metagenomic classification, at the end of the analysis [12]. Here

there are the most interesting charts offered in the pipeline to show the result of the taxonomy of an metagenomic analysis:

Table

In the case of the table represented in the Figure 8, the first column name is “domain” and it indicates that in the rows below will appear the name of the domain where the phylum indicated in the next cell belong to. In the same example, In the first row is written “0A-Metagenome” due to, in all this column, the number of counts of this dataset classified in each phylum is printed. The table can provide to the user all the lineage knowledge of the metagenome. It can also show this information but from various analyzed samples at the same time. In order to do that, a column by sample is added to display the information. Figure 8 represents three datasets “0A-Metagenome”, “0B-Metagenome”, “0C-Metagenome”. Each column of the chart has an up and down arrow to sort alphabetically the names or from highest to lowest value, depending on what information is being displayed in each column. Also, it has a magnifying glass to search for a specific information of a row. Usually, there are a lot of different entries in the same rank and the table only shows the first ten rows, the user can modify this number in an option which appear at the top of the table, also, at the bottom there are some arrows to navigate through the table. This type of representation is useful for those users that prefer a table, just a simple way to see the lineage of all the organisms in the samples and how many reads by variety there are.



domain	phylum	0A-Metagenome	0B-Metagenome	0C-Metagenome
Archaea	Thaumarchaeota	45257	56950	40940
Archaea	Nanoarchaeota	161	214	132
Archaea	Korarchaeota	513	754	434
Archaea	Crenarchaeota	7275	10692	6203
Archaea	Euryarchaeota	54275	84371	48857
Bacteria	Firmicutes	280276	454696	247611
Bacteria	Acidobacteria	105211	165274	86110
Bacteria	Tenericutes	1888	2989	1739
Bacteria	Bacteroidetes	894396	1209932	746555
Bacteria	Synergistetes	8041	12552	7043

Figure 8. *Example of the representation of three datasets in a table using MG-RAST.*

Matrix

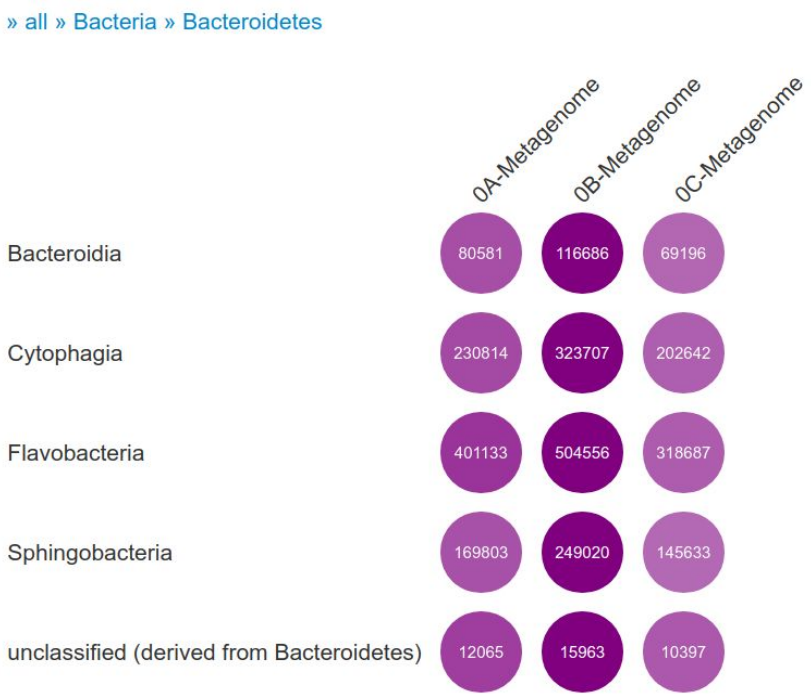


Figure 9. *Representation of three datasets in MG-RAST in a Matrix Chart.*

With this representation of the data the user can only see a level of rank at a time. In Figure 4 there is an example of a matrix which only represents the Bacteroidetes phylum. It is composed by a name that indicates which variety of the rank it is showing and a colored circle with a number inside that shows the amount of reads for each row. Different datasets can be seen at once like in the table. In this case, a colored circle for each sample is added. According to the difference between the number of counts in each circle, they will have a different variation of the color base. In the Figure 9, the circle with more reads is more purple than the others, and the circles with a similar number of counts have a similar color. The base color can be modified in the settings menu of the layout.

Pie chart

A very familiar type of chart which looks like a sliced circle where each slice, from the taxonomic point of view, is a type of variety in the rank and its size is proportional to the number of reads. It can only show one rank at a time. Furthermore, just the information of one sample can be show, considering that there is no option to change between datasets. Figure 10 represents all the varieties of the Bacteria domain of a dataset, it can be seen that Proteobacteria is the phylum with the most reads associated to it.

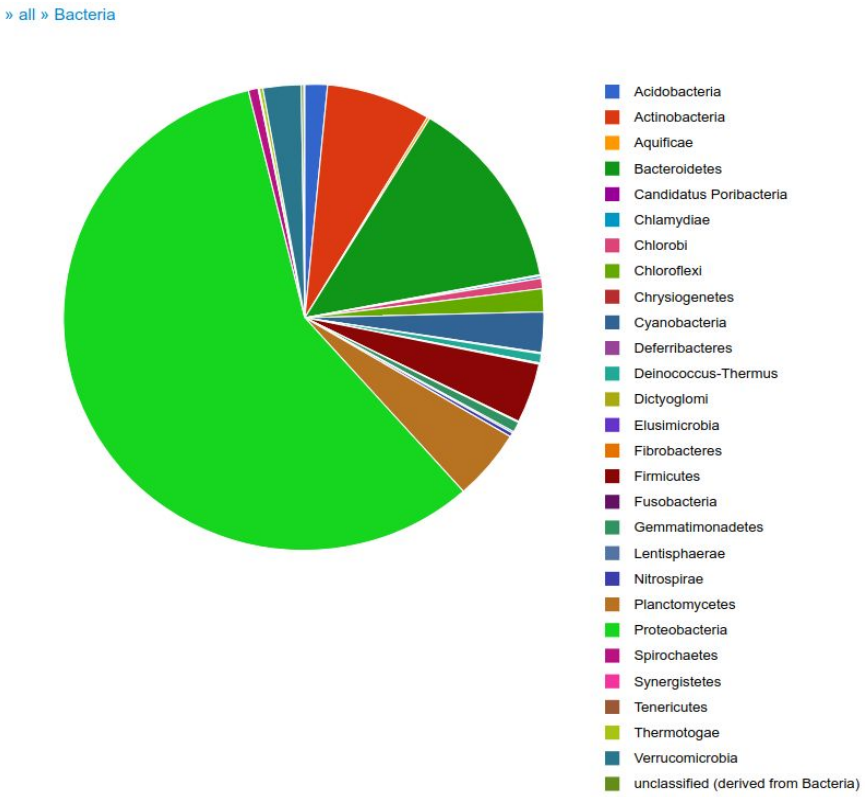


Figure 10. *Example of a Pie Chart in MG-RAST. It is represented three different datasets.*

Donut chart

A chart with an annulus shape avery similar to the pie chart. With this chart the user can also compare the distribution of the reads between various samples because the chart adds an inner ring for each sample. In Figure 11 it is easy to see that the sample with more Actinobacteria is the one of the middle ring, while the other two have approximately the same amount of reads. Nonetheless, only a rank level can be plotted at a time.

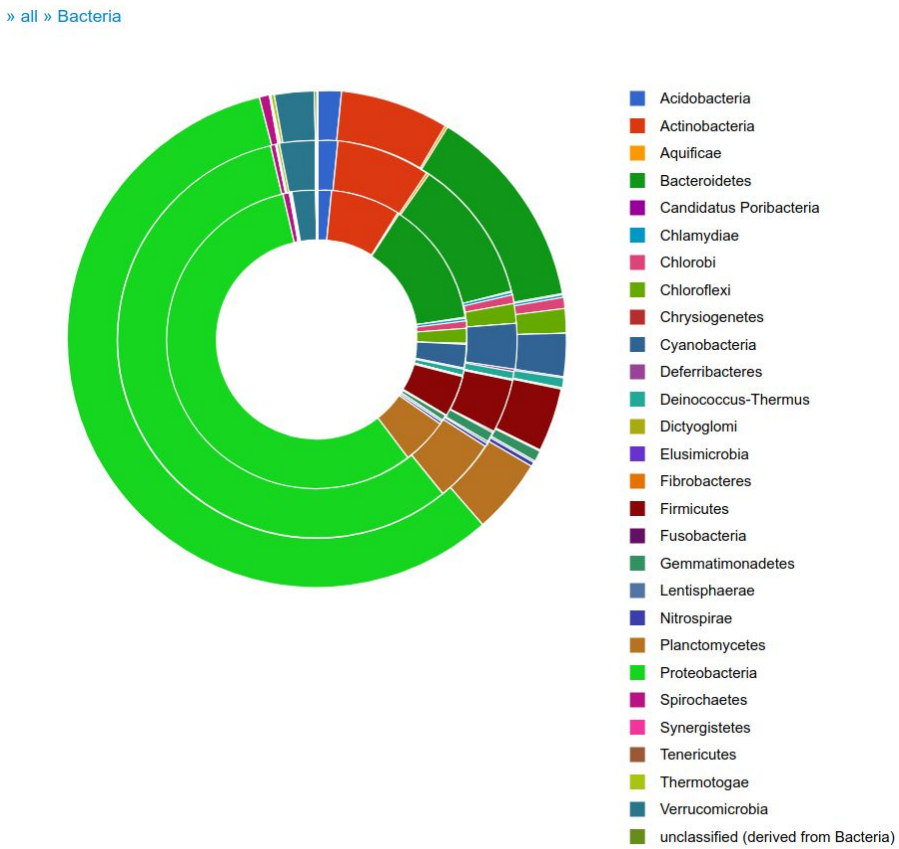


Figure 11. *Example of a Donut Chart in MG-RAST representing three different datasets at a time.*

Bar chart

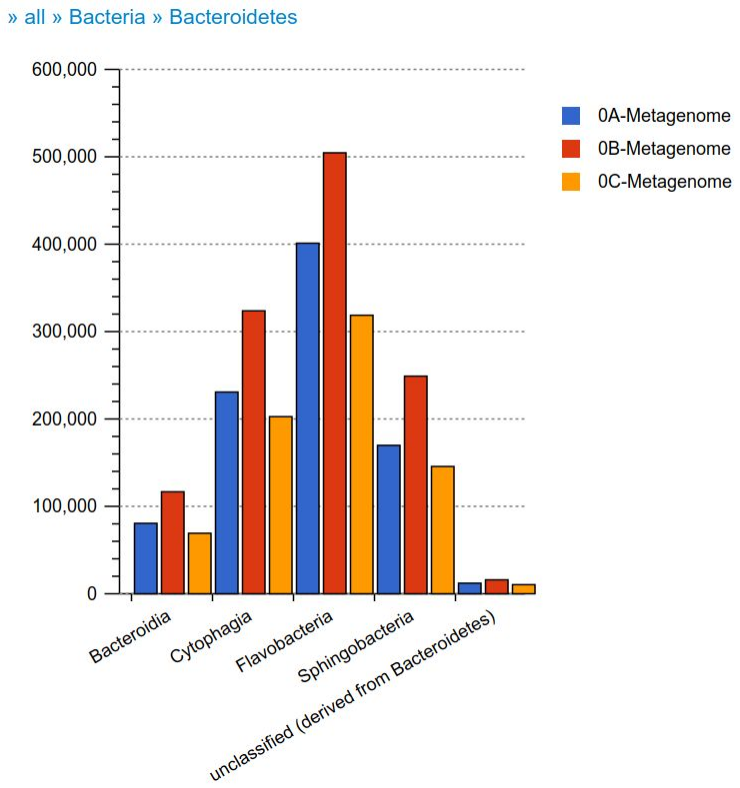


Figure 12. *Bar Chart in MG-RAST representing three different datasets at a time.*

This type of chart is made up by bars (Figure 12). It can only represent a rank level at a time, but multiple samples in the same chart by just adding bars, one by sample on each variety of the rank level. Each bar size depends on the number of the reads that belong to a category in relation to the total amount of reads in the whole sample. The chart provides various tools to perform data treatment such as normalization of the reads and logarithmic scale representation of the data. For the modification of the layout, it has the same capabilities of customization as the previous two charts.

Stacked Bar Chart

It is exactly the same as the previous graph but it condenses all the categories of a rank in just one column (Figure 13). This column is divided by the number of variants in the rank represented. Each portion of the column has a proportional size according to the number of reads associated to the classification. Moreover, each portion of the column has a different color to be able to distinguish between them.

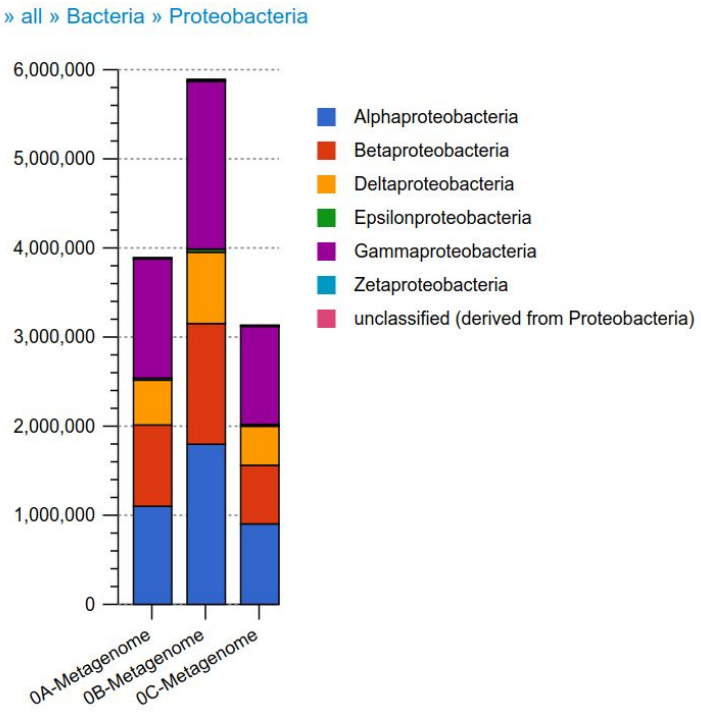


Figure 13. *Example of the representation of various datasets using the satcked bar chart of MG-RAST.*

Heatmap

This type of chart is made of a dendrogram, which is a tree diagram that represents hierarchical clustering, and a sort of matrix of colored squares (Figure 14). In this matrix, the columns are the samples and each row represents a classification inside a rank of the lineage. The square is colored depending on the number of reads associated with it, following a gradient scale calculated with the normalization of the count of each category represented. Only one rank can be displayed with this chart but it can plot the result of various samples.

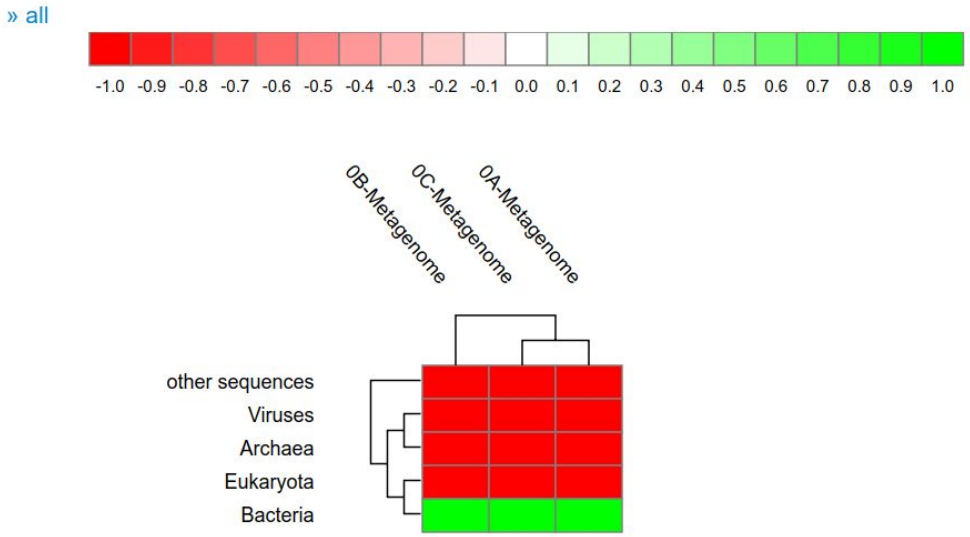


Figure 14. *Example of the usage of a MG-RAST's Heatmap for the representation of a taxonomy classification.*

MG-RAST has more graphic statistics but not all of them give this kind of information like the rarefaction plot, which shows the increase in species for increased numbers of reads, the principle component analysis plot, which shows the distances of the samples in a 2D chart, and the differential abundance plot, which displays the logarithmic abundances of the categories of two samples.

Besides, in the upper side of all the charts, a guide of the lineage is shown if the user navigates through the taxonomy classification of a sample. Moreover, with the exception of the table, the height and the width of the chart can be modified. Also, the user can change more specific characteristics depending on the chart, for example, in the pie chart the diameter can be modified, the color of the circles of the matrix chart and the squares of the heatmap and the width of each rim and their distance from the border to the center of the donut chart. Furthermore, MG-RAST offers an option to normalize the count of reads of the dataset in the table, the stacked bar chart, the matrix and the bar chart, and in the last two a logarithmic scale representation is possible.

All the tools to modify the charts itself and the data plotted in all the charts presented can be filtered by e-value or length of the read, among other filters. And the user can not only modify the chart to view it on the browser. MG-RAST also offers the option to export the chart in SVG or PNG maintaining all the changes the use has made. Data can be exported in TSV detailed

format. It is a format that stores the simple text data in a tabular structure. Moreover, the dataset can be stored in FASTA format and biom format, which is a way to represent biological samples by observation contingency tables.

3.1.3. D3.js library

One option was trying to copy the style of any of the previous charts for the representation of a taxonomic classification by programming them, using the library D3.js which has a BSD license. D3.js is a JavaScript framework used to create dynamic and interactive data visualization in web browsers, following the HTML, SVG and Canvas standards [10]. The dynamic property of the library comes from the capability of D3.js of binding the data that is to be represented with the DOM, therefore, every time the data changes, the chart will change too.

D3 visualization tool allows to implement any kind of graph according to the necessities and the desires of the user and the situation. To create the graphical part it used SVG, this format is based in XML because it uses formatted tags to create objects with associated properties and behaviors with which the image is formed. Then, more capabilities can be added to the chart and animate it using JavaScript. Moreover, as SVG is a vector image format, all the elements created with it can be scaled without any loss of quality. Due to this, D3 is used very often in publications and web pages. It is easy to use and it is very well documented, its GitHub is very popular and it receives contributions of hundreds of developers.

With D3 it is not necessary to create a graph from scratch, on its official web page there are a lot of more complete examples than the charts explained in the section before. Therefore, it was a better option to search for one that is already developed and then modify it to adapt it to the requirements of the metagenomics classification results. Among the examples, there are bar charts and pie charts like the ones seen in the MG-RAST pipeline, node-link tree charts similar to the one seen in MEGAN, and a type of chart called sunburst chart. As it can be seen in Figure 15, this chart is circle shaped as the pie chart, but it is made up of different levels of rings, one inside another, each one divided in slices that form circular trapezoid nodes, where the ones from the outer ring make different groups together that end up in an inner trapezoid node of an inner ring. This builds up a son-parent relation between contiguous rings, the outer ring is the son of the inner one, the deeper the ring the more general taxonomic information is represented.

Figure 15. *Example of a sunburst chart in D3.js*

3.1.4. MetaTreeMap

An alternative visualization to represent taxonomic results of a metagenomic analysis is MetaTreeMap [15]. This graph solution was born from the problem of the linear representation of trees which have a lot of nodes to represent, the legibility and the difficulty for perceiving the read quantity assigned to each node. This drawing method uses a flat view of nested rectangles representation of a taxonomic classification hierarchy, as it can be seen in Figure 17, following two rules: firstly, internal nodes are containers and secondly, all reads are represented in rectangular areas. MetaTreeMap has a BSD license and all the software needed to use it can be download from its web page.

MetaTreeMap is a good example of using an already developed chart of the D3.js library and adjust it to the specifications of the represented target. This visualization tool uses a treemap chart from D3 and, using JavaScript, increases its functionalities to adapt it with metagenomic data requirements. Finally this tool is embedded in a graphical user interface using Boostrap, which is a open source toolkit for developing interfaces that can be adapted to any size of the screen with HTML, CSS, and JS.

The process of drawing this chart begins with a JSON file format with read numbers associated to a taxon inside a hierarchy, as it can be seen in Figure 16. The format of the file has to have the same specific fields as those in MetaBin, which is a program used for the taxonomic binning of short reads using a homology-based approach. Nevertheless, MetaTreeMap has a script to transform custom JSON files to a file that follows the structure mentioned before. With all the taxas of this file, a skeleton tree is created and used to determine the hierarchy, this is the data structure of the MetaTreeMap. Then, the tree leaf nodes are filled with its reads counts. Finally, a treemap is drawn where each element of the hierarchy is a rectangle in the chart and each one has an area size proportional. Moreover, a sub-branch of the tree will be represented as an intermediate rectangle and the area will maintain the rule of the proportion to the number of reads.


```
{
    "name":"root",
    "children":[
        {
            "name":"cellular organisms",
            "children":[ ... ],
            "data":{
                "assigned":"195",
                "sum":"52616",
                "rank":"no rank"
            },
            "id":"131567"
        },
        {
            "name":"Viruses",
            "children":[ ... ],
            "data":{
                "assigned":"16",
                "sum":"16",
                "rank":"superkingdom"
             },
            "id":"10239"
        },

    ...

    ],
    "data":{
        "assigned":"1",
        "sum":"54271",
        "rank":"no rank"
    },
    "id":1
}
```


Figure 16. *Example of an extract of a JSON file used to build a MetaTreeMap chart*

MetaTreeMap has an option to draw various datasets in a single chart. For this, it creates one tree by sample and then it merges it all in one big tree. A common tree is used to

create the structure and a children node by sample with the number of reads associated is added to the node that has the same name as the one which is leaf in the tree of the dataset.

Furthermore, the user can navigate through the hierarchy of the lineage of a sample by clicking in the rectangles or on the treemap header to zoom in or out or just using the search bar. To obtain more information, the user just has to mouse over a rectangle. Other settings that the chart offers is to set a cutoff of representation in the phylogenic rank, modify the color palette of the rectangles and the background or the font size and save the chart and the table in various formats. If there are different dataset displayed in the chart, an option of coloring the rectangles according to its dataset is enabled. MetaTreeMap offers a normalization of the data, showing inside the rectangles the percentage it represent of the total number of reads.

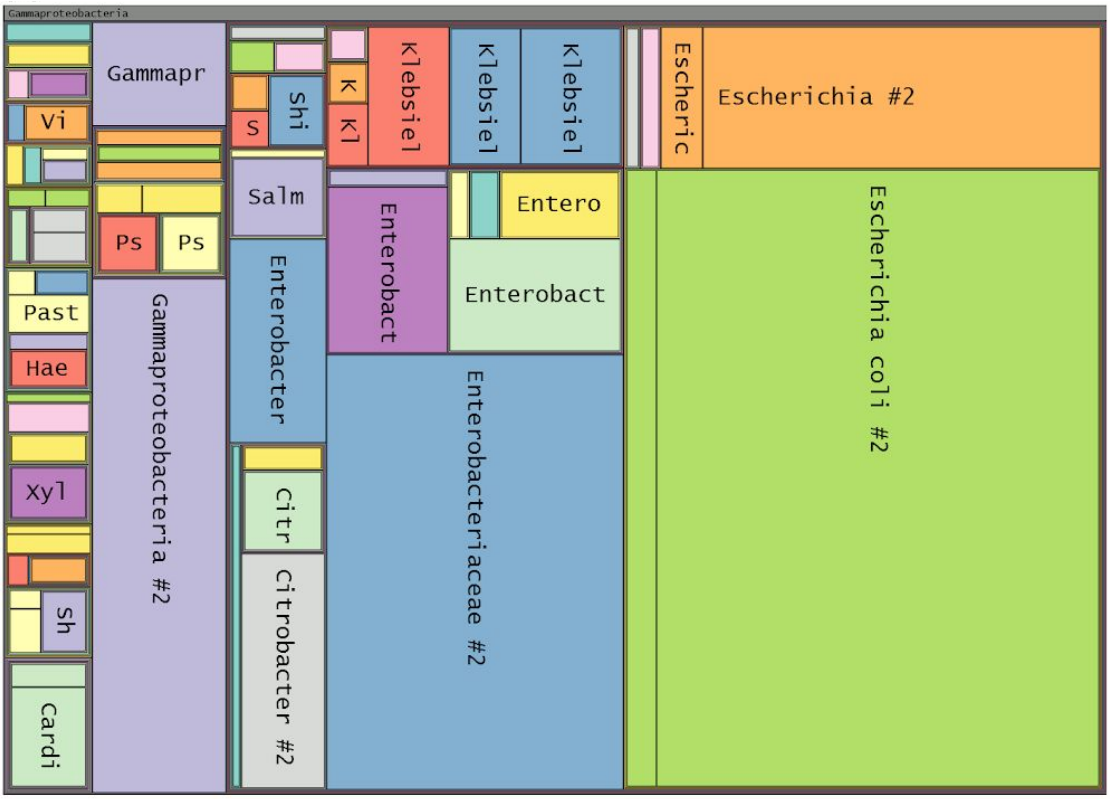


Figure 17. *Treemap view of Gammaproteobacteria*

3.1.5. Krona

Another option was to use Krona visualization to show graphically the result of the shotgun metagenomic analysis. Krona [11] is a flexible, interactive and platform independent

tool with a BSD license. It is developed at the Battelle National Biodefense Institute, with the main goal to visualize complex metagenomic data classification hierarchies in a Web browser.

The creation of the chart begins with the transformation of the data in a taxonomic tree, this can be done in different ways. As it can be seen in Figure 20 there are different ways that Krona can be used as input to build its chart. One option is importing the result of the taxonomic classification from a bioinformatics tool. Another option is to use the classifications based on NCBI Taxonomy which are the results of some of these bioinformatics tools. Krona provides an excel file to detail manually the lineage and magnitude of the data which will be plotted. This is another option, but the data can also be imported from plain text files. In order to do all this hierarchical data import in an easier way, the developers of Krona has done some Perl scripts called KronaTools that can be downloaded from its GitHub.

```
<krona>
    <attributes magnitude="grams">
        <attribute display="Grams">grams</attribute>
        <attribute display="% Daily Value">dva</attribute>
    </attributes>
    <color attribute="grams" valueStart="0" valueEnd="55" hueStart="120" hueEnd="240"></color>
    <datasets>
        <dataset>Brand X</dataset>
        <dataset>Brand Y</dataset>
    </datasets>
    <node name="Granola serving">
        <grams><val>55</val><val>55</val></grams>
        <node name="Protein">
            <grams><val>5</val><val>6</val></grams>
        </node>
        <node name="Carbohydrates">
            <grams><val>39</val><val>41</val></grams>
            <dva><val>13</val><val>14</val></dva>
            <node name="Sugars">
                <grams><val>3</val><val>5</val></grams>
            </node>
            <node name="Dietary fiber">
                <grams><val>4</val><val>8</val></grams>
                <dva><val>16</val><val>32</val></dva>
            </node>
        </node>
        <node name="Fats">
            <grams><val>8</val><val>7</val></grams>
            <dva><val>12</val><val>9</val></dva>
            <node name="Saturated fat">
                <grams><val>2</val><val>1</val></grams>
                <dva><val>10</val><val>5</val></dva>
            </node>
            <node name="Unsaturated fat">
                <grams><val>6</val><val>6</val></grams>
                <node name="Polyunsaturated fat">
                    <grams><val>3</val><val>4</val></grams>
                </node>
                <node name="Monounsaturated fat">
                    <grams><val>3</val><val>2</val></grams>
                </node>
            </node>
        </node>
    </node>
</krona>
```

Figure 18. *Excerpt of Krona XML*

Once the taxonomic tree is created, it is encoded in XML in a node structure. The tag hierarchy of the structure of the data needed by Krona to be able to represent is very customizable. The users can add as much attributes as they want with any name and with any kind of value or list of values. In the example of the Figure 18 there are two attributes, “Grams” and “% Daily Value” declared, the first one is the magnitude. The magnitude attribute means that the values of this attribute will be used in each node of the chart to compute the percentage of the total in each level, as it can be seen in Figure 19. Below the declaration of the attributes, there is a line that makes available the possibility to color the chart in base on a value of an attribute. With this option Krona is able to create a color gradient to color each node by this value. Inside this tag, the attribute is assigned and used to color the node, the range of value of the attribute and the options to control the range of the color gradient are also established there. After this, in the XML file is the declaration of the datasets, in this example case there are two different datasets. Next, the XML part transformed of the data that is represented in Figure 19 is written. As it can be seen, this part of the XML is made up by a nested “node” tag with the tags of the attributes. Each attribute tag has inside another tag with the name “val” with its value. In the case of Figure 18, in each attribute tag there are two “val” tags, this is because in the example there are two data sets represented. Therefore, the first “val” tag has the value of the node of the first data set and the second the value of the node of the second data set.

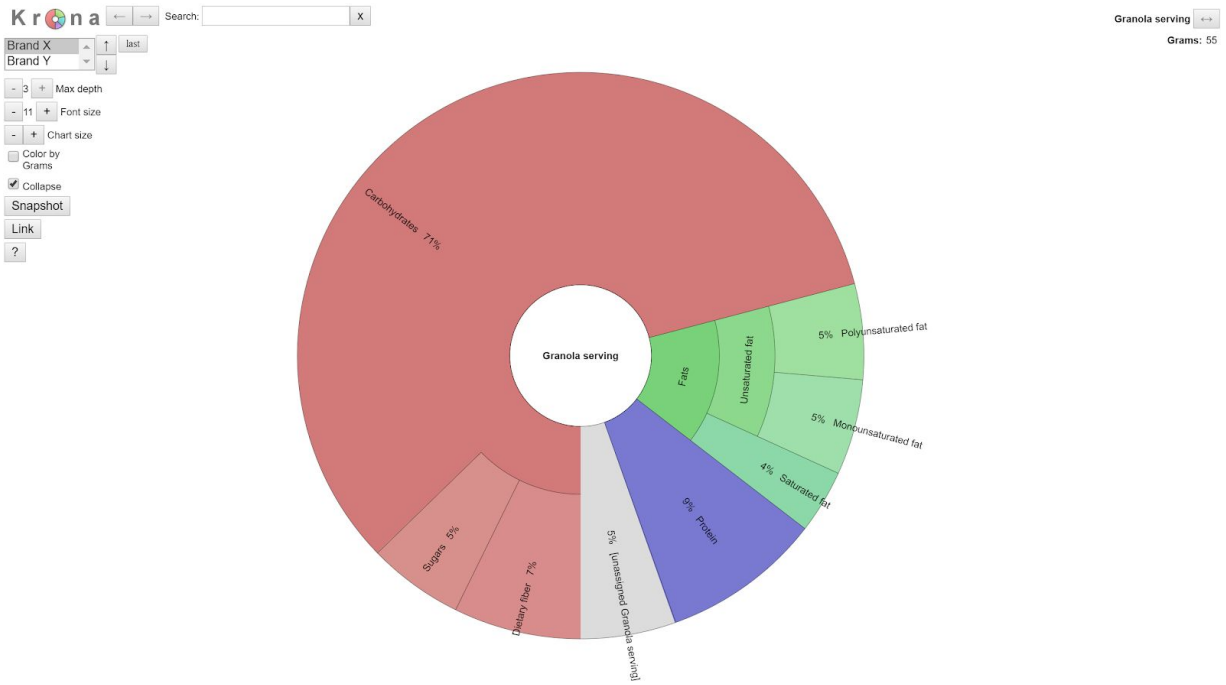


Figure 19. *Representation in Krona of the whole data of the Figure 18*

This XML code is embedded in an XHTML document, as it can be seen in Figure 20. This file is generated by using very widespread technology such as HTML5 and JavaScript. This file is structured as a normal HTML, as it can be seen in Figure 20. It has a body part with the XML and head part above with the JavaScript code, which is used to render the chart with this information. This allows that no installation is needed to open it. Moreover, as most of all the browsers have JavaScript interpreter [12], the file is compatible with the most used browsers and its earlier versions [11]. Both the portability and the great compatibility make Krona a very useful graphic tool, providing as much information as possible of each category of each rank of the taxonomic classification of a metagenomic sample. These characteristics help to make it easier to share, integrate with existing website and to visualize.

Now that the architecture of Krona has been explained, it is time to explain how the data in Krona is represented graphically. This technology uses a sunburst chart to plot the result of a taxonomic classification which, as it has been explained before, is a mix between a traditional pie chart and the modern TreeMap, as it can be seen in Figure 19. With this hybrid, the developers of Krona exploit the radial space-filling (RSF) [13] display. This kind of visualization uses the angular space and the different levels of the chart to show hierarchical information. In addition to that, the formed sectors have a size proportional to the number of reads that

correspond to each classification in the sample, showing in this way the differences in counts between categories. However, since typical lineages of the metagenomic taxonomy have a lot of ranks, RSF is unusable because the chart is very big and the information in the outer slices is difficult to see. Because of this, Krona enhanced the competence of RSF display with some useful tools to navigate through the chart. It has a great visual design, due to the proper coloration of the nodes using a novel algorithm and because not all the ranks are represented. In the cases that some contiguous generation of sector have the same amount of reads, the deepest one is collapsed, the result is that only the most specific one is shown. This option can be enable or disabled from the Krona chart's interface. But this limitation in the representation of the complete lineage makes up for the implementation of a polar-coordinate zooming, with which the user can navigate through the chart zoom in and out nodes. Moreover, Krona saves a history of all the actions that have been made by the user, which can be undone or redone using the arrows arranged in its interface. Regarding the navigation through the chart, Krona also offers a search option and small circles with more information about the node zoomed that appears while the user navigates through a metagenomic lineage and they represent each parent rank of the actual node, and the user can click on them to go to the respective rank and plot it on the chart.

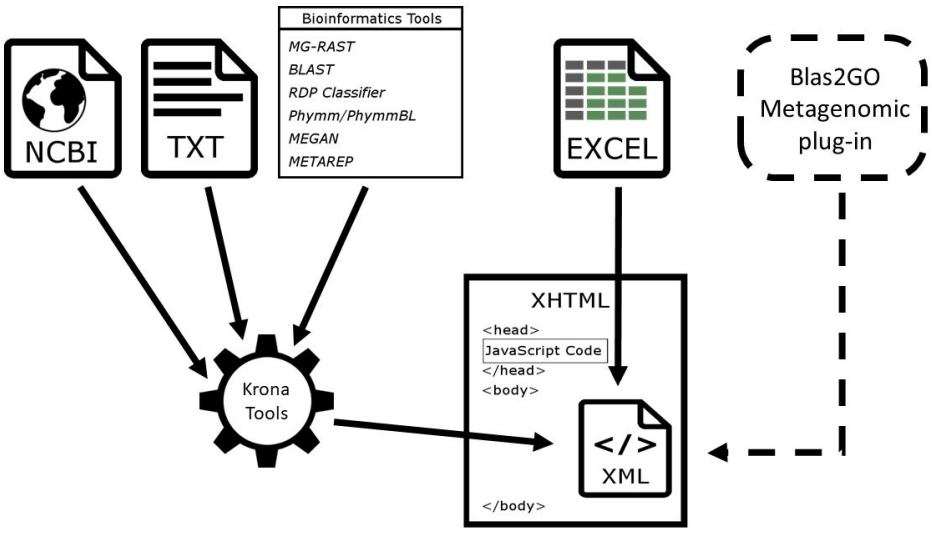


Figure 20. *Graphic diagram of the inputs of Krona and the structure of the XHTML file.*

3.1.6. Study of the Visualization Tool

	Complete Lineage	Various Samples in one Graph	Information provided
Node-Link Tree	Yes	Yes	Proportion of the number of reads in respect to the total
Table	Yes	Yes	Number of reads
Matrix	No	No	Number of reads
Pie Chart	No	Yes	Number of reads, percentage and proportion of the total number
Donut Chart	No	Yes	Number of reads and proportion of the total number
Bar Chart	No	Yes	Number of reads and proportion of the total number
Stacked Bar	No	Yes	Number of reads and proportion of the total number
HeatMap	No	No	Proportion of the total number
Sunburst	Yes	No	Proportion of the total number
MetaTreeMap	Yes	Yes	Taxonomy ID, rank, number of hits and percentage of the total count in the sample and in the metagenome
Krona	Yes	Yes	Number of reads, number of unassigned reads, rank, taxonomy ID, percentage of the total in each previous rank

Table 1. *Comparison between all the charts explained*

After all the research of how the taxonomic classification data of the result of a metagenomic analysis could be represented and see what is the situation of the visualization tools nowadays, it was concluded what type of graph is needed. It should show all the lineage at once. It should have the option to plot various samples in the same chart. It should provide as much information as possible of each category of each rank of the metagenomic sample. Ans it should be easy to understand and user-friendly.

In order to be able to choose the best visualization tool among the previously tested ones, a comparative table has been made (Table 1). In this table, each graph was evaluated according to the characteristics defined in the output requirements section, except for the similarity with the Blast2GO chart because the interface chart can be modified to have this sameness.

The conclusion obtained after this comparison was that the best options to represent metagenomic data were the Node-Link Tree, the table, the MetaTreeMat tool, and Krona. All of these charts display the whole lineage of the sample and they have the capability to represent various samples in one graph. Among them, the table was dismissed because KrakenObject is already represented with a table in Blast2GO. And the Node-Link Tree was rejected too because the other two options left can display more information about the classification. Finally, Krona was seen as the best option over the MetaTreeMap tool. The compelling reason was that Krona has an interface more user-friendly than MetaTreeMap. The first time a user faces MetaTreeMap it is difficult for their to understand what information is being represented. Additional information is needed in MetaTreeMap to explain the visualization of the data, this makes the tool less user-friendly. Moreover, Krona is very portable and interactive. Also, the user does not need a lot of time to get used to its interface.

3.2. Programming Technologies

Apart from the graphical tools to represent the results of the classification, more programming technologies have been used in this project. And now, all these tools, platforms and development conditions will be explained.

3.2.1 Technologies Used

Even though Blast2GO is a platform-independent software, the project was developed on the Linux 64-bit version of the application, more concretely Xubuntu in its 16.04 version was used. This operating system is based on Ubuntu but the main difference is that Xubuntu uses Xfce desktop environment for the user interface and Ubuntu uses GNOME desktop, so both operating systems are more or less the same. This operating system was selected because it is open source, this fact makes it possible to have a lot of documentation about it and helps to

understand it, another reason is that it is very secure and it is used to power the development machines and servers of academic, corporate, and scientific organizations of every size.

In all projects, which involve software programming, it is convenient to have an integrated development environment or IDE, for this project it was defined the usage of Eclipse in it 4.7 version called Oxygen. The main reason is that the bioinformatic platform Blast2GO was migrated to the Eclipse RCP [9] in 2014. This IDE is open source, platform-independent and has a compiler and interpreter inside, this will help to see the errors faster, and to solve them quickly. It has a powerful debug tool which is very easy to use.

Eclipse can be used to develop applications in a lot of programming languages using plug-ins, but for the develop of this application, just Java and JavaScript were used. Another reason why Java was selected for this part is because the algorithm is inside a plug-in of the software Blast2GO, and this platform is all programmed in this language [9]. This object-oriented programming language works in almost all the platform, which makes Blast2GO platform-independent.

As it was said before, JavaScript was also used in the project, that is due to some changes in the Krona chart has to be made and, in order to do it this programming language should be used. In the Krona subsection, it was explained that Krona is an XHTML document which has a JavaScript code embedded, this code controls all the appearance of the chart and all that is needed to structure it. Therefore, if some change in the style or in any of its settings is needed, it has to be done in JavaScript.

As well as JavaScript, Java and a number of libraries of this language were used to treat the data and the style and functionalities of the chart, HTML and CSS were utilized in the project, more concretely, in the XHTML file of the Krona chart..

3.2.2. Blast2GO SDK

As it was said during all this work, the project is integrated into Blast2GO as a new plug-in of the platform. Therefore, the Blast2GO API was used in order to know how to develop this software component which gives a new feature to the platform using the Blast2GO SDK.

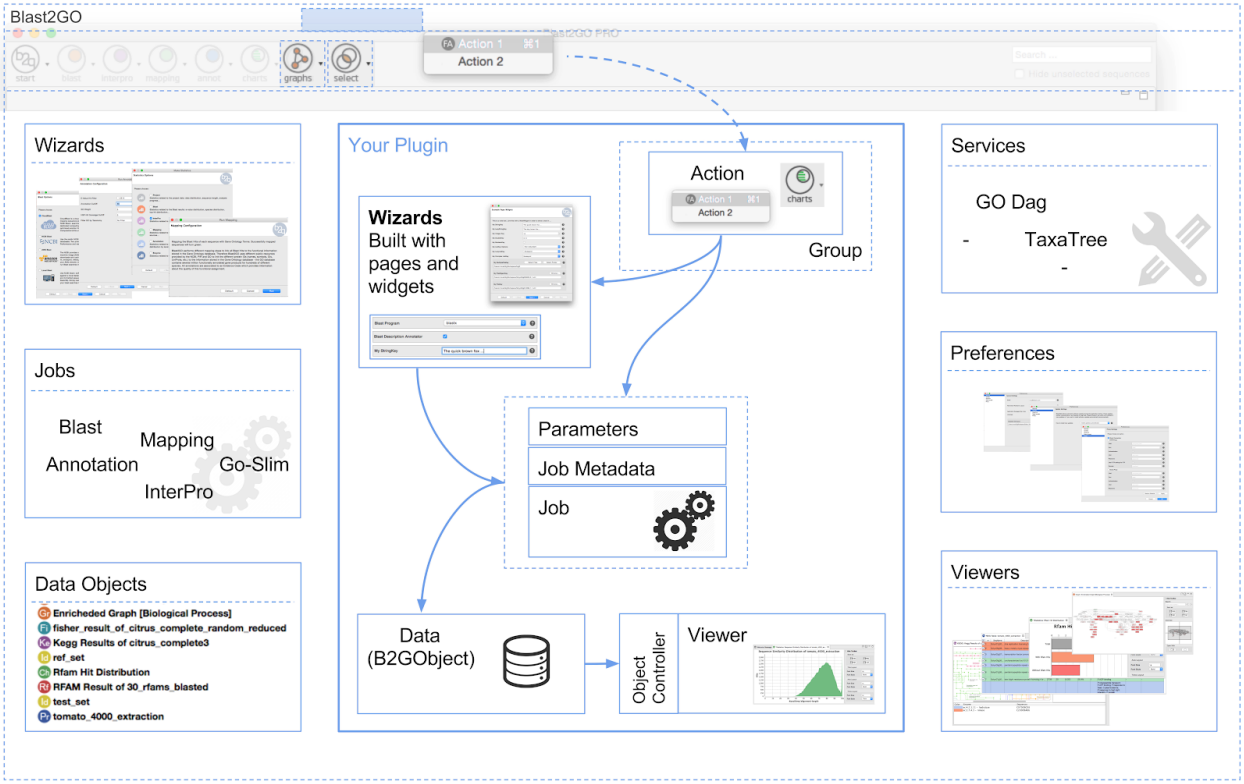


Figure 21. *Overview of the API parts of Blast2GO.*

It is interesting to explain how the structure of a plug-in in Blast2GO is and how it works. As it can be seen in Figure 21, this structure is made up by the connection between a lot of elements and the usage of services that interact to achieve the task. All of this elements and services are packages in the plug-in project that can be implemented and used with the Blast2GO SDK. Inside the plug-in project, there is a library showing all the java configuration of the plug-in, another file to show all the dependencies this plug-in has with other plug-ins and a folder with a file called “MANIFEST.MF” where this dependences can be edited.

According to the figure and with the project explained in this report, a Blast2GO plug-in can implement the following elements:

B2GServices

A service is the unification of external resources so that several Blast2GO plug-ins can access them transparently. In this project, the Taxonomy Service is used. It provides methods to obtain the name of the rank of a certain taxonomy ID, the taxonomy ID of its parent or to obtain its scientific name.

B2GJob

This is the main package of the plug-in because (Figure 22) it contains the programming code of the algorithm. In this project, the algorithm is the treatment of the data to transform it into the proper input to be represented in a Krona chart. It also defines the metadata, which contains the specification of the input and the output of the algorithm, the B2GJob and the parameters. In this case, it is where the Taxonomy service and the Kraken object are defined as an input of the job. Other reason of the importance of the Job class is because it is where the parameters are established, which are the attributes needed by the algorithm to be able to execute that can change the behaviour of it. For example, in the parameters class is where the names of the data sets of the Kraken object are loaded. Moreover, in the B2GJob is where all the interactions with the users, as the progress bar showed of the algorithm, are programmed.

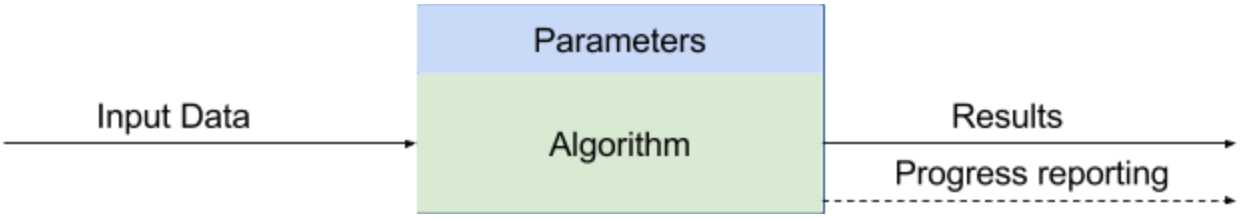


Figure 22. *Diagram of the structure of the B2GJob*

B2GAction

This part of the plug-in is used to define where menu items are located in Blast2GO. Specifications as the position of the menu button and the name are defined here. As Figure 23 shows, in this project, it was defined that the algorithm to generate the Krona chart can be accessible just from an open Kraken object in the Blast2GO interface. Hence, the B2GAction part is linked with the metadata of the B2GJob. The job specified in this metadata is the one that will be execute by the action.

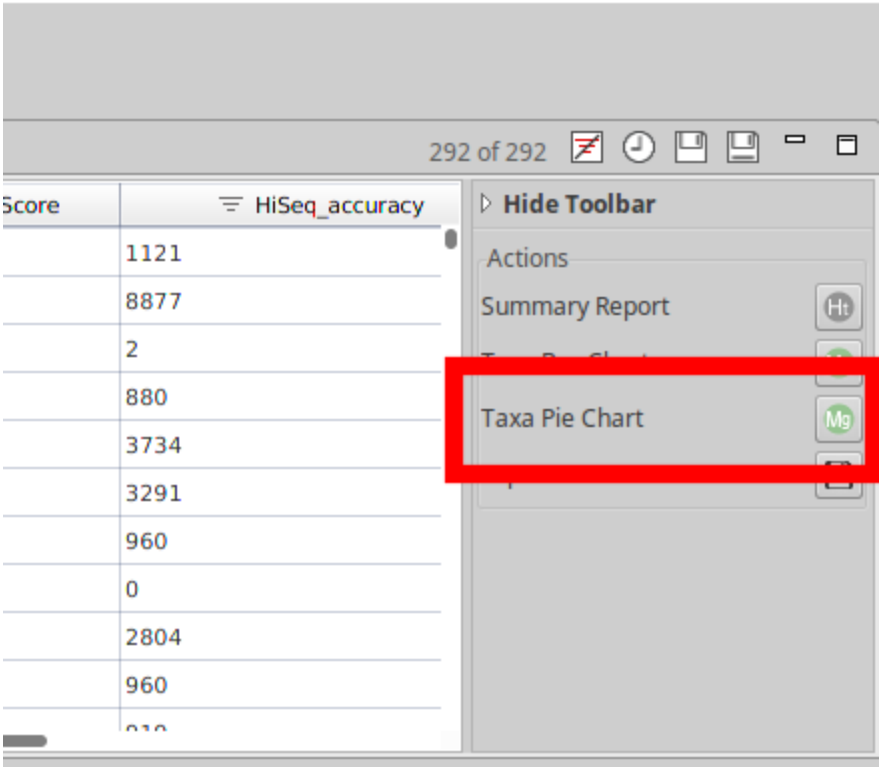


Figure 23. *Detail of the side panel of a Kraken object*

B2GWizard

A wizard is a graphical user interface which appears when an action is launched and which allows to modify the Job parameters before executing the Job. It is assigned to a B2Gaction and connected with the B2GJob parameters. A B2GWizard can be composed with multiple pages and the user can switch between them using the “Back” and “Next” buttons. Each page contains widgets linked to the algorithm parameters allowing the user to adjust them to obtain the desired result.

In the Taxonomy Chart project, when the user needs to modify some parameters of the algorithm, they have to click on the button of “Taxa Pie Chart” (figure 23) and a wizard (figure 24) to concretize this information will appear. As it can be seen, it is formed by a multiple list and a checkbox. In the first widget the name of the samples in the Kraken object is shown in a dropdown list. The “Single Chart” checkbox lets the possibility to create a Krona chart showing the distincts datasets of the Kraken object if it is checked, or to create one Krona chart for each sample respectively.

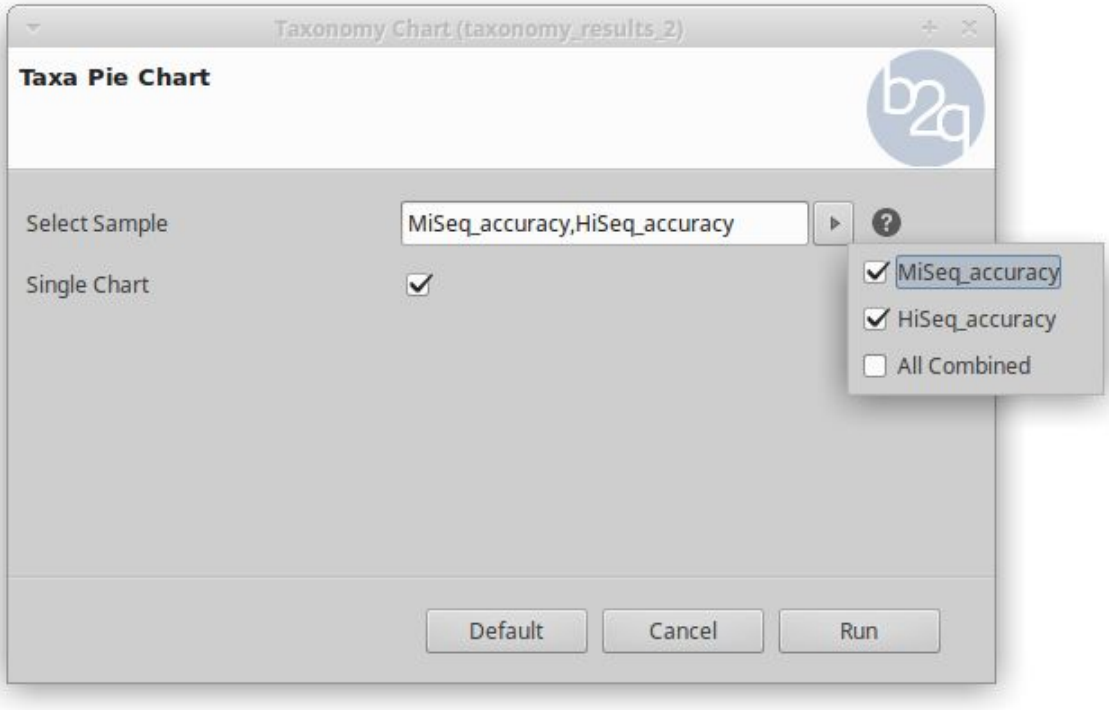


Figure 24. *Wizard of the taxonomy chart in Blast2GO*

B2GObject

In the development of a new plug-in in Blast2GO, usually it is necessary to create a new data model which can cover the necessities of this new feature. In Figure 25 there are the connections and classes that have to be implemented or extended, with some information of what each one does, in order to create a new B2GObject. In this project a new B2GObject, called “KronaChart” was created, in order to save and load this XHTML file.

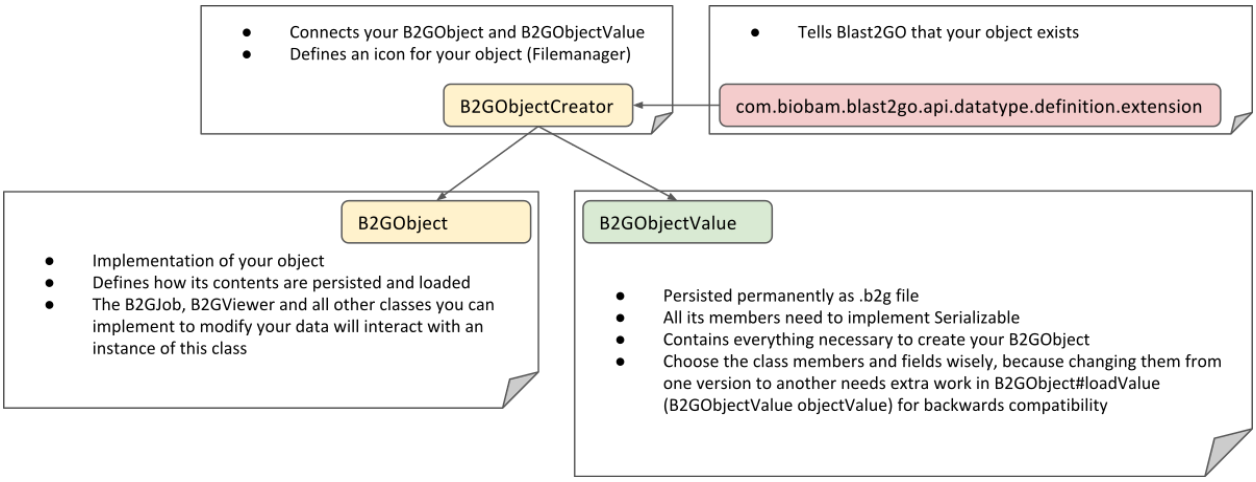


Figure 25. *Overview of an object in Blast2GO*

B2GViewer

The B2GViewer is the graphical part of the plug-in. It is linked to the B2GObject and it defines how to display the content of a data type object in Blast2GO. In the Taxonomy Chart project it was established that a new tab of the Blast2GO interface will be opened as a result of the taxonomic classification. If the user accesses to the algorithm with the button of the sidepanel in Figure 23, or if the user load a chart previously saved. This tab has an embedded Chromium browser which can show the XHTML file.

4.DESIGN

In order to integrate Krona in the Blast2GO metagenomics pipeline, it was necassy to think of a proper data structure to correctly present its graphical visualization. Also, Krona`s esthetics were modified to integrate seamlessly into Blast2GO.

4.1 Krona's Interface Redesign

Blast2GO has a lot of graphical solutions to provide to their users a very diverse representation of the results of any of their analysis.

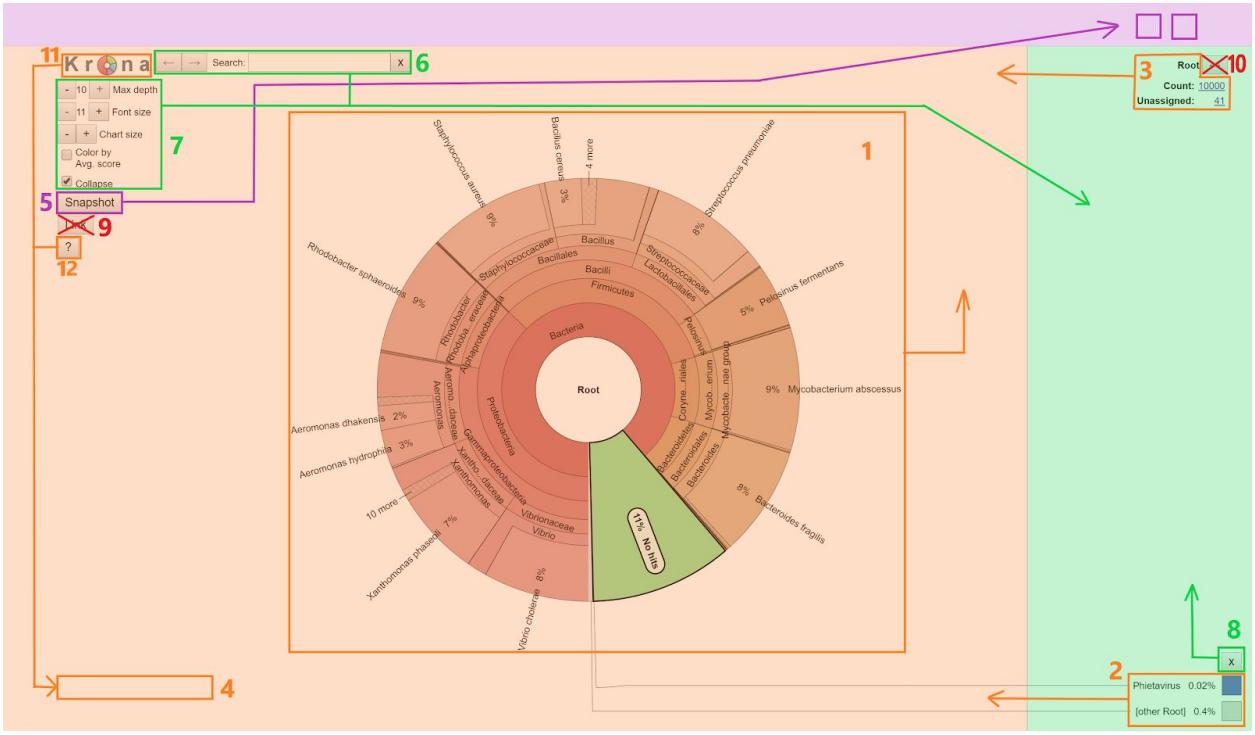


Figure 26. *Overview of Krona's interface modifications.*

As it can be seen in Figure 26, Krona's interface is substantially different to the Blast2GO charts. To not break the visual identity the whole interface of the Krona chart has to be modified. This difference is because Krona comes from an XHTML file, but the Blast2GO chart were developed using the library JFreeChart and the graphs the library Yworks. Moreover, all the HTML part of the view is created by the JavaScript code embedded in the head section of

the file. To carry out the modification of Krona's interface, this code was changed. Also, a CSS style region was created in the head section of the XHTML file. In this section, classes were created and used to establish a style guideline similar to the Blast2GO's charts to the elements of the interface.

Originally, all the elements that can be seen in Figure 26 were added in the body section of the XHMTL file. This section was split up in two panels the side panel, which is the green section in Figure 26, and the main panel, the orange section. Once Krona was integrated in Blast2GO, a toolbar, which is the violet section, was added modifying the B2GViewer using Java. All the elements which can be seen in Figure 26 were redistributed between these sections and as it is described next:

- the following elements were added in the main section:
    1. Krona chart.
    2. The legend of the small nodes in Krona chart.
    3. The information of the node in the center of the Krona chart or focused element.
    4. A label with the text "Powered by Krona" was created as a result of the combination of element 17 and 18. The first was the logo of Krona and the second was a button with a question mark. If the users clicked in any of both elements, they were redirected to the Krona's wiki in GitHub.
- In the **toolbar** element 5 was split in two new tools with which the user can export the Krona chart to PNG or PDF. In the Figure 27, an image and PDF document icon can be seen, they correspond to the new tools respectively. Moreover, there are a "Save" and "Save As" icon.
- In the **side panel** two groups of elements were created and divided with a darker line, as it can be seen in Figure 27, trying to imitate the setting groups of the Blast2GO charts:
    - Element 6 of Figure 26 sets up the first group called "Search and Navigate". This group was the one with the settings to navigates through the charts and search for an specific information. Also, A label explaining the functionality of the arrows was added next to them.

- The other group has the title “Chart Layout” and it is made up by the following elements.
    7. All the setting related with the modification of the font size, the chart size, the checkbox to color the chart by average score and to collapse the nodes.
    8. The button to show or hide the legend of small nodes. Moreover, it was changed to a checkbox.

Furthermore, some elements were removed to the original Krona’s interface. The element 9 of the Figure 26 was a link button that just provided an URL of the chart, and this functionality has not utility once Krona is integrated into the Blast2GO platform because the user can already save it and share it. The element 10 was also removed. This element was a button with which the user could expand a node focused in the Krona chart, but it can be obtain the same result if the user double click on the node.

Figure 27. *Final design of a Krona chart integrated in Blast2GO.*

The Figure 27 shows the result after all the modifications in the interface of the chart. The similar style with the Blast2GO's charts can be seen easily. Therefore, Krona Chart was perfectly integrated in a visual way into the platform.

Another thing that was changed of the original Krona are the colors of the chart when it is colored by average score. In order to change it one code line was modified, the one that control the start and the end of the color gradient boundaries. As it can be seen in the left chart in Figure 28, originally krona painted the chart in a range of colors between red and green. The color range was changed to a gradient from pink to blue. This new range of colors makes it easy to see the difference between the range of scores because there are not intermediate colors as it can be seen in the second chart of Figure 28.

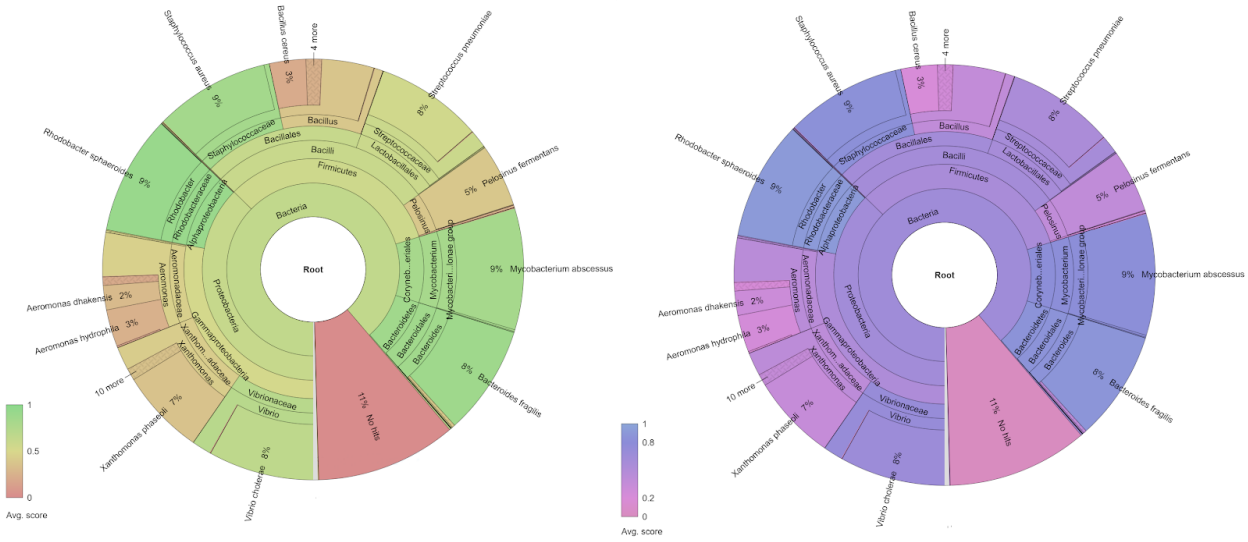


***Figure 28**. Krona chart before and after the color modification*

4.2 Taxonomic data structure

An important part of the project is the design of the proper structure of the data in order to embed it as an XML file into the XHTML file. There are two different data structures in the project very connected between them: the XML structure and the Java structure. Both parts are very similar to each other, but the first one is the result of parsing the second one. Therefore, the Java structure is based in the XML structure.

Figure 29 shows the tag hierarchy of the XML part in the XHTML of Krona chart in the project. As it can be seen in the attributes tag, each node it has at most the attributes defined between these tags. The attributes are the “Count”, “Unassigned”, “Taxon”, “Rank” and “Unassigned”. Each attribute will have its open and close tags inside the node tags, and inside the attribute tags will appear the value between “val” tags. If there is more than one sample, each attribute tags will have a pair of open and close value tags inside, as it can be seen in Figure 18. Figure 29 shows that the open node tags always have the attribute “name”. This attribute is used to show the name of the node in the chart. Moreover, it can be seen that the open taxon tag has an attribute too. This attribute is the “href” and it is used to complete the URL written in the attribute definition. In order to open a window in a browser and show the information of the taxon provided by the NCBI web page once the user clicks the value of the taxon in the chart. It also shows that the “Root” node only has “Count”, “Unassigned” and “Score” attributes. Also, The ”No hits” node neither has “Rank” nor “Taxon”.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        ...
    </head>
    <body>
        ...
        <div style="display:none">
            <krona collapse="true" key="true">
                <attributes magnitude="count">
                    <attribute>magnitude</attribute>
                    <attribute display="Count">count</attribute>
                    <attribute display="Unassigned">unassigned</attribute>
                    <attribute display="Taxon" hrefBase="http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=info&id="
                        target="taxonomy" mono="true">taxon</attribute>
                    <attribute display="Rank" mono="true">rank</attribute>
                    <attribute display="Avg. score">score</attribute>
                </attributes>
                <color attribute="score" hueStart="320" hueEnd="220" valueStart="0" valueEnd="1" default="false" ></color>
                <datasets>
                    <dataset>hiseq_with_score_2</dataset>
                </datasets>
                <node name="Root">
                    <unassigned><val>41</val></unassigned>
                    <count><val>10000</val></count>
                    <node name="Bacteria">
                        <count><val>8877</val></count>
                        <unassigned><val>12</val></unassigned>
                        <score><val>0.649218</val></score>
                        <taxon><val href="2">2</val></taxon>
                        <rank><val>superkingdom</val></rank>
                        <node ...>
                        </node>
                    </node>
                    ...
                </node>
            </krona>
        </div>
    </body>
</html>
```

Figure 29. *Example of an XML of the data part of the Krona XHTML in the Taxonomy Chart project.*

Each node tag corresponds to a taxon obtained in the result of the taxonomic classification. Therefore, the data part of the XML is a nested node tag tree which has attribute tags inside. The value of the Count attribute is the result of the sum of all the read classified in

more specific ranks and the reads classified in this taxon. The last ones form the attribute unassigned of the node which is the number of reads of the sample that couldn't be classified more specifically. These two attributes are calculated in the Java package of the taxonomy tree.

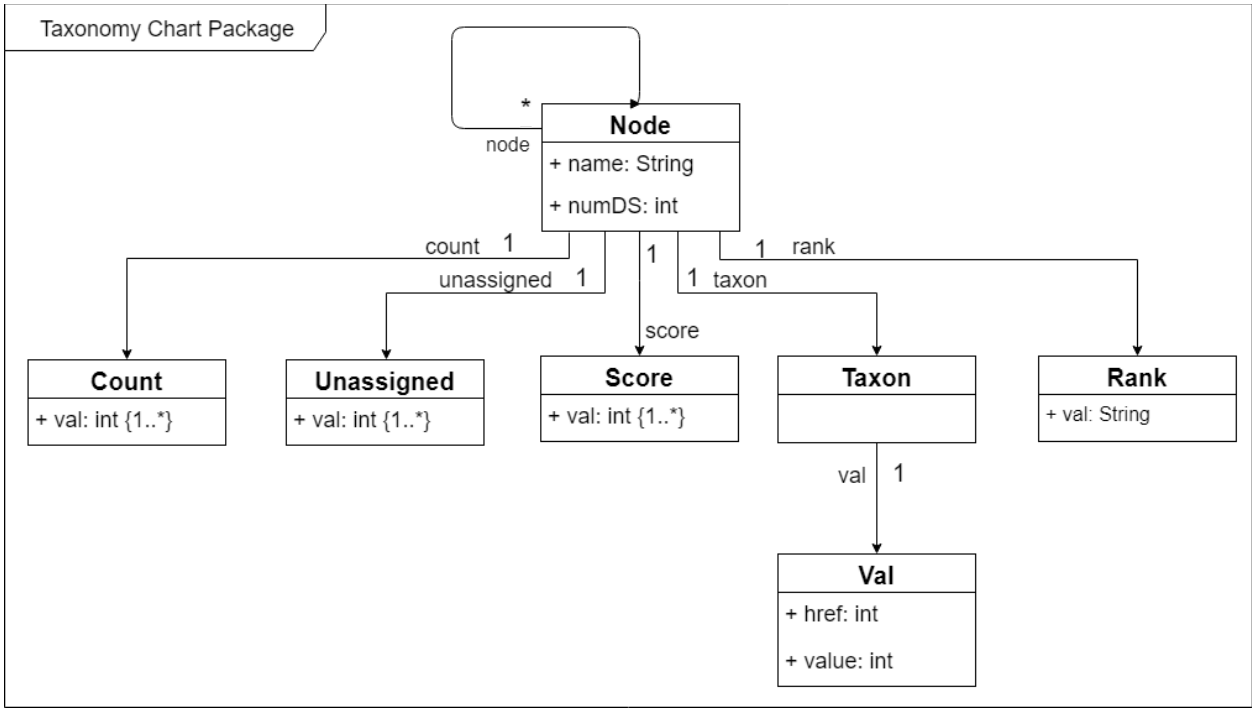


Figure 30. *UML diagram of the taxonomy tree package.*

In order to build up this structure shown in the Figure 29, a package with seven classes was created. As Figure 30 shows, there are one class for each attribute of the node tag and a value class connected with the taxon one. This last class it was needed in order to have the attribute "href" in the open "val" tag of the taxon tag. As the structure is created in Java, a parser to XML was needed. The parser is provided by "javax.xml.bind" library and transforms Java objects into an XML structure. It has the following basic elements:

- **RootElement**. In the Java code is the name on the class that will be transform in the XML tag with the same name as the class. The seven classes are defined as Root elements in order to generated the attribute tag with its value inside the node tag.
- **Attribute**. A member variable in the Java class declared as an "Attribute" will be transform into an attribute of the RootElement open tag. An attribute in an XML tag appears inside the tag not as a nested tag inside of the previous one. In the "Node" class

the attribute “name” and in the “Val” class the attribute “href” were assigned as “Attribute”.

- **Element**. It is an attribute of the Java class that appears as a nested tag in the XML tag of the RootElement where it belongs . For example, the objects of the classes related with the “Node” class and the “name” attribute were “Elements” of the “Node” class. Once the tree is parsed, each node tag has a tag inside of each attribute declared as a “Element” in the Java class.
- **Transient**. An attribute of the java class that will not appear in the XML file. In the Node class, the attribute “numDS”, which references to the number of the datasets, was declared “Transient”. This attribute only helps to access the position of a given sample in the “count”, “unassigned” or “score” attribute list.

4.3. Data treatment

The taxonomy tree previously presented is filled with the KrakenData and the KrakenScores tables of the KrakenObject. For this task, the algorithm that it will be explained below is used. This algorithm is in the B2GJob class of the Taxonomy Chart project in Java. As input it has a KrakenObject and the TaxonomyService and the output is the XHTML with the data.

This algorithm has a “run” method which is executed in the first place. There, it is distinguished if the user had chosen to do a composed chart or not. Its default is false and in this case the algorithm generates one graph per sample in the KrakenObject. Otherwise, a Krona Chart with various datasets will be generated. The following steps are followed in order to generate a Krona chart.

Create the lineage dictionary

This part is composed by four methods sorted by calling order:

1. **“taxIDFromSample(KrakenObject krakenObject, ListKeyOption sample)”**. The first one takes the KrakenObject and a list of the names of the samples and executes the two other methods of these parts for each sample, if the user had marked the “All Combined” option in the wizard, otherwise, it just call the second method.

2. **“getTaxaDict(KrakenObject krakenObject, String sampleName)”**.

   The second one has the same KrakenObject and a name of a sample of the list as input. At the end of the second method, the third method is called by passing a dictionary of taxonomy IDs as an argument. This dictionary is generated going through the KrakenData of a sample in the KrakenObject by taxonomy ID in the sample. The dictionary elements are tuples whose key is the taxonomy ID and the value is the count of that ID in that sample.

3. **“createDictionary(Multiset<String> taxaIDDict)”**.

   The third method fills the taxonomy dictionary, which is a global variable in the class, with the lineages of the taxonomy IDs of the previous dictionary. The entries of the Taxonomy Dictionary are tuples whose value are the count obtained previously and the key is a string with the following format:

   “number<taxID>string<rank>name//number<taxID>string<rank> name of the parent//…”.

   In the case that the the taxonomy ID does not correspond to a species, the first part of the string is “*number<taxID>string<rank>Other//...*”. The value of this key will be the number of unassigned read of this node . If the taxonomy ID has the value 0, which correspond to the reads that couldn’t be classified, 1, which are reads classified but not associated with an organism or 131567, reads identified as a cellular organism but it couldn’t be classified more specifically, the string key is “*null<taxID>null<rank>No hits//null<taxID>null<rank>Root*”.

4. **“getTaxonomy(int id)”**.

   The fourth method helps the last one to create the whole lineage of the taxonomy ID. The method receives a taxonomy ID as input, then runs backwards through the lineage of the ID obtaining its parent taxon. The rank of the parent taxon must always be among those in the RANKS constant. There are seven permitted ranks: species, genus, family, order, class, phylum, superkingdom. If this condition is met, it is added to the string in the format mentioned previously. This occurs until the parent taxon belongs to the superkingdom range.

Therefore, the algorithm starts with a KrakenObject an a list of the names of the samples analyzed and a dictionary with the lineage of each taxonomy ID of the KrakenOject as a key and their count in a sample as the value is obtained as a result of the first part of the job.

Creation of the taxonomy tree

“TaxonomyTree” is a global variable in the class and is unique by Krona chart. Therefore, if the user chose the option to create one Krona chart by sample, the taxonomy tree would be initialize between the creation of the second chart.

The method in charge of creating the tree receives as input the taxonomy dictionary, the number of the sample that will be added to the taxonomy tree, which in case of being the first one will be 0, the number of samples that there are, and a dictionary where the taxonomy tictionary of the sample that has been added to the taxonomy tree will be added (“taxonomyTree(Map<String, Integer> taxonomyDict, Map<String, Integer> dataSetDict, int idDS, int numDS)”). The last argument mentioned will only be used if it is not the first sample to be added to the taxonomy tree, as the method distinguishes between whether or not it is the first sample. This dictionary is used to check if the taxonomy ID already has the node in the taxonomy tree or not. If it does, the count would only be added to the list of counts of the node at the position indicated by the number of the sample passed as an argument. This speeds up the creation of the taxonomy tree. The number of samples is used to create the count node, average score and unassigned lists with the required size.

The construction of the taxonomy tree begins with the first branch. To do this, “count”, “unassigned” and “score” variables are initialized and a node is created. Next, the first key of the taxonomy dictionary is taken and an it is divided per rank into a list using the string “//”. The leaf of the lineage is obtained in the first position of this list. The leaf is also divided into a list using the string “<taxID>|<rank>” to obtain the taxonomy ID, the rank and the name to add it to the node. If the name is "Other", the node is set to null and the count obtained through the taxonomy dictionary is added as “unassigned”. Otherwise, the information is added to the node created. Afterwards, the remaining lineage is covered, and the following is done for each ancestor. First the last node created is added to the branch if it is not null. If it is null, it means that the previous node had the name “Other”. Therefore in the creation of this ancestor node the unassigned will be other than zero and when the node is filled, it will not have any child nodes. This will also happen if the ancestor is named "Root". Otherwise, the node is filled normally. This node will be part of the branch in the next iteration. This process is repeated for each key of the Taxonomy Dictionary, with the difference that, in the loop after the first leaf node, it is checked if the ancestor node already exists in the tree. If it already exists, the node will be

added as a child of the existing one and will be moved to the next lineage. Otherwise, the node is added to the branch, a node is created with the ancestor information, and the next ancestor in the lineage is passed on. After iterating over all keys in the taxonomy dictionary the taxonomy tree is completed.

Creation of the Krona Chart

Once the Taxonomy Tree is filled with the information of one or various samples, it can be parsed into XML format. Firstly, the count variable of nodes in the tree are covered by sample in order to compute the correct count in each position of the list. For this, a method in the Node class ("getCountByDS (int id)") is called with the number of the sample as argument. This method opens child nodes until it finds a leaf node and adds to the parent the count of the child and the unassigned number, if it has. Then, the method "getScoreByDS (int id, KrakenObject krakenObject, String sampleName)" to set the score value to the nodes is called. It receives the KrakenObject, the name of a sample and the number of this sample as arguments. The method goes through the nodes of the taxonomy tree searching, into the KrakenScores, the value assigned to the taxonomy ID of the node in the sample. After these two steps, the taxonomy tree and the names of the samples are converted to XML format and added to the Krona XHTML file. This file is transfered to the B2GViewer and displayed.

4.4. New feature of Krona chart

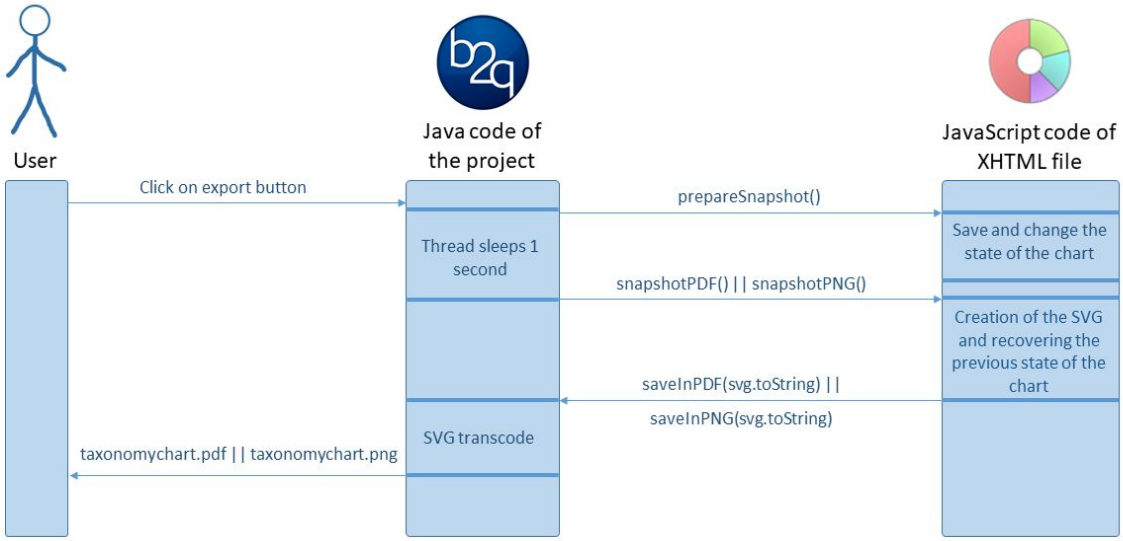


Figure 31. *Communication diagram of the export of the chart feature.*

One of the output's requirements was that the visualization tool could export the graphical representation into PNG or PDF format. In the original Krona interface, there is a button that takes screenshots of the graph in SVG. As already mentioned in this section, this button was replaced by two buttons that export to PDF and PNG. In order to do that, a connection between the script of the XHTML file and the View class of the Taxonomy Chart project was established. As Figure 31 shows when a user clicks on any of the export buttons, the Java code calls a method created in the JavaScript code called "prepareSnapshot()" and the java thread sleeps for one second. This method saves the state of the chart in this moment and then clears the search bar, deletes the watermark grid of the nodes and makes the chart and the font size bigger in order to show the chart in best conditions, as it can be seen in Figure 32. The JavasScript method "snapshotPDF()" or "snapshotPNG()" is called afterwards, depending on what the user clicked. Both methods create a snapshot in SVG of the chart in the state described earlier, and then recover the previous state of the chart. This SVG is transformed into a string and passed to a Java method called "saveInPDF" or "saveInPNG". These methods use a library called "org.apache.batik.transcoder" to transform according of the format in which the SVG is going to be saved. Finally, the file is saved in the file with the name chosen by the user in the wizard.

5. RESULTS

The final step of the Taxonomy Chart project was to perform a metagenomics analysis in order to see the result of this project and to try out the correct operation of the new plug-in into the Blast2GO platform. The taxonomy classification is based in an analysis previously done and reported in a scientific paper [16]. This helps to validate if the new plug-in allows to obtain the same conclusions.

The main objective of the analysis is to demonstrate that the cultural tradition, lifestyle and diet of two different geographical locations can produce differences in gut Metabolites and microbial composition and functions. In particular, The Mediterranean diet rich in fiber, vegetables, and fruits is being compared against a more industrialized diet which is enriched in processed carbohydrates, animal proteins, and fats. Therefore, the metagenomes which have been analyzed correspond to two fecal samples one of healthy adolescents from Egypt (Child 1), representing the Mediterranean diet, and one from the United States of America (Child 2), representing the industrialized diet.

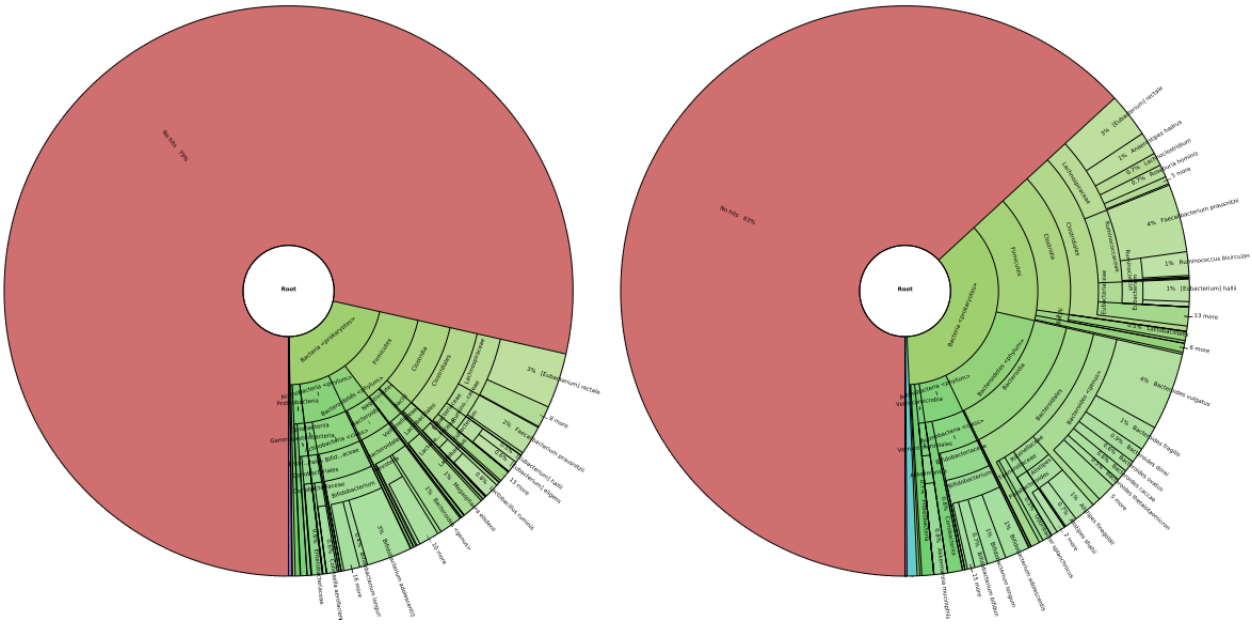


Figure 32. *Taxonomy Chart of the Child 1 and Child 2 respectively.*

For the analysis with the metagenomics tool of Blast2GO, the FASTA files of each child from the database of the European Bioinformatics Institute (EMBL-EBI) were downloaded. The size of these files is 2.5 GB for Child 1 and 2 GB for Child 2. These files contain the nucleotide reads already preprocessed by the pipeline of the EBI. This preprocessing consists of a first low quality filtering with which the reads with low quality ends and with more of a 10% of the undetermined were removed. Then, reads with less than a 100 of nucleotides or and duplicated sequences were removed. Finally, the reads with low complexity regions were discarded.

The big Kraken database which has 100 GB of k-meros from organisms of the superkingdom of Bacteria, Archaea and Virus was used. The analysis ended after fourteen hours. The Krona chart was generated perfectly and opened into a new tab of the Blast2GO.

The first thing that can be seen is that Child 1 has less hits than Child 2 (Figure 32). Also, the paper states that the number of Methanobacteria were statistically significantly higher in Child1. Searching the name using the text field, the node in the chart was being found, and switching between sample, it can be proved that the conclusion was right. The number of counts of Methanobacteria was notably higher in Child1 compare to Child2, as can be seen in Figure 33 and 34. Furthermore, the reads of the Child 1 were classified in this taxon with more confidence than in the Child 2, giving to the conclusion a higher significance. Other conclusions of the paper were confirmed satisfactorily using the taxonomy chart. For example, the Gammaproteobacteria has a higher count in Child 1 and the Verrucomicrobia showed up more in Child 2 than in Child 1.

With these examples, it was demonstrated that the taxonomy chart has a useful role in metagenomics investigations. And it works perfectly, helping the user to obtain meaningful insights. In order to check that all the developed functionalities work properly in the Blast2GO, a series of use case in Jira Software were created and executed satisfactorily by workers of Biobam.

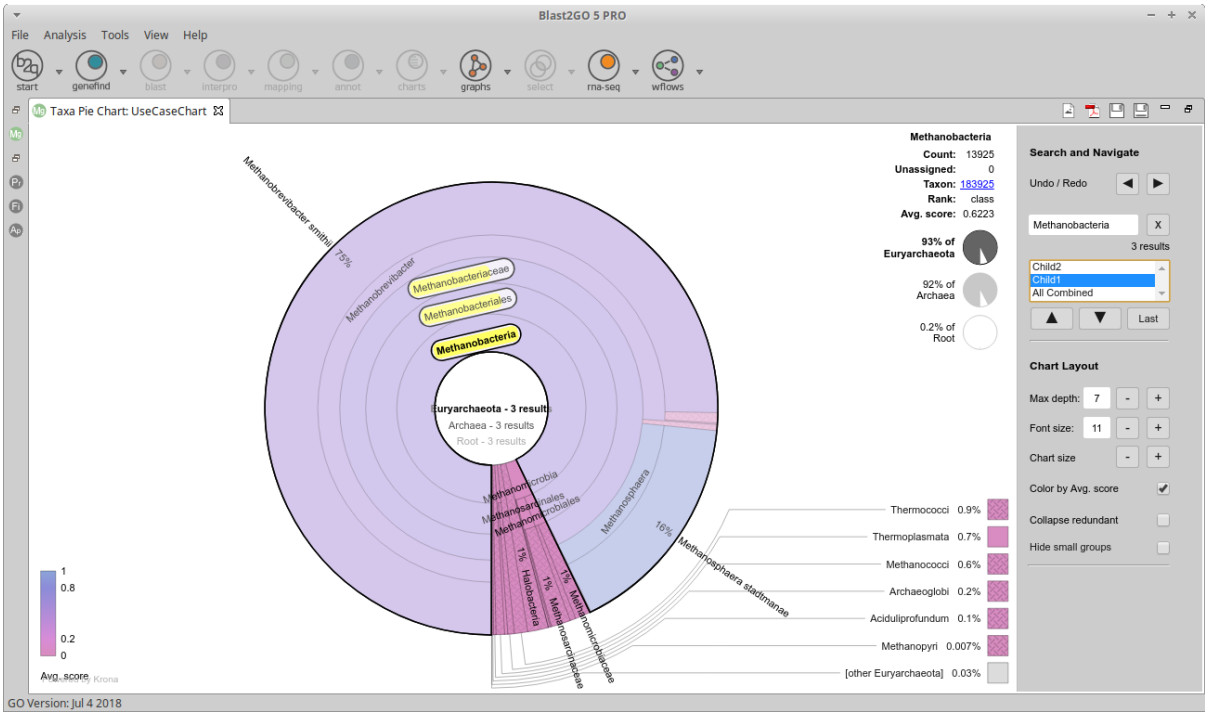


Figure 33. *Representation of the Euryarchaeota in Child 1.*

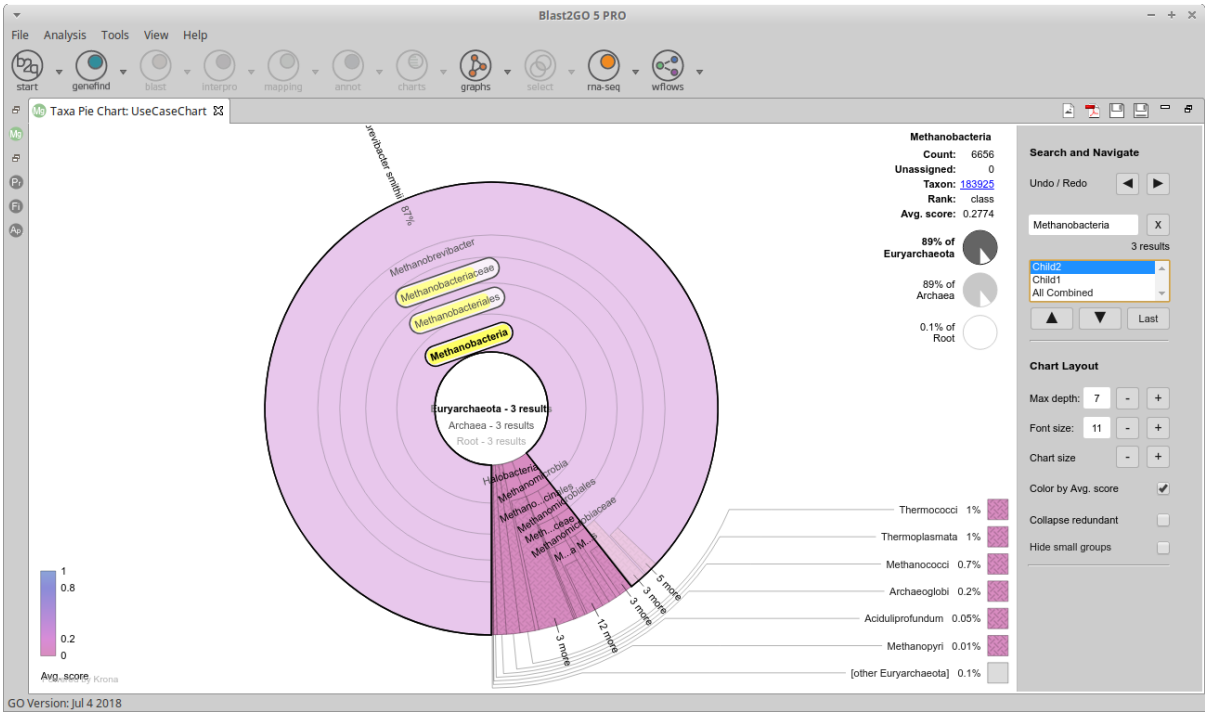


Figure 34. *Representation of the Euryarchaeota in Child 2.*

6. OUTLOOK

Actually, the Taxonomy Chart project could be improved because the sum of the count in each taxon is computed twice. The first time is in the RankedData of the KrakenObject and the second time is in the creation of the Krona chart part of the data treatment, where the taxonomy tree is already filled and the previous step before the transformation in XML format is to sum the counts calling the method of the "Node" class "getCountByDS". Therefore, A way to make this pipeline more efficient and to take less time to show the results was found. In order to accomplish it, instead of computing the second time the sum, the RankedData variable of the KrakenObject could be accessed to obtain the count of the taxon in a certain sample and add the obtained value to the count in the node of the taxon in the taxonomy tree when this node is created. The same can be done to set the average score to the nodes instead of traversing the whole tree once it is generated, using the KrakenScores of the KrakenObject.

The reason why the algorithm was programmed as explained in the design section is because the KrakenObject was developed in parallel by another worker of the company. At the time the Taxonomy Chart project was finished, the KrakenObject had the structure of the KrakenData, which currently is a member variable of the object. Therefore, the sum of the counts for each taxon in the taxonomy tree were needed. Afterwards, it was decided to include the average score and the KrakenObject was modified, but the Taxonomy Chart project continued to function properly.

Another improvement of the project would be the addition of a statistical method of comparison between the different samples that have been analysed. Its function would be to show that the results obtained are biologically relevant and statistically significant. This could be achieved by incorporating the STAMP [17] software package into the project.

7. CONCLUSIONS

At the end of this project, Krona chart has been seamlessly integrated into the Blast2GO software as a part of the new pipeline that Biobam is willing to offer. Therefore, it can be said that the main goal of the development of this work has been accomplished, with all its sub-objectives.

A comparison has been made between various charts widely used in other platforms and solutions that provide metagenomics classification pipeline services. Firstly, the JavaScript library D3.js was chosen to re-implement a handful of the charts seen in the studied guide models, but this idea was dismissed. The experience was useful to confirm that Krona was the perfect option to visualize the data of taxonomic classifications. It is interactive and draws the complete lineage of the read which allows to get a lot of complex information at a glance.

To integrate Krona tool into the Blast2GO's metagenomics workflow it was added as a plug-in using the Blast2GO API as a guide, which is explained in the Biobam developer zone. This plug-in contains an iterative algorithm that retrieves the result data of the pipeline and modifies it to construct a custom taxonomic tree. This information is then transformed into an XML file which is embedded into the final XHTML file. At last, this last file is displayed in a browser window inside of the Blast2GO application.

To conclude this project and to check if it works properly, a complete metagenomics analysis has been performed as a use case of the new workflow added to Blast2GO. As a result, a real in-depth understanding of how a metagenomics analysis is carried out and how to get valuable biological insights was acquired.

8. REFERENCES

[1] [Vélez, D. U. (2009). Metagenómica: Una oportunidad para el estudio de la diversidad microbiana en Colombia. Revista Colombiana de Biotecnología, 11(2), 4-7.]

[2] https://www.definicionabc.com/ciencia/filogenia.php

[3] Oulas A, Pavloudi C, Polymenakou P, et al. Metagenomics: Tools and Insights for Analyzing Next-Generation Sequencing Data Derived from Biodiversity Studies. Bioinformatics and Biology Insights. 2015;9:75-88. doi:10.4137/BBI.S12462.

[4] Thomas T, Gilbert J, Meyer F. Metagenomics - a guide from sampling to data analysis. Microbial Informatics and Experimentation. 2012;2:3. doi:10.1186/2042-5783-2-3.

[5] Jovel, J. et al. Characterization of the gut microbiome using 16S or shotgun metagenomics. Front. Microbiol. 7, 459 (2016).

[6]Florian P. Breitwieser, Jennifer Lu, Steven L. Salzberg; A review of methods and databases for metagenomic classification and assembly, Briefings in Bioinformatics, , bbx120, https://doi.org/10.1093/bib/bbx120

[7] Huson DH, Auch AF, Qi J, Schuster SC: MEGAN analysis of metagenomic data. Genome Res 2007, 17(3):377–386. 10.1101/gr.5969107

[8] Wood, D. E. & Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biology 15(2014).

[9] Nica, Bogdan Robert. Migración de la aplicación bioinformática Blast2GO a un sistema RCP (Rich Client Platform). PFC (2014).

[10] D3.js Home Page [https://d3js.org].

[11] Ondov et al.: Interactive metagenomic visualization in a Web browser. BMC Bioinformatics 2011 12:385.

[12] A. Grosskurth and M. W. Godfrey, "A reference architecture for Web browsers," 21st IEEE International Conference on Software Maintenance (ICSM'05), 2005, pp. 661-664. doi: 10.1109/ICSM.2005.13
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1510168&isnumber=32336

[13] Jusufi, Ilir, Andreas Kerren and Yangheng Wang. “A New Radial Space-Filling Visualization Approach for Planar st-Graphs.” (2012).

[14] Dumont MG, Lüke C, Deng Y and Frenzel P (2014) Classification of pmoA amplicon pyrosequencing using BLAST and the lowest common ancestor method in MEGAN. Front. Microbiol. 5:34. doi: 10.3389/fmicb.2014.00034

[15] Hebrard, Maxime, and Todd D. Taylor. “MetaTreeMap: An Alternative Visualization Method for Displaying Metagenomic Phylogenic Trees.” Ed. John Parkinson. PLoS ONE 11.6 (2016): e0158261. PMC. Web. 4 June 2018.

[16] Shankar, V., Gouda, M., Moncivaiz, J., Gordon, A., Reo, N. V., Hussein, L., & Paliy, O. (2017). Differences in Gut Metabolites and Microbial Composition and Functions between Egyptian and U.S. Children Are Consistent with Their Diets. mSystems, 2(1), e00169–16.

http://doi.org/10.1128/mSystems.00169-16

[17] Donovan H. Parks, Robert G. Beiko; Identifying biologically relevant differences between metagenomic communities, Bioinformatics, Volume 26, Issue 6, 15 March 2010, Pages 715–721, https://doi.org/10.1093/bioinformatics/btq041