1 Pangenome Construction using Roary

1.1 Introduction

Given a set of genomes, the pan genome is the collection of all genes the set contains. Roary, the pan genome pipeline, takes closely related annotated genomes in GFF3 file format and calculates the pan genome.

For more in depht information about Roary, please feel free to have a look the paper:

> **Roary: Rapid large-scale prokaryote pan genome analysis**
> Andrew J. Page, Carla A. Cummins, Martin Hunt, Vanessa K. Wong, Sandra Reuter, Matthew T. G. Holden, Maria Fookes, Daniel Falush, Jacqueline A. Keane, Julian Parkhill
> *Bioinformatics, 2015;31(22):3691-3693 doi:10.1093/bioinformatics/btv421*

or visit the Roary manual.

1.2 Learning outcomes

By the end of this tutorial you can expect to be able to:

- Describe what a pangenome is
- Prepare data for input to Roary
- Perform QC on input data and understand why QC is important
- Run Roary to create a pangenome with and without a core alignment
- Understand the different output files produced by Roary
- Draw a basic tree from the core gene alignment produced by Roary
- Query the pangenome results produced by Roary
- Use Phandango to visualise the results produced by Roary

1.3 Tutorial sections

This tutorial comprises the following sections:
1. What is a pan genome
2. Preparing the input data
3. Performing QC on your data
4. Running Roary
5. Exploring the results
6. Visualising the results with Phandango

1.4 Authors

This tutorial was created by Sara Sjunnebo.

1.5 Running the commands from this tutorial

You can run the commands in this tutorial either directly from the Jupyter notebook (if using Jupyter), or by typing the commands in your terminal window.

1.5.1 Running commands on Jupyter

If you are using Jupyter, command cells (like the one below) can be run by selecting the cell and clicking *Cell -> Run* from the menu above or using *ctrl Enter* to run the command. Let's give this a try by printing our working directory using the *pwd* command and listing the files within it. Run the commands in the two cells below.

```
pwd
```

```
ls -l
```

1.5.2 Running commands in the terminal

You can also follow this tutorial by typing all the commands you see into a terminal window. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

To get started, select the cell below with the mouse and then either press control and enter or choose Cell -> Run in the menu at the top of the page.

```
echo cd $PWD
```

Now open a new terminal on your computer and type the command that was output by the previous cell followed by the enter key. The command will look similar to this:

```
cd /home/manager/pathogen-informatics-training/Notebooks/ROARY/
```

Now you can follow the instructions in the tutorial from here.

1.6 Let's get started!

This tutorial assumes that you have Roary and Prokka installed on your computer. For download and installation instructions, please see:

- The Roary GitHub-page
- The Prokka GitHub-page

To check that you have installed Roary correctly, you can run the following command:

```
roary -help
```

This should return the help message for Roary.

Similarly, to check that you have installed Prokka correctly, you can run:

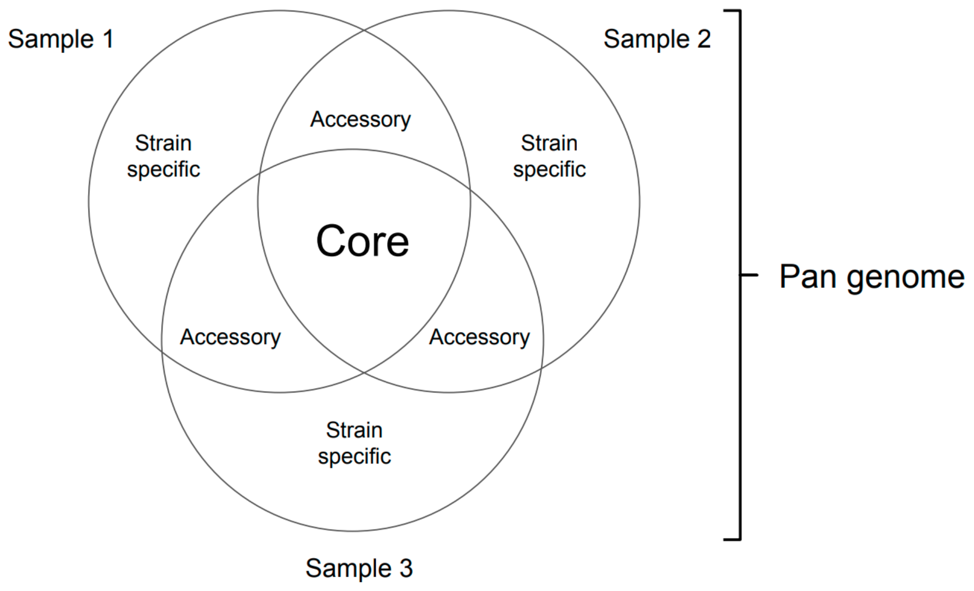
```
prokka -help
```

This should return the help message for Prokka.

To get started with the tutorial, head to the first section: What is a pan genome
The answers to all questions in the tutorial can be found here.

2 The Pan Genome

The pan genome for a prokaryote population is the complete set of genes that it contains. This includes genes present in all of the genomes, and genes that are only present in some, or even only in one of the genomes. The subset of genes present in all of the genomes is called the **core genome**. These are often highly conserved genes with important functions, for instance house-keeping genes. The subset of genes that are not present in all, but in two or more of the genomes, is called the **accessory genome**. The accessory genome often contains genes that have been transferred between baterial strains, for example genes linked to virulence or drug resistance. Genes present in only one of the genomes can be referred to as **strain specific**.



Pan genome

As you can imagine, the pan, core and acessory genomes can provide important insight into the genetic structure of prokaryotic genomes. By analysing the pan genome we can gain a better understanding of key processes like evolution and selection. Roay is a software tool that allows you to calculate the pan genome from annotated bacterial genomes. It is fast and accurate and can conveniently be run on most modern PCs. In this tutorial we are going to guide you through a complete pan genome analysis, starting with annotation of the genomes, working through running the pan genome pipelin, and finally visualising the results.

2.1 Check your understanding

Q1: The pan genome contains:
a) Only genes present in one genome in a population
b) All genes from all genomes in a population
c) Only genes present in all genomes in a population

Q2: Core genes are:
a) Often important for basic cell functions
b) Present in only a subset of the genomes of a population
c) Often related to drug resistance

The answers to these questions can be found here.

Now that you know a bit more about pangenomes, let's head over to the next session: Preparing input data
You can also head back to the index page.

3 Preparing the data

In this tutorial we have included three assemblies of *Streptococcus pneumoniae*. The assemblies are available for download from the ENA and the accession numbers are included below. If you have access to the the clustre at the Wellcome Sanger Institute the lane ids are also listed below.

Name	Accession	Sanger Lane ID
sample1	GCA_900194945.1	13681_1#18
sample2	GCA_900194155.1	13682_2#34
sample3	GCA_900194195.1	13682_2#39

If you are using the cluster at the Wellcome Sanger Institute and want to create a symlink to one of the samples in your working directory, you can use the command below. However, note that this is not neccessary for the sake of this tutorial.

```
pf assembly -t lane -i 13681_1#18 -l .
```

3.1 Roary input files

Roary takes annotated assemblies in GFF3 format as input. The files must include the nucleotide sequence at the end of the file, and to make it easier for you to identify where genes came from, each input file should have a unique locus tag for the gene IDs.

All GFF3 files created by Prokka are valid with Roary and this is therefor the recommended way of generating the input files. We are now going to look closer at how you can use Prokka to annotate your genomes.

3.2 Annotation

Prokka is a tool that performs whole genome annotation. It is easy to install and use and as mentioned the GFF files that it outputs are compatible with Roary.

Our three assembled *S. pneumoniae* genomes are located in a directory called “assemblies”.

```
ls assemblies
```

To run Prokka on a single file using the default settings, you can use the following command:

```
prokka sample1.fasta
```

If you have a lot of assemblies that you want to analyse, running this for each sample will soon become tedious. Instead, we will use a for-loop to run Prokka on all the fasta files in the assemblies directory. We will also use the following options for Prokka:

Option	Description
–locustag	Specifying a locus tag prefix
–outdir	Specifying a directory to put the output in
–prefix	Specifying a prefix for the output files

By specifying a unique locus tag we make it easier to identify which sample different genes came from when we look at the results from Roary. The outdir and prefix options will make it easier for us to keep track of our files.

```
for F in assemblies/*.fasta; do FILE=${F##*/}; PREFIX=${FILE/.fasta/}
           prokka -locustag $PREFIX -outdir annotated_$PREFIX \
           -prefix $PREFIX $F; done
```

This is going to take around 5 minutes to run, so be patient.

Once this has finished, you should have three new directories called annotated_sample1, annotated_sample2 and annotated_sample3. Have a look to see that it worked:

```
ls -l
```

```
ls -l annotated_sample1
```

As you can see, for sample1 we now have a number of annotation files. There is more information about the different output files, along with information about other usage options, on the Prokka GitHub page. For now, we are only interrested in the GFF files that were generated as this is what we are going to use as input for Roary.

Note: If you are working on the Sanger Institute cluster, Prokka is automatically run as part of the annotation pipeline. To create a symlink to the GFF file, you can use the command below (though this is not neccessary for the sake of this tutorial):

```
pf annotation -t lane -i 13681_1#18 -l .
```

Also for Sanger users, to run Prokka independently of the automated pipeline, you can use the script called *annotate_bacteria*. Run the below command for more information:

```
annotate_bacteria -h
```

3.3 Check your understanding

Q3: Why do we need to run Prokka?
a) It will perform QC on our data
b) It will annotate our data
c) We don't, Roary can handle fasta files as input

Q4: Why do we use the –locustag option when we run Prokka?
a) To make it easier to keep track of the output files
b) Because Roary won't work without it
c) To make the Roary results easier to interpret

The answers to these questions can be found here.

Now continue to the next section of the tutorial: Performing QC on your data.
You can also revisit the previous section or return to the index page

4 Performing QC on your data

The results you can get from any analysis will only ever be as good as the data you put into it. To avoid spending countless hours performing analysis without receiving any satisfactory results, or worse yet erronious or misleading results, it is important to QC your data before starting. There are a number of checks you can make to ensure your dealing with high quality data, and we will walk you through some of them here.

4.1 Contamination

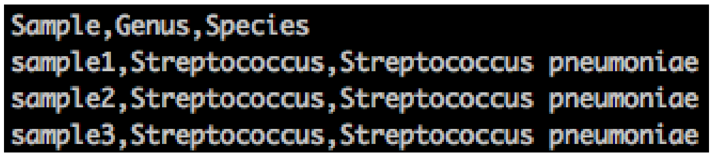
In order to get meaningful results from Roary, the samples should be closely related. If you have lots of contamination in your data, for instance if one of your samples is from a different species, you will get very few genes in your core genome, if any at all.

It is always a good idea to check that your samples are the species you expect them to be. You can use tools such as Kraken for this. Roary comes with a qc option that will run Kraken for you and generate a report listing the top species of all the samples. For this to work you need to have Kraken installed and a Kraken database available. You won't be needing it for the sake of this tutorial but it is highly reccommended if you plan to do any real analysis.

The following command can be used to generate a qc report with Kraken (substituting the path to the database to wherever you downloaded it):

```
roary -qc -k /path/to/kraken/db *.gff
```

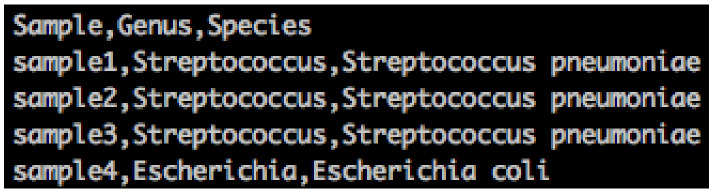
The report will look something like this:



```
Sample,Genus,Species
sample1,Streptococcus,Streptococcus pneumoniae
sample2,Streptococcus,Streptococcus pneumoniae
sample3,Streptococcus,Streptococcus pneumoniae
```

QC report

As we expected, these three samples are all of the same species. Let's assume that we initially had a forth sample that we wanted to use in this analysis. We thought that this sample was also from *S. pneumoniae*, but once we run roary with the qc option, we get the following output:



```
Sample,Genus,Species
sample1,Streptococcus,Streptococcus pneumoniae
sample2,Streptococcus,Streptococcus pneumoniae
sample3,Streptococcus,Streptococcus pneumoniae
sample4,Escherichia,Escherichia coli
```

QC report with contamination

This tells us that the most prevalent species in sample 4 is in fact *Escherichia coli* so we will exclude this sample from our analysis before we carry on.

For Sanger users Kraken is already installed and is run as part of the automated QC pipeline. To create a symlink to the Kraken report, you can do:

```
pf qc -t lane -i 13681_1#18 -l .
```

The size of the assemblies can also provide a useful hint. If one of the assemblies is much smaller or bigger than the others there is a chance that this is not of the same species as the rest.

4.2 Coverage

To get decent assemblies out of your raw data, you need a genome coverage of at least 30x. For a quick estimate of your coverage, you can divide the number of bases in your raw data with the number of bases in the reference genome of the species. For the samples used in this tutorial, the coverage is listed below. The genome of *S. pneumoniae* is approximately 2,200,000 bases.

Sample	Nr of Bases	Coverage
sample1	262705400	120x
sample2	218026200	99x
sample3	173524000	79x

4.3 Fragmented assemblies

If the assemblies are very fragmented (thousands of contigs), the genes may be too fragmented to be of much use.

These are just some of the most basic things that you can do to make sure your data looks alright. There is much more that can be done but we won't go into any further detail in this tutorial.

4.4 Check your understanding

Q5: Why is it important to QC your data?

Q6: You're not getting any core genes when you run Roary. What could be the reason?

The answers to these questions can be found here.

Now you should be ready to run Roary to generate a pangenome, so head to the next section, Running Roary.
You can also revisit the previous section or return to the index page.

5 Running Roary

At this stage you should have three GFF files generated by Prokka, each in its own directory. Provided your QC looked alright, you are now ready to run Roary to generate the pan genome.

We are going to run Roary twice, first with the default settings, and then using MAFFT to generate a core gene alignment. For both of these runs we will want all the annotation files in the same directory, so lets take a copy of them to our current directory:

```
cp annotated_sample*/*.gff .
```

5.1 Run Roary with default settings

Running Roary with the default settings is very straightforward. All you need to do is to run `roary *.gff` and it will create a pan genome using all GFF files in the current directory. We want to run Roary twice with different settings, so in order to keep track of our output files from each run we will also specify an output directory where Roary should put the results. Give the following command a go:

```
roary -f output_no_alignment *.gff
```

This will run for a minute or two.

We will have a closer look at the results in the next section, so for now let us just see that there are some output files in the directroy we asked Roary to create:

```
ls -l output_no_alignment
```

5.2 Run Roary with MAFFT

To be able to create pretty trees and cool visualisations, we want to genereate a multi-FASTA alignment of the core genes. To do this, we will now run Roary again, but this time with some more options.

Option	Description
-e	Create a multiFASTA alignment of the core genes
–mafft	Use with -e to use MAFFT instead of PRANK
-p	Number of threads to use

By default, Roary will use PRANK when the -e option is speified. It is accurate but slow. MAFFT is less accurate but very fast so we are going to use this instead by specifying the –mafft option. To further speed things up, we are going to use 8 threads (the -p option). For all usage options, you can have a look at the Roary website.

```
roary -f output_with_alignment -e -mafft -p 8 *.gff
```

This will take a bit longer to run than the previous command, about 5 minutes. Once finished you should have a directory called output_with_alignment containing the output files, this time including a core_gene_alignment.aln file. Just quickly check that this is the case and then we will head over to the next section: Exploring the results.

```
ls -l output_with_alignment
```

5.3 Check your understanding

Q7: Why do we want to run Roary with MAFFT?
a) Because it's quicker than to run Roary without the -e option
b) To get more accurate results
c) To generate a core gene alignment

Q8: Why do we use the -p otion?
a) We have to when we use MAFFT
b) To speed up the run
c) To get a pretty tree

The answers to these questions can be found here.

You can also revisit the previous section, or go back to the index page.

6 Exploring the results

6.1 Output files

Let's have a look at the results. We will focus on the output from the second run as this will be the same as the first run but will also include the core gene alignment produced by MAFFT. We will start by looking at the most important output files and after this we will look at how you can query your pan genome and draw a simple tree form the core gene alignment.

6.1.1 summary_statistics.txt

The summary_statistics.txt file contains a summary of the number of genes in the pan, core and accessory genomes. It provides an overview of the genes and how frequently they occur in the input isolates. Usually, you can expect the total number of genes in this file to be about 1,000 genes per million bases of your species reference genome. In this case, the genomes are around 2 million bases, so we would expect a total number of genes to be in the order of 2,000. Let's have a look and see if this is the case.

```
cat output_with_alignment/summary_statistics.txt
```

As you can see, we have around 2,500 genes which seems about right. If you get a lot fewer or many more genes than expected this could be an indication of an issue with your input data, for example contamination.

6.1.2 gene_presence_absence

The gene_presence_absence files lists each gene and which samples it is present in. The .csv file can be opened in Excel.

Let's have a look at the first ten lines of the file:

```
head output_with_alignment/gene_presence_absence.csv
```

The columns are tab separated and contains the following information:

1. The gene name (the most frequently occurring gene name from the sequences in the cluster)
2. A non unique gene name
3. Functional annotation (the most frequently occurring functional annotation from the cluster)
4. Number of isolates in the cluster
5. Number of sequences in the cluster
6. Average number of sequences per isolate (normally 1)
7. Genome fragment
8. Order within fragment
9. Accessory fragment
10. Accessory order within fragment
11. Comments on the quality of the cluster

12. Minimum sequence length in nucleotides of the cluster
13. Maximum sequence length in nucleotides of the cluster
14. Average sequence length in nucleotides of the cluster
15. Presence and absence of genes in each sample, with the corresponding source Gene ID

The .Rtab file contains a tab delimited binary matrix with the precence and abscence of each gene in each sample. This makes it easy to load into R using the read.table function, giving you access to a number of useful tools. The first row is the header containing the name of each sample, and the first column contains the gene name. In the matrix, 1 indicates the gene is present in the sample and 0 indicates it is absent.

6.1.3 pan_genome_reference.fa

This fasta file contains a single nucleotide sequence from each of the clusters in the pan genome. The name of each sequence is the source sequence ID followed by the cluster it came from. This file can be of use for reference guided assembly, whole genome MLST or for mapping raw reads to it.

6.1.4 .Rtab

Roary comes packaged with a script called create_pan_genome_plots.R. It requires R and the R-package ggplot2, and can be used to generate graphs from the .Rtab files, showing how the pan genome varies as genomes are added.

6.1.5 accessory_binary_genes.fa.newick

This is a tree in newick format, created using the binary presence and absence of accessory genes. It can for example be viewed in FigTree. The tree is only a quick and dirty tree, generated to provide a rough overview of the data. To generate a more accurate tree, we will use the core gene alignment a bit further on.

6.1.6 core_gene_alignment.aln

This is the multi-FASTA alignment of core genes that we created in the second run, using MAFFT. We will soon use this as input to build a phylogenetic tree.

6.1.7 clustered_proteins

In this file each line lists the sequences in a cluster. We will use this later on in the tutorial to query the pan genome.

For more information about the different output files, including some that we haven't mentioned here, please feel free to have a look at the Roary web page.

6.2 Query the pan genome

Roary comes with a script called query_pan_genome that can be used to examine the gene differences between groups of isolates. To have a look at the usage options for this script, you can do:

```
query_pan_genome -h
```

This will show you the following usage options:

```
Usage: query_pan_genome [options] *.gff
Perform set operations on the pan genome to see the gene differences
between groups of isolates.

Options: -g STR    groups filename [clustered_proteins]
         -a STR    action (union/intersection/complement/gene_multifasta/
                     difference) [union]
         -c FLOAT  percentage of isolates a gene must be in to be core [99]
         -o STR    output filename [pan_genome_results]
         -n STR    comma separated list of gene names for use with
                     gene_multifasta action
         -i STR    comma separated list of filenames, comparison set one
         -t STR    comma separated list of filenames, comparison set two
         -v        verbose output to STDOUT
         -h        this help message

Examples:
Union of genes found in isolates
         query_pan_genome -a union *.gff

Intersection of genes found in isolates (core genes)
         query_pan_genome -a intersection *.gff

Complement of genes found in isolates (accessory genes)
         query_pan_genome -a complement *.gff

Extract the sequence of each gene listed and create multi-FASTA files
         query_pan_genome -a gene_multifasta -n gryA,mecA,abc *.gff

Gene differences between sets of isolates
         query_pan_genome -a difference --input_set_one 1.gff,2.gff --
input_set_two 3.gff,4.gff,5.gff

For further info see: http://sanger-pathogens.github.io/Roary/
```

As you can see, this also shows us some examples uses. Give the first one a go, using the clustered_proteins file in the output_with_alignment:

```
query_pan_genome -a union \
         -g output_with_alignment/clustered_proteins *.gff
```

This will give us a file called pan_genome_results that contains a list of all genes in all samples, i.e. the pan genome. Have a look at the first ten lines of the newly generated file:

```
head pan_genome_results
```

As you can see, the list contains the names of the clusters (this is usually the most frequently occurring gene name from the sequences in the cluster or, if there is no gene name, a generic unique name group_XXX). For each cluster, there is a tab separated list of each sample specific gene belonging in that cluster.

In a similar way, you can use query_pan_genome to get a list of the core genes:

```
query_pan_genome -a intersection \
    -g output_with_alignment/clustered_proteins *.gff
```

and a list of the accessory genes:

```
query_pan_genome -a complement \
    -g output_with_alignment/clustered_proteins *.gff
```

query_pan_genome can also be used to extract the protein sequences for genes you are particulatly interested in. Try extracting the sequences for three genes by specifying the -n option and a comma separated list of the cluster names:

```
query_pan_genome -a gene_multifasta \
           -g output_with_alignment/clustered_proteins \
           -n patA_1,mnmG,hsdS_2 *.gff
```

You should have three new files, one for each gene you specified. Have a look at pan_genome_results_patA_1.fa:

```
cat pan_genome_results_patA_1.fa
```

This multi-FASTA file contains the three protein sequences in the specified cluster (patA_1).

There is yet more functionality of query_pan_genome, but we won't go into that here. For further information, please feel free to visit the Roary web page.

6.3 Draw a tree form the core gene alignment

The tree created by Roary (accessory_binary_genes.fa.newick) is just a quick tree to provide a rough insight into the data. To create a more accurate tree you can use the core gene alignment as input to a tree building software of your choice. RAxML is very accurate, however it is also fairly slow so in this tutorial we are going to use FastTree. You don't have to run this step, but if you want to you need to make sure FastTree is installed on your computer first.

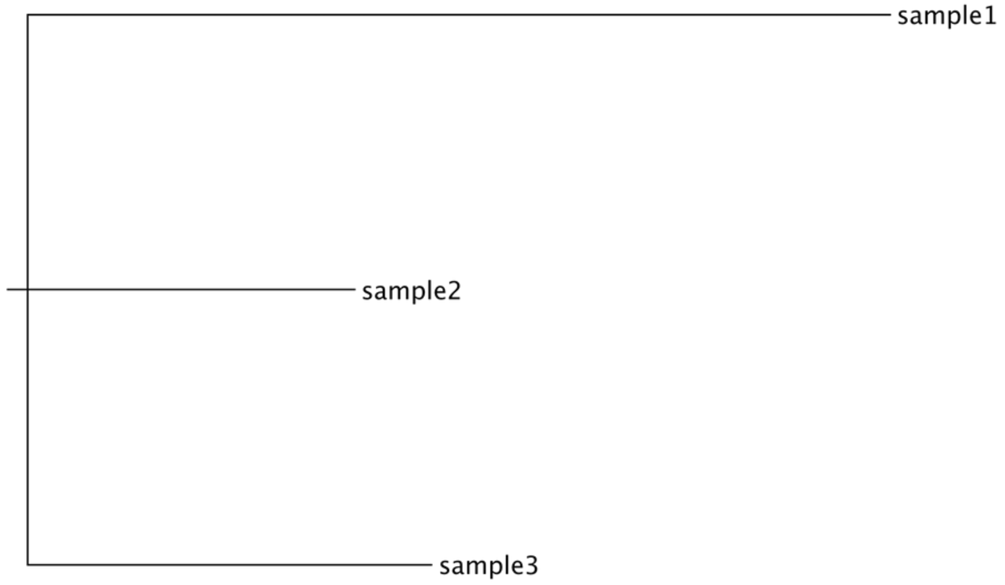
To create a tree in Newick format from a nucleotide alignment using a generalized time-reversible model (the -gtr option), do:

```
FastTree -nt -gtr output_with_alignment/core_gene_alignment.aln \
           > tree.newick
```

The tree in this case will look like:

`(sample1:0.006228253,sample2:0.002364375,sample3:0.002920483);`

We can view this in FigTree, which will look something like:



Phylogenetic tree

In the event that you did not run this step, a copy of tree.newick has been placed in the ROARY/tree/ directory for the next section of this tutorial.

6.4 Check your understanding

Q9: Approximately how many genes would you expect to see in the summary_statistics.txt file if you are working with a species with a genome size of 5,000,000 bases?
a) 500
b) 5000
c) 50,000

Q10: What does the accessory_binary_genes.fa.newick file provide?
a) A pylogenetic tree ready for publishing
b) Nothing, it is useless
c) A quick insight to the data

Q11: For query_pan_genome, what option should you use to get the accessory genome?
a) union
b) intersection
c) complement

The answers to these questions can be found here.

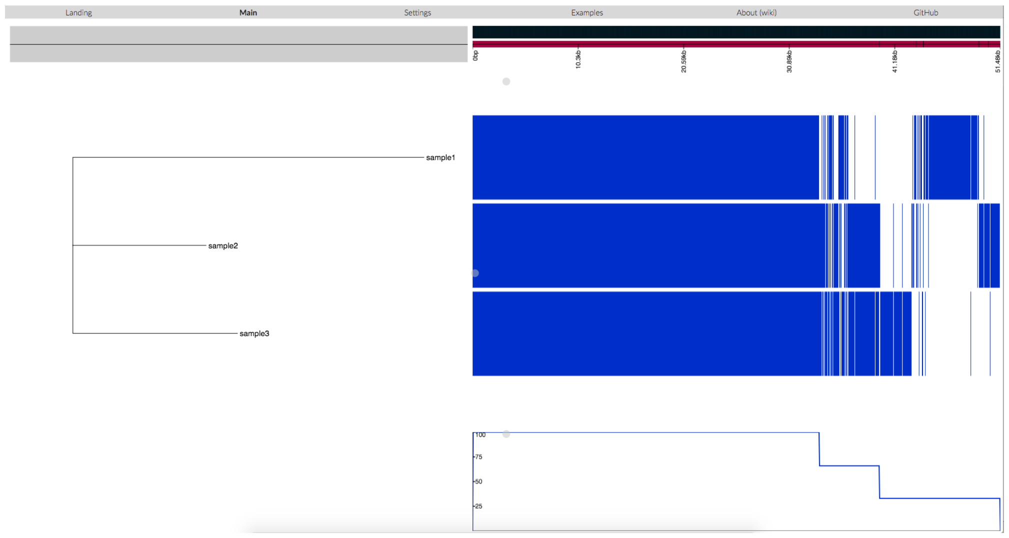
Now that you know a bit about the output from Roary, let's make use of them by visualising the results using Phandango.

You can also revisit the previous page, or go back to the index page.

7 Visualising the results with phandango

Phandango is a web based tool for visualising genomic analysis results such as phylogenetic trees and the output of Roary and other tools. Using phandango is really easy, you just drag your files onto the browser and drop them and Phandango will display them for you automatically. For more in depth information please feel free to have a look around the phandango wiki page.

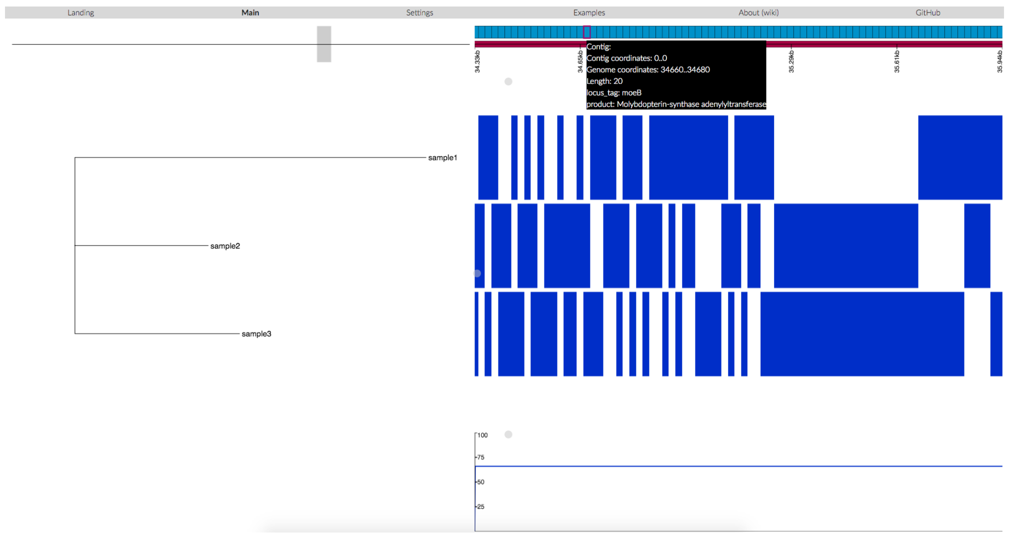
In this final section of the tutorial we are going to use Phandango to look at the pylogenetic tree we generated using FastTree, and the gene_presence_absence.csv file we got from Roary. Simply dragging the two files and droping them on the phandango front page will give you the following view:



phandango

The image shows our tree compared to a matrix with the presence and absence of genes in the pan genome. The graph on the bottom right provides a summary of the matrix above, indicating the percentage of isolates carrying a gene at each position.

You can zoom in on a particular area you are interested in simply by scrolling, and to see the annotation for a particular gene you can place your pointer over the corresponding bar at the top of the page, like in the figure below.



Contig:
Contig coordinates: 0..0
Genome coordinates: 34660..34680
Length: 20
locus_tag: moeB
product: Molybdopterin-synthase adenylyltransferase

zoom

In this case we are looking at a gene called moeB. We can see that the gene is present in sample 2 and 3, but not in sample 1, and that the product is an Molybdopterin-synthase adenylyltransferase.

Go ahead and play around a bit in Phandango. You can alter the layout in *settings* in the navigation bar at the top of the page, or by clicking and dragging the grey circles on the page. To download the current view in svg format, simply press *p*.

7.1 Check your understanding

In Phandango, zoom in on the gene cluster at position 25080.
Q12: What is the name of this gene cluster?
Q13: Is this a core gene?

The answers to these questions can be found here.

This marks the end of this tutorial. You can either revisit the previous page or head back to the index page.