Syntenizer 3000
A tool for bacterial genome visualization & analysis

Camous Moslemi

Abstract

Genomic analysis often requires the use of many different tools, and scripts either offered by third parties or custom written for the task at hand. Syntenizer3000 (S3K) started its life as a simple tool for measuring synteny for splitting out paralogs from gene groups in a Rhizobium dataset consisting of approximately 200 sequenced strains. In order to achieve this the dataset, which consisted of PROKKA generated annotated assemblies, and subsequent ProteinOrtho generated gene groups, had to be parsed and stored in internal data structures.

During the course of the project whenever additional analysis of the dataset was required it was convenient to code it as part of S3K as once the genome had been loaded into internal data structures it was easy to manipulate and extract as desired.

This culminated in the tool steadily growing to become quite a versatile piece of software for genomic analysis and visualization of bacterial populations with capabilities including the following:

- Measuring synteny between any two genes in the dataset using two different algorithms that each focus on different syntenic aspects.
- Use the aforementioned algorithm to assign a synteny score to each gene group which measures the average syntenic harmony between the members of gene groups generated by third party tools, such as ProteinOrtho, OrthoMCL etc.
- Use synteny scores to split paralogs out of groups, resulting in gene groups devoid of paralogs.
- Generate detailed circos diagrams of each strain to visualise information such as, synteny, GC3s content, gene abundance across whole dataset, paralog charts and highlight groups of interest.
- Generate synteny maps for each gene group which visualize the syntenic relationship of up and downstream genetic neighbourhood of the genes present in a gene group.
- Generate a genomic Presence/Absence matrix based on gene groups.
- Generate Gene Relational Matrix (GRM) based on presence/absence of strains in gene groups.
- Encode gene presence/absence information as a GAPIT compatible SNP matrix for bacterial GWAS analysis.
- Generate useful statistics for each strain, such as average GC content, total number of contigs, the number of contigs of a certain size etc.
- Use input annotated gff files to produce detailed statistics about each gene group, such as synteny scores, protein products etc.
- Classify contigs based on gene group and synteny information, with accompanying confidence levels, provided some contigs have already been pre-classified using other methods.

1 Methods

1.1 Introduction

For the sake of computational efficiency S3K has been coded in C++, although it is held by a lack of multithreading support. A complete suit of analysis performed on a 200 strain large bacterial dataset can be achieved in a few hours even on a computer with modest specs, although at least 4 gigabytes of memory is recommended.

Syntenizer relies on other industry standard tools to have processed the bacterial genome before it can be loaded and processed. The sequenced genome should already be assembled, annotated using PROKKA and have orthologous gene groups generated for it using ProteinOrtho or a similar tool.

To do any useful analysis S3K first, and at the very minimum, needs one general feature format (gff) file for each strain, as well as a file defining gene groups. Two gene group file formats are supported. The gene groups output file from ProteinOrtho, or an internal format utilized by S3K which is outlined in the manual.

1.2 Synteny

1.2.1 Defining Synteny

We base our analysis of synteny on the concept of positional orthology as defined in a paper by [Dewey 2011]. The paper defines positional orthology, or synteny, as two genes that are homologous to one another, with neighbouring genes which are also homologous to one another (Figure 1).

Strain1: ===A1===B1===C1===
Strain2: ===A2===B2===C2===

Figure 1: An example of two strains exhibiting synteny. The homologous pairs (A1 and A2), (B1 and B2), (C1 and C2) are neighbouring genes in both strains.

If sequence data analysis reveals in strain1 three neighbouring genes A1, B1 and C1, are homologous to the genes A2, B2 and C2 respectively in strain2 then we can say of the three homology pairs not only exhibit orthology but also positional orthology, or synteny.

1.2.1 Measuring Synteny

We base our measurement of positional orthology on orthologous gene groups. We will regard two genes as being potentially orthologous if they are placed in the same orthology gene group. In effect our synteny measuring algorithm will only regard genes in terms of the orthologous group that they belong to (Figure 2).

Strain1: ===group1===group2===group3===
Strain2: ===group1===group2===group3===

Figure 2: The genome of the two strains from Figure 1 regarded in terms of the orthologous group of each gene, rather than the gene itself.

Syntenizer3000 can assign a score to the strength of synteny between two genes. To help illustrate how the scoring algorithm functions we will use an example of two potentially orthologous genes 1 and 2 that both reside in the same group but stem from different strains. We wish to examine synteny between these to genes and for that purpose we will look at their [n] closest neighbouring genes. By neighbouring genes we mean the genes placed directly up and downstream to the gene in question in the genome of the host organism. In Syntenizer3000 the value of [n] is set to 40.

Reimagining our previous example with an [n]=6 will lead to the situation illustrated in Figure 3 where we examine the gene groups of the the 3 closest genes immediately up and downstream of the genes A1 and A2 on their respective DNA strands.

Strain1: ===group7===group6===group5===A1===group2===group3===group4===
Strain2: ===group7===group6===group5===A2===group2===group3===group4===

Figure 3: The 6 neighbouring genes of genes A1 and A2 illustrated in terms of their orthologous group. Perfect synteny exists between the two genes for [n]=6.

The example illustrated in Figure 3 is an example of two genes demonstrating perfect synteny for an [n]=6, because each their [n] neighbouring gene pairs are predicted orthologs.

Synteny analysis of genomes has been employed in studies involving orthology verification [Frantzeskakis et al., 2018], examining chromosomal reconstruction using synteny blocks in distantly related species such as humans and mice [Sinha et al., 2007], and HGT detection [Adato et al., 2015]. Until Syntenizer3000 no synteny based tools capable of assigning a numerical value to the degree of synteny between two genes, and use this measurement to remove paralogs from gene groups has been made available for general research use.

In the course of developing the final synteny scoring algorithm several scoring methods were developed of increasing complexity. Below is an explanation of each.

I. SIMPLE SCORING METHOD

The first method, called SIMPLE, works by for each gene in the [n] neighbourhood of gene A1 trying to find a gene in the [n] neighbourhood of A2 which belongs to the same orthology group. For each match the synteny score between A1 and A2 is increased by 1. So the maximum score possible is [n], indicating perfect synteny in the [n] neighbourhood of both genes, while a score of 0 indicates that no synteny exists between the two genes.

II. OLD SCORING METHOD (deprecated)

A weakness of the SIMPLE scoring method is that the provided score does not say much about potential changes in the configuration of genes in each neighbourhood.

Strain1: ===group3===group4===group7===A1===group2===group6===group5===
Strain2: ===group2===group5===group3===A2===group6===group4===group7===

Consider the extreme example above in which the neighbourhoods of two genes do not overlap at all, but since genes from each group in the neighbourhood of A1 are also found in the neighbourhood of A2 the SIMPLE score would still be a perfect [n].

If the neighbourhoods of the two genes do not align perfectly the scoring algorithm should attempt to take that into consideration and assign a partial match score to genes that are in the same gene group, but are not present in the same order in their respective neighbourhoods.

Let us look at such a scenario below where we only focus on two genes in the neighbourhood of A1 and A2.

Strain1: ===XXXX===group2===XXXX===A1===XXXX===XXXX===XXXX===
Strain2: ===group2===XXXX===XXXX===A2===XXXX===XXXX===XXXX===

In this scenario A1 and A2 share a gene in their [n] neighbourhood, meaning that synteny is present, but the placement order of each gene in their respective neighbourhood does not match exactly. For such scenarios we developed a scoring method named OLD, now deprecated, designed to assign a partial synteny score to such matches. This is because the misalignment could have been caused by a gene insertion or deletion event. Our scoring algorithm still wishes to regard neighbourhoods disrupted by such events as closely related, albeit not as closely related as neighbourhoods that enjoy perfect syntenic alignment between neighbouring genes.

The partial match score is calculated as: 1 / (distance + 1).

In the above example the distance is 1, because the order of the two genes in their respective neighbourhoods differs by one gene. So the score assigned is 0.5, whereas it would have been 1.0 if both genes were located the same distance away from A1 and A2. Bear in mind that the example is only focusing on two genes within the neighbourhood, and the total synteny score between A1 and A2 would depend on the rest of the genes in the neighbourhood.

III. Sophisticated SCORING METHOD

One issue with the OLD scoring method was that gene deletions would result in significantly reduced scores depending on where they occur in the neighbourhood of a gene being scored against another. Consider the scenario below where the gene in group3 has been deleted in the neighbourhood of gene2.

Strain1: ===group2===group5===group3===A1===group7===group4===group6===
Strain2: ===group8===group2===group5===A2===group7===group4===group6===

This will lead to partial scores for each gene on that side of the neighbourhood. For large sizes of [n] this would result in significantly reduced synteny score. The legitimacy of this concern for large neighbourhood sizes became clear after using S3K to visualise synteny in gene groups. When examining the [n]=40 neighbourhood of core groups, which had genes from most strains lots of deletions and insertions were evident.

A better way to score the neighbourhoods of two genes, which would not punish deletions, or insertions, as harshly as the OLD scoring method would be by first attempting to align the neighbourhoods akin to the Needleman-Wunsch (NW) algorithm does for DNA sequences [Needleman & Wunch, 1970].

This is the basis of the SOPHISTICATED scoring algorithm which is a modification of the NW algorithm to repurpose it to aligning gene neighbourhoods rather than nucleotides. The NW algorithm is configured with a gap penalty of -0.5, match score of 1 and mismatch score of 0.

Strain1: ===group2===group5===group3===A1===group7===group4===group6===
Strain2: ===group8===group2===group5===A2===group7===group4===group6===

For the above example the algorithm would align the two neighbourhoods as seen below.

Strain1: ===group2===group5===group3===A1===group7===group4===group6===
Strain2: ===group2===group5=== -- ===A2===group7===group4===group6===

The score assigned would then be 4.5 which is the result of 5 perfect group matches, minus 0.5 penalty for the gap (--) that needed to be introduced to align the neighbourhoods correctly.

One thing to note about this way of scoring synteny is that although it does not punish insertion or deletions harshly, genomic inversions will nevertheless be imposed a heavy penalty.

Strain1: ===group2===group5===group3===A1===group7===group4===group6===
Strain2: ===group6===group4===group7===A2===group3===group5===group2===

Let us consider the scenario above which would result in a very low synteny score even though all the gene groups present in the neighbourhood of A1 are present in the neighbourhood of A2 as well. This is due to the nature of how the NW algorithm works which can only account for insertions and deletions, not inversions.

To account for inversions the NW algorithm would have to be run once twice, once with one strand inverted.

However, this extra step is rather costly as it doubles the required computation time. In order to test its necessity we exported all the gene pair scores calculated for a full run of S3K measuring the average synteny for all groups. Upon comparing scores of both inverted and normal orientations it turned out that out of the 116963805 gene pairs scored only in 157084 (0.134%) of the cases did the reverse orientation score higher. Upon visualisation these examples turned out mostly to stem from small gene groups consisting of two or three genes.

With such a low occurrence it can be argued that the computational penalty to account for them might is not be worth it, and is thus ignored by S3K.

IV. SCORE NORMALISATION

The synteny scoring algorithm described only works for measuring synteny between one gene pair. This is only useful for measuring synteny in gene groups with only two members. However, most of the orthologous groups in our dataset contain many more genes, with some core groups exceeding even the number of strains due to presence of paralogs. Therefore a method was developed to assign a synteny score to each gene group by taking into account how each gene in said group shared synteny with each other gene in that group. Thus the synteny score of a gene group is a measure of the syntenic harmony shared between all the members of said group.

Considering an orthologous gene group with three members, the total synteny score the group is calculated as the sum of synteny scores for each possible unique gene pairing, divided by the number of possible unique gene pairings in said group.

GroupA: **[A1], [A2], [A3]**

[**GroupA**]:Score = (*SyntenyScore*(**[A1]**, **[A2]**) + *SyntenyScore*(**[A1]**, **[A3]**) + *SyntenyScore*(**[A2]**, **[A3]**)) / **3**

We divide by the number of unique gene pairing combinations in the group in order to normalise the scores across groups and make them comparable. After normalisation the score will measure the average synteny of an orthology group, based on the synteny of each gene in the group against every other gene in the same group. This score is comparable across groups, with the maximum attainable value being equal to the size of the neighbourhood [n] for all three scoring methods.

1.3 Paralog Removal

Many techniques to detect paralogs have been attempted, such as by using phylogenies [Kim et al., 2008], BLAST scores [Moreno-Hagelsieb et al., 2007], and even synteny [Lechner et al., 2014]. However, the need for only retaining one copy of a gene from each strain in each gene group is somewhat unique, and has not addressed by any of the aforementioned techniques which are mostly concerned with establishing correct phylogenies among distantly related species. In order to best serve this need an algorithm was developed based on S3K and its ability to measure synteny.

The algorithm works by first removing all paralogous genes from a gene group, pooling them into a separate paralogous group, while retaining all none paralogous genes in a now purely orthologous gene group. Then from the orthologous group the gene which has the highest synteny with every other gene in the orthologous group is picked as a representative of said group.

Next, each gene in the paralogous group is scored against the representative gene using the SOPHISTICATED score. The gene in the paralogous group which scores the highest synteny against the none paralogous group representative is inserted back into the orthologous group.

If a gene from the same strain has already been inserted back into the orthologous group, then a new orthologous group is formed with said gene acting as the representative and initial sole member. Then in subsequent passes the genes from the paralogous group are scored against all existing representatives, and inserted into the group that they fit best in based on synteny. This is continued until the pool of paralogous genes is emptied.

In the step where the most suitable group is chosen for a paralogous gene one could have as well scored the gene against all the members of a group rather than just the representative gene from said group. The reason why a candidate is chosen instead of scoring against the whole group is that a synteny score against smaller groups has a bigger chance of being high. Having fewer members reduces the chance of outlier genes bringing down the average synteny. This is especially a problem when considering that new candidate groups start off with just a single member, and the chances of a paralogous gene exhibiting good synteny with a single gene is higher than with two or three genes for an example.

Group	genes	score	paralogs	candidates	groups	orphans	orphan score
group5509	192	5.05	45	3	{169 6.17} {22 2.00}	SM159	2.43

Above is an example output from the disambiguation step. The group being disambiguated had 192 genes, an average synteny score of 5.05, and contained 45 paralogs. The disambiguation step resulted in 3 candidates which ultimately netted two groups and one orphaned gene. Orphaned genes are genes in gene groups with a single member. Gene groups need to have at least two members to be regarded as a group, so single member groups are discarded. In such a scenario the gene, in this case a gene from strain SM159, will not be part of any group after the disambiguation step and is therefore orphaned.

The two groups formed with several members are listed in the groups column. One group has 169 members, and an average synteny score of 6.17, while the other has 22 members, and a synteny score of 2.

One potential problem with the approach above is that if paralogs from different strains are removed they will sometimes form the basis of a new group even if their synteny score is zero. This was a deliberate choice of the algorithm and done in order to avoid throwing away any data that might be useful. In theory two paralogs with a zero synteny score are likely the result of separate gene duplication events taking place after speciation. The lack of synteny suggests that they are placed in completely different places in the host genomes, and as such may take on different roles.

Many other alternate approaches could have been taken to disambiguation which could be better generally, or better in specific scenarios at hand. Therefore S3K has the ability to generate scores for a user selected list of gene pairs. Using this capability one could use synteny information to develop alternate approaches, if they should be more suited to the task at hand.

1.4 Contig Identification

1.4.1 Introduction

S3K has the ability to guess and/or confirm the class (chromosomal / plasmid) of a contig in a dataset with a portion of contigs already classified by other means. The classification works via a two way consensus based on gene group and synteny information.

1.4.2 Gene group Consensus

This consensus is made by going through each gene on a contig. For each gene the gene group that it belongs to is further examined. The contig classification of every other member of that gene group is noted. A tally is made of each gene residing on a chromosomal contig as well as one for each gene on a plasmid contig, along with the plasmid type. Once the gene group of every member of every gene on said contig have been examined the proportion of total chromosomal or plasmid genes encountered compared to the total number genes encountered is calculated.

If the proportion of chromosomal or plasmid genes encountered is significantly higher than the other then this is taken as a sign that the fragment belongs to that same classification. This is a reasonable assumption to make if the contig has enough many genes on it. For a large enough contig the majority of the genes in the gene groups should be expected to reside on the same type of contig, be it chromosomal or plasmid.

1.4.3 Gene synteny consensus

This consensus relies on the same principle as the gene group method, except it uses synteny to examine the group of each gene on a contig is using synteny. The genes of each gene group present on a contig being classified is split up according to whether it is located on a chromosome or on a plasmid of a certain type. The gene from the contig being classified then has its synteny scored against each of these subgroups. If the gene being classified is from a chromosome it should have higher synteny with the genes from its gene group that are known to be located on a chromosome. The same applies to plamids of a certain type.

1.4.4 Final Classification

The final classification of a contig is calculated by taking the resulting parameters into consideration. In essence these parameters must be taken into consideration. Percentage of gene group genes located on a chromosome, percentage located on plasmids, synteny shared with chromosomal genes, and synteny shared with plasmid genes.

Let us say that the parameters for a contig being classified are such:

Percentage chromosome: 0.6 (60%)
Percentage Plasmid: 0.1 (10%)

Synteny chromosome: 0.4 (40%)
Synteny Plasmid: 0.04 (4%)

If “Percentage Plasmid” is subtracted from “Percentage chromosome” then the sign of the result will be positive if there is a lean towards chromosome, and negative if a lean towards plasmid.

The absolute value of the result is then a measure of the magnitude of the tilt, and the same can be done for synteny scores which are here divided by the neighbourhood size to change their range into a number between 0 and 1.

In the above example both percentage and synteny parameters have a chromosomal lean, which is a good indicator that said contig is of chromosomal origin. A way to determine the confidence of the classification using these parameters is by adding the two parameters together. If both have the same sign then they will reinforce the absolute value, drawing it closer to a perfect score of 2.0, but if they have opposite signs they will bring the absolute value closer to zero which is to be considered more ambiguous.

The viability of this approach was tested on a dataset which had many contigs pre-classified using a combination of repA, repB and repC based classification combined with reference genome cross checking.

The classification of these contigs were confirmed by being in agreement with classification assigned using S3K, while many unclassified contigs which the previous method had failed to classify were also classified, often with strong leans.

The algorithm that calculates the final classification was tweaked for the dataset it was developed for. As such it might not work optimally for other datasets. For this reason it is recommended that users develop and tweak their own classifications by using the gene group and synteny consensus scores calculated by S3K. These scores can be found in the “Chromosomal Percentage”, ”Plasmid Percentage” “Chromosomal Synteny”, and “Plasmid Synteny” columns of the contig classification output file from S3K. The Individual plasmid type scores are found in the metadata column as exampled below.

Rh1|143|9|0.358612 Rh5|21|8|0.188346 Rh7|26|7|0.178441

Each plasmid type encountered are present in the metadata in the above example three are present, Rh1, Rh5 and Rh7. The metadata concerning each plasmid follows the plamid ID separated by the ‘|’ character. The first parameter is the number of times the given plasmid was encountered, which in case of Rh1 is 143, followed by the average synteny which is 9 in case of Rh1, and finished off with the overall plasmid score, which in case of Rh1 is 0.35.

It is important to note that the overall and confidence scores for classifying plasmid types are calculated differently as tweaked for the example dataset.

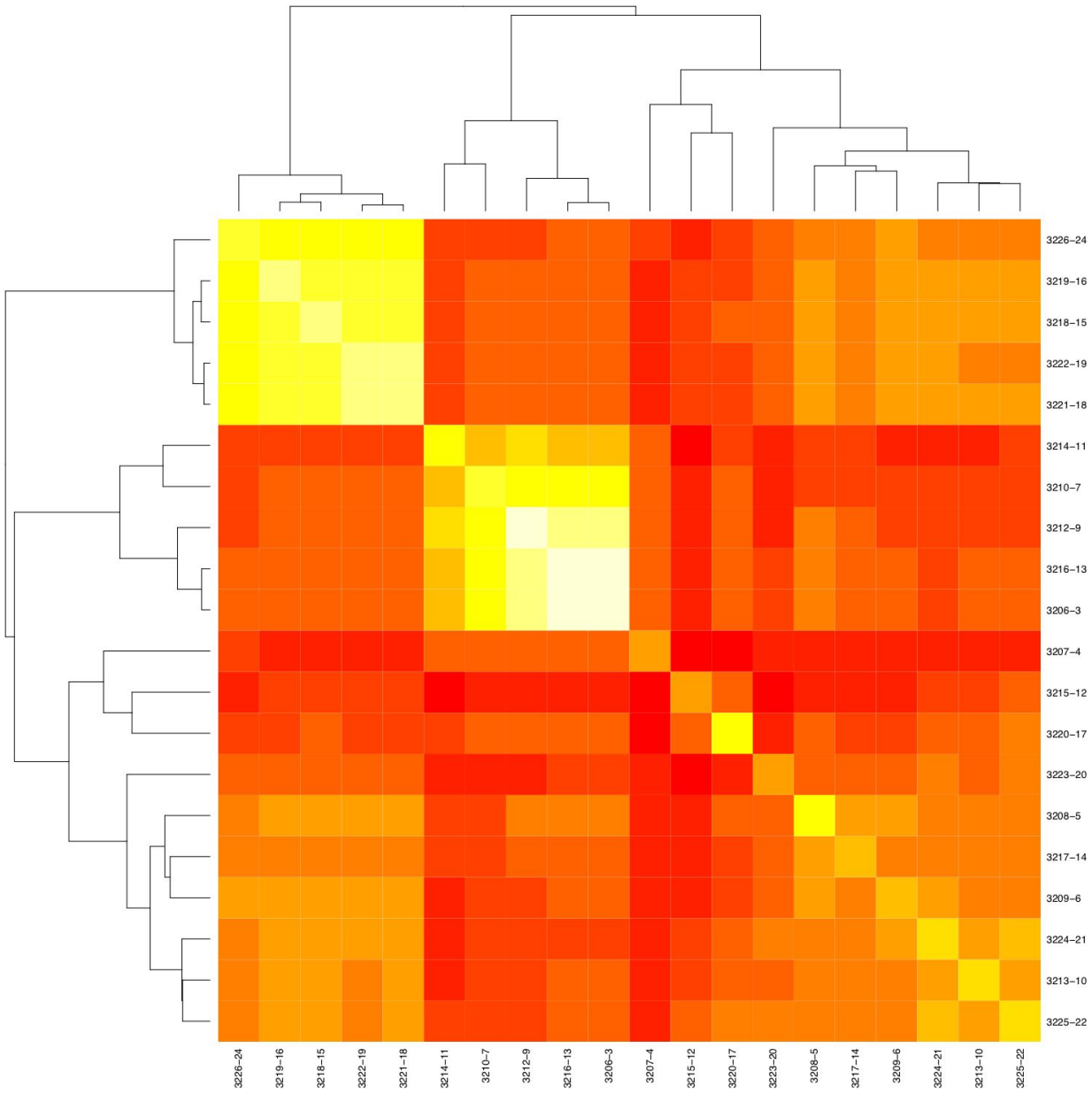
1.5 Gene Relation Matrix

The gene relation matrix is calculated using a presence/absence matrix (M). The rows of the matrix represent all existing orthologous gene groups, and the columns represent each existing strain in the dataset. In the cells a 0 will represent that the given strain does not have any genes in that gene group, and a 1 means that it does.
The dot product of this matrix with its transpose (MT) will net a matrix whose cells tally the number of gene groups they have in common (PGRM).

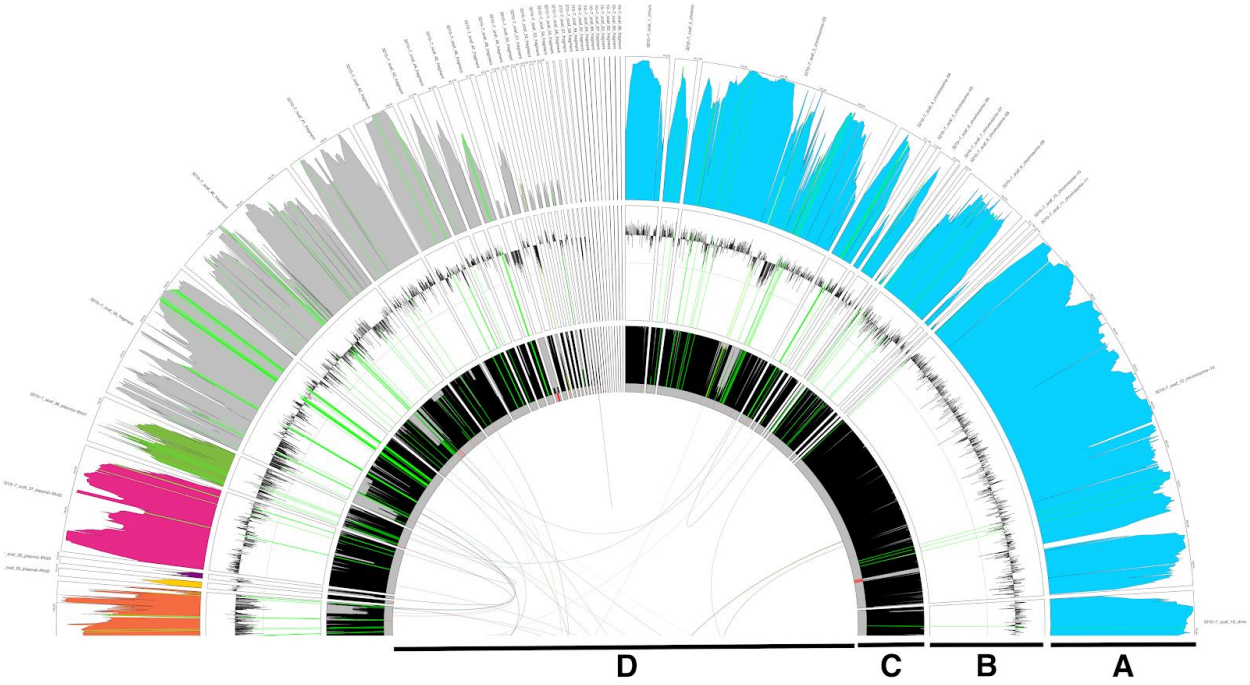
The diagonal of this matrix tallies the number of gene groups each strain shares with itself, or the number of gene groups present in the given strain. In order to normalise the matrix the average of the diagonal is calculated and each cell in the matrix is divided by this average.

$PGRM = M \cdot M^T \cdot mean(diagonal(M \cdot M^T))$

The resulting matrix will have numbers close to 1 for diagonals if the number of gene groups in each strain is close to each other, and the numbers in the other cells will be a measure of how similar two strains are based on the number of gene groups they share. This will serve as a measure of their relatedness based on gene group presence/absence and can be visualized using a heatmap generated by the GenerateHeatmap.R script accompanying S3K. Example seen below.


3226–24
3219–16
3218–15
3222–19
3221–18
3214–11
3210–7
3212–9
3216–13
3206–3
3207–4
3215–12
3220–17
3223–20
3208–5
3217–14
3209–6
3224–21
3213–10
3225–22

1.6 Strain Charts



S3K can generate data which accompanying R scripts need in order to visualise certain genomic features in strains. These charts consist of 4 circular lanes that each visualize different data about a given contig of the strain in question. Above is the top half of a strain which is formatted to fully utilize all features of S3K strain visualisation. Each segment of a circular lane represents one contig whose identifier is printed on the outside of the chart. Below is an explanation of the data within each of the four lanes as labelled in the above cross section.

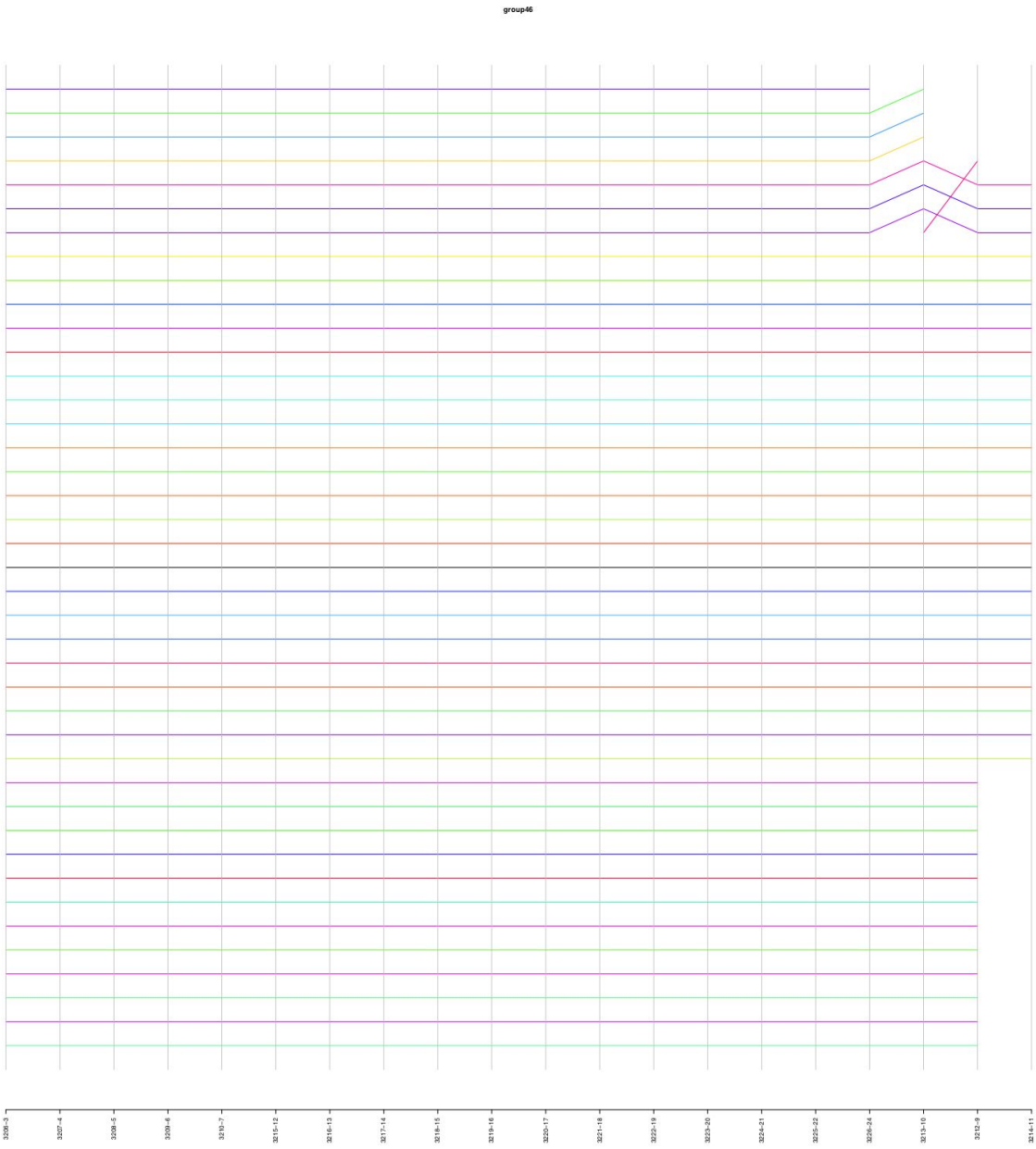
A: Synteny score of the given gene against other genes of its group visualized as a bar chart raising from inside towards the outside of the circular lane. The higher the bar the higher the synteny score of the given gene against its gene group. Thus areas with high bars denote areas where synteny is well preserved in given strain, compared to all other strains in the dataset. Bar chart are inside circular segments, each of which is a contig which may be coloured according to its type if the input dataset is properly formatted, and S3K properly configured to recognize them. Specific gene groups can be highlighted in custom colours by properly configuring S3K.

B: A chart of how much the GC3s content of a gene rises above or below the average for the strain. The faint grey circle running across the middle of this lane marks a GC3s content of 50%. As with lane A specific gene groups can are highlighted here if S3K has been properly configured to do so.

C: Bar chart depicting the abundance of each gene in the population. The height of the bar depends on the percentage of strains that possess that same gene. If the gene is unique to this strain then the bar flips down into the grey region and is coloured red to make it noticeable.

D: A chart of paralog pairs where each curved coloured line connects a paralog pair.

1.7 Synteny Charts



S3K can generate data which accompanying R scripts need in order to visualise the synteny of each gene group in the dataset.

Above is an example of such synteny in the genetic neighbourhood of a gene group being visualised. Each vertical line represents 41 neighbourhood genes in the strain specified by the identifier on the lower axis. Horizontal lines with the same colour trace the placement of the same gene in the neighbourhood in different strains. The horizontal black line in the middle represents the gene from the gene group whose neighbourhood is being visualised. Each of the 20 lines above and below the black line represents the 40 genes placed up and downstream of it. Each colour represents one gene, so whenever a line shifts position the placement of that gene in said strain is different. This could have been caused by deletion or insertions events depending on whether the line shift closer or away from the middle. When a certain coloured line is only depicted in some strains that means the gene it depicts is only found in those strains. In the above example strain 3213-10 has had a gene inserted 14 genes upstream of the group gene being visualised. Note that colours are generated by random and so two colours representing different gene groups may look similar to the human eye.

1.8 GC3s Content

GC3s content was calculated for the nucleotide sequence of each gene by calculating the average portion of nucleotides in the third position of a codon that were either guanine (G) or cytosine (C). Only codons that offer variability in the third codon were used in this calculation so the following codons were excluded: ‘TAA’, ‘TAG’, ‘TGA’, ‘TGG’ and ‘ATG’.

The translation of codons to amino acids uses a universal encoding shared across all organisms, but each species can differ in their preference for alternate codons. Therefore the GC content of introgressed DNA fragments originating from other species may help set them apart from surrounding host genes. The advantage of calculating GC3s content rather than the traditional GC content is that by focusing on just the nucleotides in codons coding for the same amino acid, but which can vary, a stronger signal will be produced.

1.9 Declustering

Although all the genes in large gene groups have been placed in the same group due to them exhibiting sequence similarity they might nevertheless not fit in together according to synteny. In such instances the declustering functionality of S3K will attempt to split groups into smaller syntenic subgroups, if such exists. The algorithm for doing this works by removing all genes from a group, then attempt to put lump them back together based on synteny. First the gene with the highest average synteny score with all other genes is selected, and made the first member of a new gene group. Then each remaining gene that has a synteny score above a given threshold (0 by default), is made member of the same group.
If no genes remain then the new group is the same as the old group and there was no syntenic subgroups present. If some genes do not score above the threshold they will be made part of new subgroups based on synteny. Any remaining lone genes that have a synteny score below the threshold with every other gene in the group will be removed entirely.
Any subgroups formed by the declustering process will have a “_split” suffix added to their group ID, which is followed by the given number of said subgroup.

Declustering is a very slow process and cannot be toggled at the same time as disambiguation, so it is necessary that each process be run in separate sessions. If both are desired the resulting gene groups file output from one process can be used as input into the second process. Due to the disruptive nature of each process on groups the order of doing each will matter.

2 Usage

2.1 Sample Dataset

For a demonstration of the proper use of Syntenizer3000 a sample dataset consisting of the approximately 200 bacterial strains that the tool was developed to analyze has been made available for download:
data

2.2 How to build

Run build.sh in the folder with the source code to build an executable on a UNIX derived OS. GCC with at least C++11 support or equivalent must be installed. An executable binary names ‘Syntenizer’ will be built if all goes well.

2.3 How to run

Run without any input parameters to get a list of possible arguments and a short explanation of what they do.

Below is an example of running the program on the sample dataset (linked in 2.1) with a full suite of analysis enabled:

./Syntenizer --gffdir=/data/input --outdir=/data/output --genegroups=/data/input/gene_groups.csv --contigcolours=/data/input/ContigColours.csv --syntenizegroups --generatestraincharts --generatesyntenycharts --chromosomeidentifier=chromosome --plasmididentifier=Rh --fragmentidentifier=fragment --genegrouphighlightcolours=/data/input/GeneGroupHighlightColours.csv --ffndir=/data/input --generatefnagroups --classifyfragments --disambiguategroups

In the above example all input files, including *.gff, *.fnn, gene groups and others are located in the folder '/data/input', and all files generated by the tool are output into the folder '/data/output'.

If the program has been configured to output circular strain chart data or gene group synteny map data then the next step is to render these using the provided R scripts 'GenerateCharts.R', 'GenerateSyntenyMaps.R' and 'GenerateHeatmap.R'. These scripts need to be manually run by users with a single input parameter, which is the path to the folder where the chart or map data generated by Synternizer has been output, which in the above example would be '/data/output'. So for the above example the following commands will render all possible charts:

```
Rscript GenerateCharts.R "/data/output"
Rscript GenerateSyntenyMaps.R "/data/output"
Rscript GenerateHeatmap.R "/data/output"
```

2.4 Things to note

If issues or uncertainties crop up while trying to use the program you can refer to the example dataset provided. A full suit of analysis can be performed using said dataset so it should serve as a guide on how input data should be formatted for the program to work.

Only GFF3 files generated by PROKKA have been tested. Other versions or files generated by other tools may contain variations that might cause incompatibility issues. In that case contact the author for possible solutions or fixes.

Two gene group file formats are supported. The direct gene group file output from ProteinOrtho, plus an internal format detailed below.

Each gene group in the internal format is defined on a space (ASCII 0x20) separated line starting with The group id as the first item. All subsequent items should be a concatenation of the strain id and gene id with the character '|' (ASCII 0x7C) separating them. See gene_groups.csv in the example dataset for an example.

To generate basic circular charts the program only needs properly formatted GFF3 files and accompanying gene group file. However, the GC3s lane requires the presence of gene sequence data. If these are part of the GFF3 files then they will be parsed automatically. Alternatively they can be provided using gene group *.fna files using the input parameter.

In order for contigs to be coloured according to their type (chromosome/plasmid/fragment) then the contig ID's in the GFF3 files have to contain an string pattern that identifies them as such. The program can then be configured to recognize these using the appropriate options as detailed below. For examples of this working correctly use the example input parameters provided above on the example dataset.

When Syntenizer is run with a bare minimum of input parameters, which are the *.GFF file input folder, output folder and gene group file three files will be output:

gene_groups.csv, which contains a Syntenizer formatted version of the the input gene groups. If the gene group input file was a Syntenizer formatted file then this output file should be identical to the input file in terms of contents. Since gene group highlighting on charts requires referencing to gene group ID's, and since ProteinOrto formatted gene groups lack these it is necessary to run the ProteinOrtho gene groups through Syntenizer and use the obtained

Syntenizer formatted gene group output file as a basis for creating charts if gene groups need to be highlighted.

presence_absence_matrix.csv: A file that contains a presence/absence matrix based on gene groups. The ';' (ASCII 0x3B) delimited file represents which strains are present "1" or absent "0" in each gene group.

presence_absence_gapit.csv: A GAPIT formatted presence/absence matrix. This file can be directly loaded and passed to GAPIT for use in GWAS analysis of the given bacterial strains.

gene_relation_matrix.csv: This file contains a gene relation matrix based on presence/absence of strains in gene groups. This file is needed for rendering a gene relation heatmap using the GenerateHeatmap.R script. Consult the manual for information on how the matrix is calculated.

strain_metrics.csv: This file contains useful statistics on the input dataset. Below is a short explanation of each column.

Strain ID - Name of the given strain. Contigs - The number of contigs in the given strain. bp - The number of determined basepairs in the given strain. ubp - The number of undetermined basepairs in the given strain. GC - The GC-content ratio [0:1] of the given strain. CDS - The number of coding sequences in the given strain. Note that CDS = Orthologs+Paralogs+Uniques. Orthologs - The number of orthologous genes in the given strain. Paralogs - The number of paralogous genes in the given strain Uniques - The number of unique genes in the given strain. tRNA - The number of tRNA in the given strain, if present in the GFF file. rRNA - The number of rRNA in the given strain, if present in the GFF file. tmRNA - The number if tmRNA in the given strain, if present in the GFF file.

0-10k - The percentage of total strain basepairs located on contigs between 0 to 10kbp in size.
10-250k - The percentage of total strain basepairs located on contigs between 10 to 250kbp in size.
250k-500k - The percentage of total strain basepairs located on contigs between 250 to 500kbp in size.
500k-750k - The percentage of total strain basepairs located on contigs between 500 to 750kbp in size.
750k-1000k - The percentage of total strain basepairs located on contigs between 750 to 1mbp in size.
1000-5000k - The percentage of total strain basepairs located on contigs between 1 to 5mbp in size.
5000k - The percentage of total strain basepairs located on contigs larger than 5mbp in size.

2.4 Input options

Below is a list and explanation of all input parameter options offered by Syntenizer.

--gffdir=dir Sets the directory where strain gff files are located. All files with the .gff extension will be loaded. The strain ID is set to the name of the file (with the the extension removed) i.e. the file calb.gff defines the genome of the calb strain. If contig size and contig sequence data are present in the gff file then these will be parsed. If they are not present the size of each contig is set to the end of the final coding sequence gene defined on said contig. Alternatively sequence data for each gene can be provided using .ffn files. Consult the demo dataset for examples of file formatting. The function of a gene defined using the 'product=' tag will be parsed if it is present. Consult the demo dataset as for examples.

--outdir=dir Sets the output directory for all files generated by Syntenizer3000. This directory must exist beforehand, and any files already present with similar name and extension as output files will be overwritten.

--genegroups=file Set the file specifying gene groups. Syntenizer3000 can parse two different ways of encoding this information. The first is the direct output file format from the orthology tool ProteinOrtho. The second is an internal format, i.e. as below:

Group1: strain1|gene1 strain2|gene1

Each tab delimited line starts with a group ID finished with a colon. Then a list of genes in said group each consisting of the ID of the gene strain and ID of the gene in said strain, separated by the | ASCII character (0xC7).

--ffndir=dir Sets the directory where gene sequence .ffn files are located. All files with the ffn extension are parsed, and the gene sequences found within are attached to their corresponding genes. Sequences for unknown genes are ignored. Sequences that may already have been parsed from .gff will be compared to the ones parsed from .ffn files and result in an error if not an exact match.

--contigcolours=file Sets the file specifying the contig colours in strain charts. Consult the file colours.csv from the demo dataset for an example of how the files is to be formatted. The first tab delimited line is a header defining the role of each subsequent tab delimited line. Each line contains two items, the first is a string pattern, and the second is a valid R colour i.e red, blue, #C0C0C0 etc. The ID of each contig being rendered is searched for the defined string patterns and if there is a match the colour associated with the pattern will be applied to the contig. If the contig ID matches several of the defined string patterns then the colour of the string pattern nearest the end of the contig colour file will be the one used.

--syntenizegroups Generate synteny score for each gene group into the file groups_synteny.csv, in addition to other useful information about said gene group. Below is an explanation of each column in the ; delimited file: Group - ID of the group Connectivity - The connectivity of the group as parsed from the ProteinOrtho file. Genes - The number of genes in

group. Strains - The number of unique strains of the genes in group. Orthologs - The number of orthologous genes in group. Paralogs - The number of paralogrous genes in the group Chromosome Genes - The number of genes residing on chromosomal contigs in the group. This stat only works if a valid chromosome identifier. Consult the demo dataset for an example of this functioning correctly. Plasmid Genes - The number of genes residing on plasmid contigs in the group. This stat only works if a valid plasmid identifier. Consult the demo dataset for an example of this functioning correctly. Fragment Genes - The number of genes residing on fragment contigs in the group. This stat only works if a valid fragment identifier. Consult the demo dataset for an example of this functioning correctly. Protein Products - All the protein products of the genes in the group, separated by the / character. The protein product is parsed from .gff files using the 'product=' tag if present. In an ideal world each gene group should only contain one protein product, and differing protein products could be a sign of a badly constructed group. Consult the demo dataset for an example of this functioning correctly. Synteny Score SIMPLE - A number between 0 and 40 representing the average SIMPLE synteny score of the group. Consult the manual for details on how this is calculated. Synteny Score SOPHISTICATED - A number between 0 and 40 representing the SOPHISTICATED score of the group.

--disambiguategroups Generates file disambiguated_gene_groups.csv with gene groups devoid of paralogs.

--generatefnagroups Generate gene group .fna files which define the sequence of each member of the gene group.

--generatestraincharts Generates the data files needed for rendering strain charts using the GenerateCharts.R script.

--generatesyntenycharts Generate the data files needed for rendering group synteny charts using the GenerateSyntenyMaps.R script.

--chromosomeidentifier=pattern Sets the character pattern that identifies a chromosomal contig if it is found to be part of its ID. This parameter is needed for contig classification to work. Consult the demo dataset for an example of how this could work.

--plasmididentifier=pattern Sets the character pattern that identifies a plasmid contig if it is found to be part of its ID. This parameter is needed for contig classification to work. Consult the demo dataset for an example of how this could work.

--fragmentidentifier=pattern Sets the character pattern that identifies a fragment contig if it is found to be part of its ID. Two number digit characters are assumed to exist right after this pattern in any matched contig ID, which identify the plasmid type, i.e. Rh01. This parameter is needed for contig classification to work. Consult the demo dataset for an example of how this could work.

--syntenizegenepairs=file Sets the file specifying specific gene pairs to be syntenized into the gene_pairs_synteny.csv output file. Consult the file GenePairs.csv in the demo dataset for an example of formatting.

--genegrouphighlightcolours=file Sets the file specifying highlight colours for gene group in rendered strain charts. Each ; delimited line defines two items, the first being a gene group ID, and the second a valid R colour said gene group should be highlighted with on rendered strain charts. Consult the file GeneGroupHighlightColours.csv in the demo dataset for an example of formatting.

--classifyfragments Generate file classified_fragments.csv with fragment classifications based on synteny and gene group information. Below is an explanation of each column in the ; delimited file: Strain - The ID of the strain this contig belongs to. Contig - The ID of the contig. Genes - The number of coding genes on this contig. Chromosome - A number between 0 and 1 describing the fraction of chromosomal genes in the gene groups of the genes on this contig. Plasmid - A number between 0 and 1 describing the fraction of plasmid genes in the gene groups of the genes on this contig. Chromosomal Synteny - A number between 0 and 40 describing the average synteny between chromosomal genes in the gene groups of each gene on this contig. Plasmid Synteny - A number between 0 and 40 describing the average synteny between plasmid genes in the gene groups of each gene on this contig. Coverage - A number between -1 and 1 calculated as "chromosome - plasmid". Consult the manual for an explanation of this parameter. Syneny - A number between -1 and 1 calculated by "(Chromosomal Synteny / 40) - (Plasmid Synteny / 40)". Consult the manual for an explanation of this parameter. Magnitude - A number between -2 and 2 calculated as +/- Coverage + Synteny. Consult the manual for an explanation of this parameter. Confidence - The confidence attached to the contig classification. This can be IDEAL / HIGH / MEDIUM / LOW / NA .
Classification - The classification of this contig, which can either be Chromosome, the plasmid identifier followed by two digit type or Ambiguous. Consult the manual for an explanation of this parameter. Metadata - Extra information on regarding the plasmid classification calculations.

--declustergroups / --declustergroups=threshold generate file declustered_gene_groups.csv with groups split into syntenic subgroups based on provided threshold or the default synteny threshold of 0.