Estimating Pangenomes with Roary

Farrah Sitto[1] and Fabia U. Battistuzzi*[,1,2]

[1]Department of Biological Sciences, Oakland University, Rochester, MI

[2]Center for Data Science and Big Data Analytics, Oakland University, Rochester, MI

*Corresponding author: E-mail: battistu@oakland.edu.

Associate editor: Barry G. Hall

Abstract

A description of the genetic makeup of a species based on a single genome is often insufficient because it ignores the variability in gene repertoire among multiple strains. The estimation of the pangenome of a species is a solution to this issue as it provides an overview of genes that are shared by all strains and genes that are present in only some of the genomes. These different sets of genes can then be analyzed functionally to explore correlations with unique phenotypes and adaptations. This protocol presents the usage of Roary, a Linux-native pangenome application. Roary is a straightforward software that provides 1) an overview about core and accessory genes for those interested in general trends and, also, 2) detailed information on gene presence/absence in each genome for in-depth analyses. Results are provided both in text and graphic format.

Key words: **Roary, pangenome, core genes, accessory genes.**

Protocol

The concept of a pangenome, the collection of all genes shared by multiple strains of a species, was first introduced by Tettelin et al. (2005) and has been selectively applied to investigate genomic variability at the species level in a few tens of species (both prokaryotes and eukaryotes) (Vernikos et al. 2015; McInerney et al. 2017). Since then, the applicability of the pangenome concept has grown alongside the exponential increase in sequenced genomes for subspecies lineages (e.g., strains, isolates, subspecies). The power of knowing the pangenome of a species resides in (i) guiding sequencing efforts to identify new unexplored genetic diversity within a species (represented by an open pangenome), (ii) providing information on shared and unique traits of strains within a species (exemplified by core and accessory genes), and, more recently, (iii) using it to identify species boundaries (represented by a high frequency of core genes).

These large-scale applications of a pangenome necessitate a fast and accurate software that can analyze and produce results for tens or hundreds of lineages in a reasonable amount of computational time. One such software is Roary (Page et al. 2015), a Linux-native software that takes as inputs GFF3 (General Feature Formats version 3) files (easily obtainable from NCBI) and outputs a series of files with statistics on genes shared by all or most (core and soft core genes) lineages or only by some genomes (accessory, further subdivided into shell and cloud genes). This software is complemented by python scripts and other software that produce a graphical view of the results.

Although other software are available for pangenome reconstructions, such as PGAP, PanX, get_homologues, and Pantools (Zhao et al. 2012; Contreras-Moreira and Vinuesa 2013; Sheikhizadeh et al. 2016; Ding et al. 2018), we found Roary to be the simplest and most flexible to use and, therefore, a good starting point for the novice to pangenome analyses. The potentially most challenging aspect of using Roary is its command-line interface, which doubles as strength because it makes it easy to be integrated into computational pipelines or large-scale analyses. To acquire some basic knowledge of command-line interface in Linux there are many online resources, such as https://ryanstutorials.net/linuxtutorial/commandline.php; last accessed December 9, 2019 or https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners; last accessed December 9, 2019, that will help users better understand the step-by-step process to install and use Roary described below.

Step 1: Installation of Roary

Roary is a Linux-native software that can be installed on Linux, MacOSX, and Windows machines in a variety of ways. In this section, we will provide a series of commands that will allow you to install Roary in a Linux environment (see Step 5 for installation in different operating system) (we show commands to be typed with a different font). The easiest way to run Roary is to install it in a Linux environment using the package manager "conda," which is part of the Anaconda distribution. This will work also in a MacOSX environment and the Linux Subsystem in Windows with very minor modifications (see Step 5).

The first step is to download Anaconda (https://www.anaconda.com/distribution/; last accessed December 9, 2019) for the appropriate operating system and select the most recent version of Python that is supported and updated regularly (currently it is Python 3.7) (e.g., for Linux: Anaconda3-2019.03-Linux-x86_64.sh). Open a terminal window and type `bash ~/Downloads/Anaconda3-2019.03-Linux-x86_64.sh` (if the file was downloaded in a different directory change `~/Downloads` to the correct location). Press

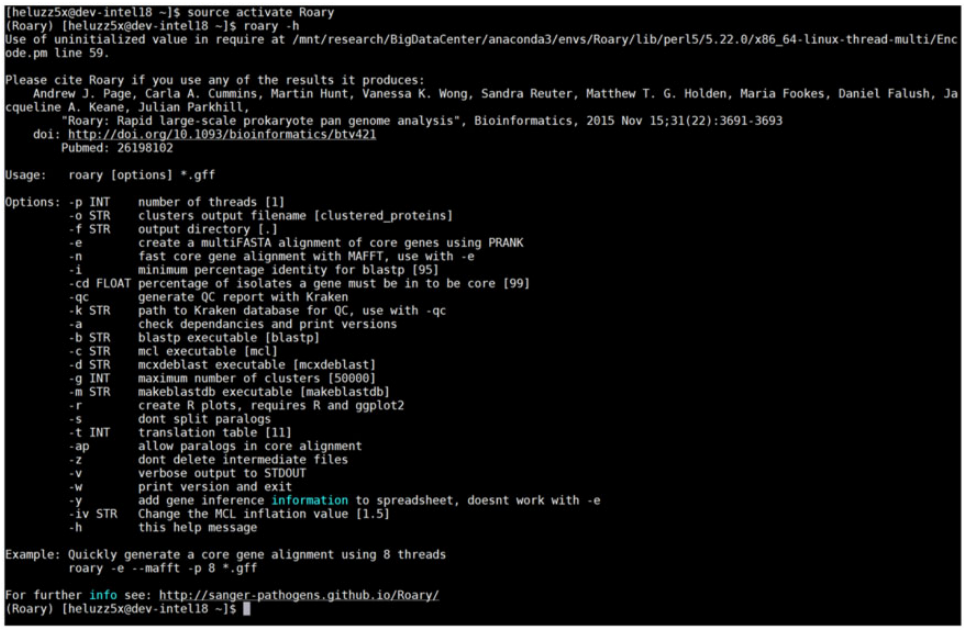

Fig. 1. Roary help file. The list of options available to complete an analysis with Roary is shown with the command: `roary -h`.

Enter to start the installation and space bar to visualize the license agreement. You will be prompted to accept the default location for installation by pressing Enter (or change the installation location), and the installation will start (it can take a minute or so to start seeing progress on the screen). Finally, answer "yes" to initialize Anaconda3 by running conda init and, at the end, you will see "Thank you for installing Anaconda!" Enter the command `source ~/.bashrc` for the installation to take effect. These instructions can also be found at https://docs.anaconda.com/anaconda/install/linux/; last accessed December 9, 2019. To test the installation, type in the Linux terminal `conda -V` and it will return the version of conda you just installed. Once conda has been installed correctly, the next step is to create an environment in which Roary will run. This can be achieved with the following command at the command prompt (shown in Linux as $): `conda create -name Roaryenv` (note that you can use any name for the environment instead of Roaryenv). In order to work within this environment, you will need to activate it (this step will need to be repeated every time you open a new terminal window): `source activate Roaryenv`.

Next, install Roary in your newly created environment with the following 5 "conda config" commands:

```
conda config --add channels r
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda
conda install roary
```

To check whether installation is successful type `roary -h` to visualize the list of parameters Roary uses (fig. 1). The location in which Roary is now installed does not have to also be the one that will include your input and output files. We suggest creating a separate directory in which to upload the input files and where the output files will be saved.

Step 2: Input Files

The format of the input files for Roary is GFF3 (General Feature Format version 3). This format includes a series of information in a specific order and needs to be followed strictly for Roary to accept the input file (see https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md; last accessed December 9, 2019 for a description of the format). There are two primary ways to obtain GFF3 files: from the NCBI website or from the software Prokka by converting .fna files into GFF3 (see Step 5). An easy way to obtain the input files without additional software installation is to download genome *.gbff files from NCBI and then run the *bp_genbank2gff3.pl* script. This is a Perl script that is installed along with Roary and that can be found in the Roary conda environment in the directory "bin." It is also available through BioPerl (https://bioperl.org/INSTALL.html; last accessed December 9, 2019) and can be easily run in the terminal window. Note that for this script to work, Perl needs to be installed in the system you are using (https://www.activestate.com/products/activeperl/downloads/; last accessed December 9, 2019). For example, let us say that you are interested in estimating the pangenome of three strains of *Bifidobacterium animalis A6*, *KLDS2.0603*, and *RH*. From the Genome function in NCBI (https://www.ncbi.nlm.nih.gov/genome; last accessed December 9, 2019) you can browse by organism and search for *B. animalis*. The individual assemblies can

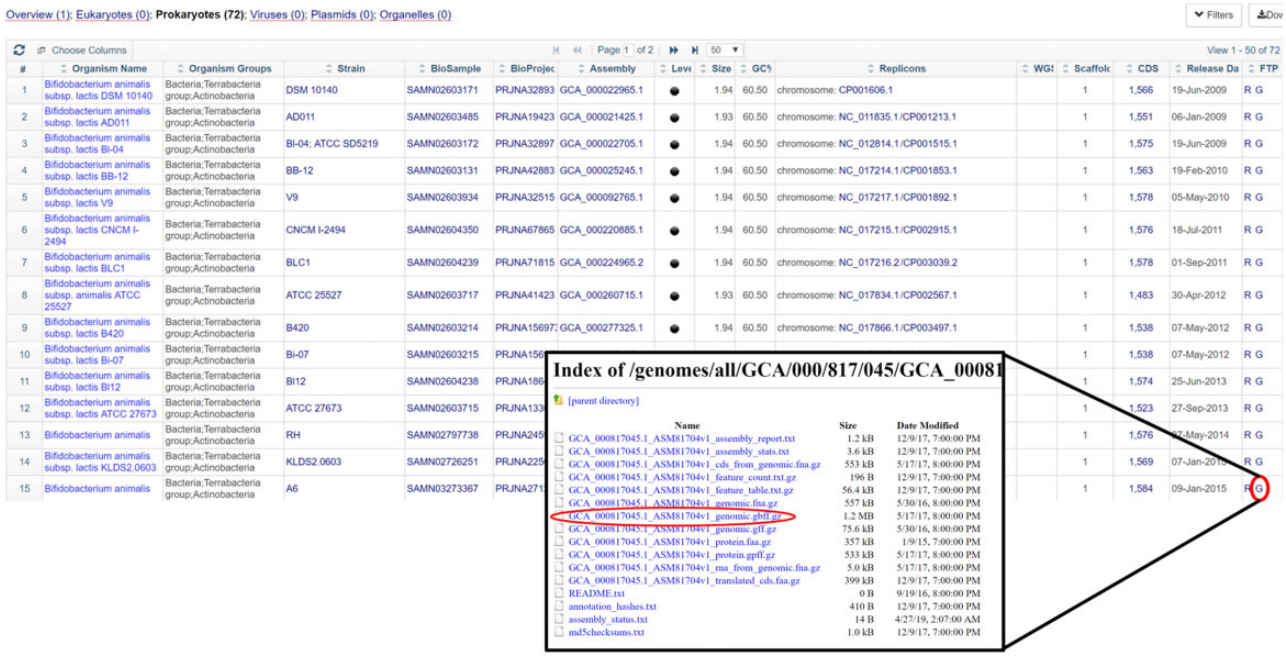

Fig. 2. How to obtain input files in GFF3 format from NCBI. Links to proceed to download are circled in red.

be visualized by selecting "Prokaryotes." After having identified the strains of interest, select the GenBank FTP site on the right-hand side and download the *.gbff.gz file for each of them (fig. 2).

Next, move all the downloaded gbff files into a single directory (if you have used a Windows machine to download the files, upload them into the Linux machine) and, from terminal, issue the command `perl bp_genbank2gff3.pl *.gbff.gz`.

If you are using the perl script within the Roary environment, you will need to specify the path to the script (e.g., `perl /home/Roaryenv/bin/bp_genbank2gff3.pl`). To identify the path of this perl script, use the command `which bp_genbank2gff3.pl`. If your current working directory is not the same as the one where the gbff files are, either navigate into that directory and use the above command or add the path of the directory before the "*" (e.g., `perl /home/Roaryenv/bin/bp_genbank2gff3.pl /home/Roary/Inputs/*.gbff.gz`). This command will create as many output files as the input files all with an extension *.gff. These will be the input files for Roary.

Step 3: Parameters and Commands

Roary can be run very easily with a single short command: `roary *.gff` (remember to activate the Roary environment [`source activate Roaryenv`] every time you use terminal window for the first time).

This command will run Roary with default parameters (see below) from within a directory that contains all the gff3-converted files obtained from Step 2. All output files generated will be located in this same directory, which could make downstream analyses more difficult. To specify an output directory, add the option **-f** to the command: `roary -f output_dir *.gff` (where output_dir is user-defined).

Options in Roary fall broadly into three categories: file access, analysis settings, and visualization. The "file access" settings are the least likely to need modification. They include those that allow users to manipulate the location of inputs/outputs and the location (path) of where the software that Roary depends on is located. Roary requires *mcl*, *blastp*, *mcxdeblast*, and *makeblastdb* that are installed along with Roary within the environment in conda. However, users can use a different location of these software, if preferred. Additionally, users can provide directory names for outputs (option **-f**).

The "analysis settings" parameters allow users to refine the sensitivity of the analysis itself to identify core and accessory genes. These are most likely the parameters that users will want to modify to explore the robustness of the results to variations. For computational speed, the **-p** option will allow users to select the number of threads to use during the computation. Many new computers are multicore with multithreads for each core, so selecting >1 (e.g., `roary -f output_dir -p 10 *.gff`) for this parameter is likely to speed up the analysis. For the pangenome calculation, the two most important parameters are the threshold (in percentage) of isolates required to define a core gene (**-cd**: default is 99%) and the minimum percentage identity for sequence comparisons performed by BlastP (**-i**: default is 95%). Decreasing the threshold of isolates will increase the number of core genes identified, and increasing the minimum identity will partition the genes in more and smaller clusters.

Finally, to visualize results, Roary has a series of options. The standard option, which requires no additions to the previous command, will produce a series of text outputs (see Step 4). If the user desires an additional graphical output, the option **-r** can be added to produce plots using R (this option

will need R and ggplot2 to be installed). Note that the graphs can also be obtained after the results have already been produced because Roary will output R formatted files in addition to text files. Finally, one of the most useful parameters for visualization is the possibility of creating alignments from core genes (options **-e** and **-n**). Such files are potentially important for downstream analyses including phylogenetic tree reconstruction and SNP identification. Additional visualization tools are provided as separate scripts and packages (e.g., roary_plots.py) that can be found on the main Roary website (https://sanger-pathogens.github.io/Roary/; last accessed December 9, 2019).

Step 4: Interpretation of Output Files

A simple run of Roary will produce 17 output files, of which the summary_statistics.txt and the gene_presence_absence.csv are the most important. The summary_statistics text file reports the number of genes in each of four categories (core, soft, shell, and cloud) and also the total number of genes in the pangenome. These values effectively describe the nature of the pangenome of the species analyzed. The gene_presence_absence file provides additional information including the individual gene IDs of sequences that belong to each of the categories in the summary statistic (although this is not clearly stated, it can be easily inferred by calculating the ratio of the number of genes present in each cluster and the total number of genomes analyzed).

Other output files (starting with "number_of_") provide information specific to each category (i.e., core or accessory). It should be noted that for these outputs (e.g., number_of_conserved_genes.Rtab) the results for ten random iterations of the input files are shown. This is important because pangenome calculations will vary depending on the order in which genomes are added and results obtained from multiple orders will allow to establish minimum and maximum boundaries around the core and accessory gene estimates. These files are provided in R format to facilitate downstream analyses. One example of such an analysis is to obtain curves for the number of core and accessory genes to determine whether the pangenome is closed or open (Tettelin et al. 2005). This can be easily done using the Rtab outputs from Roary and the *create_pan_genome_plots.R* script (available in the Roary conda environment). Unfortunately, there is no statistical analysis carried out automatically on the curves but it can be done separately, for example, by fitting an exponential curve and calculating its distance to the empirical curve through a least square method or using Heap's law (Tettelin et al. 2008).

To view results graphically, there are two outputs (ending in _graph.dot) that allow the user to glean over information regarding the relative position of genes that belong to either accessory or core categories. These files can be visualized using the open-source software Gephi (www.gephi.org; last accessed December 9, 2019) and can be useful, for example, to investigate patterns in gene clusters such as operons.

An interesting additional feature of Roary is the possibility of comparing different pangenomes to identify genes that are uniquely present in one set of strains and not others. This kind of analysis can be done calling the `query_pan_genome -a difference --input_set_one 1.gff, 2.gff --input_set_two 3.gff, 4.gff -g clustered_proteins` (where the *.gff files are the names of the genomes of interest in two subsets). Finally, the same `query_pan_genome` function can be used to output genes that are unique, shared by all, or shared by some of the strains (e.g., `query_pan_genome -a union -g clustered_proteins *.gff`).

A good description of all the output files created by Roary is available in the supplementary material of the Roary publication (Page et al. 2015) and, in a less detailed way, on the github page (https://sanger-pathogens.github.io/Roary/; last accessed December 9, 2019).

Step 5: Installation on MacOSX or Windows and Use of Prokka

Installation on MacOSX

Download Anaconda3 (https://www.anaconda.com/distribution/; last accessed December 9, 2019) for MacOSX and select the most recent version of Python that is supported and updated regularly (currently it is Python 3.7) (e.g., for Anaconda3-2019.03-MacOSX-x86_64.sh). Follow the instructions at https://docs.anaconda.com/anaconda/install/mac-os/; last accessed December 9, 2019, which are very similar to those for the Linux operating system. Once conda is installed, follow the instructions given for Linux (see Step 2) to create a Roary environment and install Roary.

Installation on Windows

Because Roary is a native Linux software, it cannot run directly in Windows. There are two ways of running Roary on a Windows machine: First, Windows 10 users (version 1709 and later) can install the Linux Subsystem on Windows; second, it can run within a virtual machine. For the first scenario, launch Control Panel > Programs and Features > Turn Windows Features on or off and check "Windows Subsystem for Linux." Then, open Microsoft store, search for "Linux," and select the Linux distribution desired (e.g., in this tutorial we use Ubuntu). Install and launch the new distribution and follow the prompts to complete the installation process in the command line window (https://docs.microsoft.com/en-us/windows/wsl/install-win10; last accessed December 9, 2019 and https://docs.microsoft.com/en-us/windows/wsl/initialize-distro; last accessed December 9, 2019). To be able to use Roary within your new Linux distribution, follow the instructions described above for the Linux installation. For the second scenario, download the VirtualBox installer from VirtualBox (https://www.virtualbox.org/wiki/Downloads; last accessed December 9, 2019). Double click the executable and proceed with the installation. Download also the virtual machine (VM) created by the authors of Roary from ftp://ftp.sanger.ac.uk/pub/pathogens/pathogens-vm/pathogens-vm.latest.ova. After starting the virtual box, go to

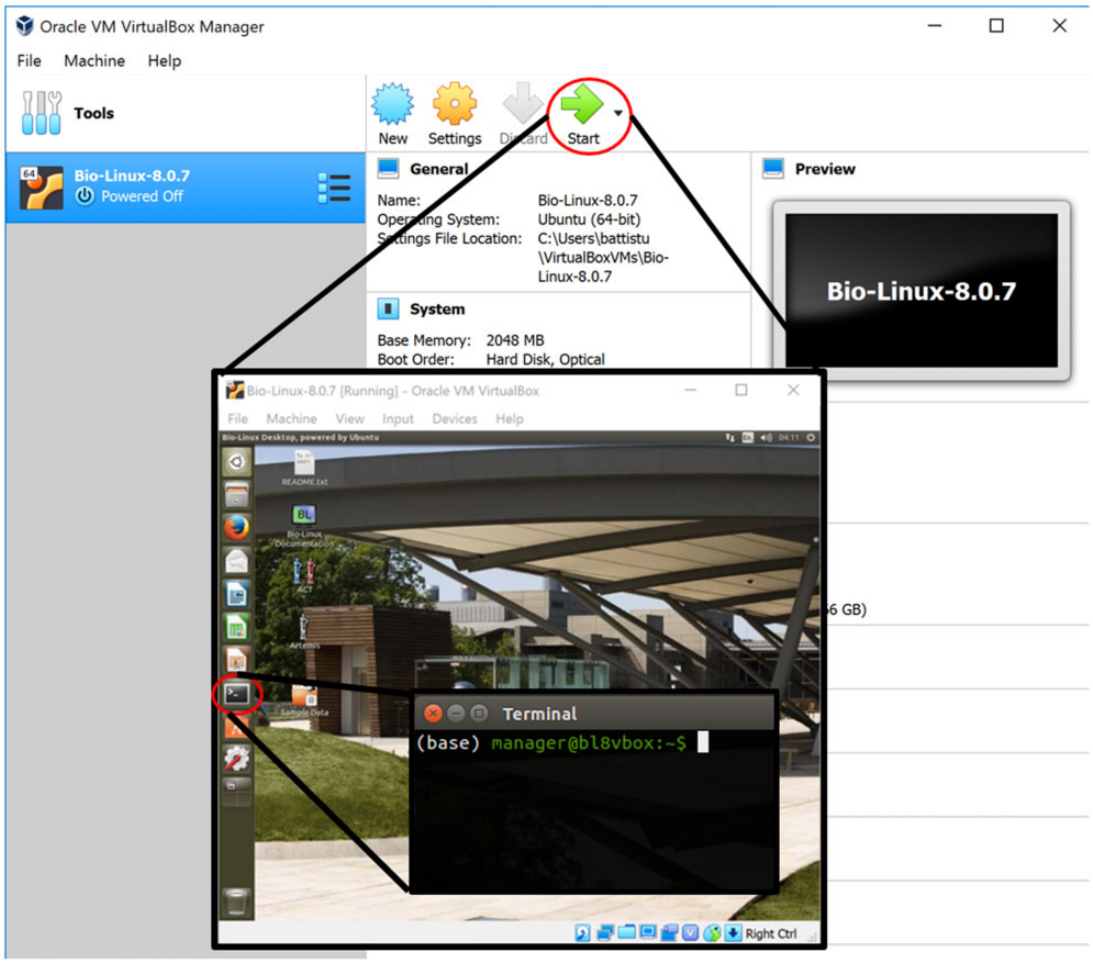

FIG. 3. Virtual machine environment to run Roary on Windows. Circled in red are the icons to start the virtual machine and the terminal window.

File → "Import appliance" and select the VM (*.ova file) you downloaded. To start the VM, click on the green arrow icon and a new window will open showing the VM desktop. On the left-hand side, click on the terminal window icon (fig. 3) and type `sudo apt-get install virtualbox-guest-utils` (the password is manager).

To be able to use Roary within the VM, you will follow the Linux installation instructions. However, this requires that files are shared between the host (Windows) and the VM. To achieve this, a shared directory has to be created and used to exchange files. Within the Windows machine, go to the Anaconda website and download the Linux version as shown in Step 1. Save this file in a directory you will share with the VM. Then, switch to the VM, select Devices → "Shared folders" → "Shared folder settings" and click on the "Add folder" icon on the right-hand side. Provide the path of the location of the Anaconda installer, assign a name to the VM (e.g., RoaryVM), a path where it will be mounted (e.g., /mnt/share/) and check "auto mount" and "make permanent" to ensure that the folder will be recognized upon restart of the VM. Then, in the VM terminal, type `sudo mkdir/mnt/share/`(the password is again manager) and then `sudo mount -t vboxsf RoaryVM/mnt/share/`. If the shared folder is not visible, repeat the mounting command.

The contents of the shared directory are now visible from the VM (`ls/mnt/share/`) and can be used to proceed with a normal Roary installation for Linux. Input and output files for Roary can be exchanged through the shared folder if the path is provided at the command line (e.g., `roary -f /mnt/share/RoaryVM/output /mnt/share/RoaryVM/input/*.gff`).

Prokka to Create Input Files

An alternative way to converting gbff files into input files for Roary is to use Prokka. This is particularly useful when gbff files are not already available, as it may be the case for sequencing projects that are in progress. First, using terminal in Linux (or in MacOSX or Windows) type `conda install -c conda-forge -c bioconda prokka`. To check whether Prokka was installed correctly, type `prokka -h` and the menu options of Prokka will be listed.

Next, download *.genomic.fna.gz files from NCBI for the strains of interest, extract them, and upload these uncompressed files into the Linux/MacOSX/Windows machine. In the terminal window type: `prokka -kingdom Bacteria -outdir prokka_GCA_XXXXX -genus YYYYY -locustag GCA_XXXXX GCA_XXXXX_ASMZ`

ZZZZ_genomic.fna where XXXXX is the genome and ZZZZZ is the assembly number of one of the strains and YYYYY is the genus of the same strain (e.g., for one of the three *B. animalis* strains mentioned in Step 1: `prokka –kingdom Bacteria –outdir prokka_GCA_000816205 –genus Bifidobacterium –locus-tag GCA_000816205 GCA_000816205.1_ASM81620v1_genomic.fna`). Repeat for all the strains (each strain will take a few minutes to process). Each run will produce multiple output files, one of which is the GFF3 format required by Roary.

Applications of a Pangenome

The concept of a pangenome has become useful in many different fields, from classification to genome evolution. The original and most typical application of the results of a pangenome analysis is to identify the cumulative curve of genetic variability that can be attributed to a species as more and more individual genomes are sequenced. In a sense, this way of analyzing prokaryotes (or viruses) mirrors basic population genetic studies in eukaryotes where the sequencing of multiple individuals is necessary to understand the range of polymorphisms within a species (Muzzi and Donati 2011; Nguyen et al. 2015). Indeed, pangenome approaches are starting to be used in read mapping software to account for polymorphisms that would otherwise be lost or lead to errors in read alignments (Nguyen et al. 2015; Eggertsson et al. 2017). In the case of the pangenome, gene counts are used as proxy of genetic variability with genes unique and new to each strain adding to the overall genetic makeup of a species. The expectation is that, as the number of strains analyzed grows, the number of new genes will approach 0 and the total size of the pangenome will stabilize (reaching a plateau in an initially exponential curve) leading to the definition of a closed pangenome (Tettelin et al. 2005, 2008). If the plateauing is not observed, the pangenome is defined open and it is expected that more genomes will need to be sequenced to be able to estimate the total genetic complement of the species. Tettelin et al. (2008) have proposed to compare the new genes' accumulation curve with Heaps' law to determine statistically whether a pangenome is open or closed. However, even with this statistical framework, it is not possible to evaluate the functional weight, if any, of each new gene and, therefore, their biological importance or evolutionary driving force remains unknown. In other words, it is possible that new genes identified in a strain will not be maintained within that genome over long evolutionary time frames (because of selective or neutral forces; McInerney et al. [2017] but see also Rodriguez-Valera and Ussery [2012]) and, therefore, they may not effectively contribute to the long-term genetic makeup of the species. Additionally, considering the very small number of sequenced genomes available compared with predicted species numbers (Locey and Lennon 2016), it is possible that a newly sequenced strain will reopen a currently closed pangenome.

A more recent application of pangenomes is to better define the concept of species in prokaryotes (Moldovan and Gelfand 2018). Defining prokaryotic species boundaries is a long-standing issue that, for now, has been approached using DNA similarity thresholds (e.g., average nucleotide identity measures; Jain et al. 2018). However, a pangenome approach has the advantage of adding an evolutionary perspective by considering not only identity (**-i** parameter in Roary) but also orthology/paralogy and gene flow (Bobay and Ochman 2017; Moldovan and Gelfand 2018).

Finally, pangenome results can be used to investigate the correlation between the spread of some genes and the traits they encode. A corollary software, Scoary (Brynildsrud et al. 2016), is available to work with Roary's outputs to identify those genes (core or accessory) that are associated with specific traits. Such analysis could explain current trait distributions and the evolutionary history of those traits (Abreo and Altier 2019).

Alternative Resources

The number of software that can estimate a pangenome is growing. Originally, Roary was compared with a few other software, like PGAP, and was shown to be computationally more efficient (speed and memory usage) while producing comparable results (Page et al. 2015). Other tools that have been developed since Roary was released include PGAP-X, PanTools, and panX (Sheikhizadeh et al. 2016; Ding et al. 2018; Zhao et al. 2018). PGAP-X is unique in its visualization features that allow to observe the alignment of multiple genomes at once. PanTools, instead, fills a unique niche because it is built to analyze eukaryotic genomes (with genes that have introns and exons) that Roary cannot analyze. Finally, panX differs from Roary because it is able to analyze genomes with higher genomic diversity between them whereas Roary is recommended for highly similar (within species) genomes.

Acknowledgments

We thank Cody Clark and Victoria Hall for installation and testing of Roary. This work was supported by the National Institute of General Medical Sciences at the National Institute of Health (R15GM121981 to F.U.B.) and the National Aeronautics and Space Administration (NNX16AJ30G to F.U.B.).

References

Abreo E, Altier N. 2019. Pangenome of *Serratia marcescens* strains from nosocomial and environmental origins reveals different populations and the links between them. *Sci Rep*. 9(1):46.

Bobay L-M, Ochman H. 2017. Biological species are universal across life's domains. *Genome Biol Evol*. 9(3):491–501.

Brynildsrud O, Bohlin J, Scheffer L, Eldholm V. 2016. Rapid scoring of genes in microbial pan-genome-wide association studies with Scoary. *Genome Biol*. 17(1):238.

Contreras-Moreira B, Vinuesa P. 2013. GET_HOMOLOGUES, a versatile software package for scalable and robust microbial pangenome analysis. *Appl Environ Microbiol*. 79(24):7696–7701.

Ding W, Baumdicker F, Neher RA. 2018. panX: pan-genome analysis and exploration. *Nucleic Acids Res*. 46(1):e5.

Eggertsson HP, Jonsson H, Kristmundsdottir S, Hjartarson E, Kehr B, Masson G, Zink F, Hjorleifsson KE, Aslaug J, Adalbjorg J, et al. 2017. Graphtyper enables population-scale genotyping using pangenome graphs. *Nat Genet*. 49(11):1654–1660.

Jain C, Rodriguez-R LM, Phillippy AM, Konstantinidis KT, Aluru S. 2018. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat Commun*. 9(1):5114.

Locey KJ, Lennon JT. 2016. Scaling laws predict global microbial diversity. *Proc Natl Acad Sci U S A*. 113(21):5970–5975.

McInerney JO, McNally A, O'Connell MJ. 2017. Why prokaryote have pangenomes. *Nat Microbiol*. 2(4):17040.

Moldovan MA, Gelfand MS. 2018. Pangenomic definition of prokaryotic species and the phylogenetic structure of *Prochlorococcus* spp. *Front Microbiol*. 9:428.

Muzzi A, Donati C. 2011. Population genetics and evolution of the pan-genome of *Streptococcus pneumoniae*. *Int J Med Microbiol*. 301(8):619–622.

Nguyen N, Hickey G, Zerbino DR, Raney B, Earl D, Armstrong J, Kent WJ, Haussler D, Paten B. 2015. Building a pan-genome reference for a population. *J Comput Biol*. 22(5):387–401.

Page AJ, Cummins CA, Hunt M, Wong VK, Reuter S, Holden MTG, Fookes M, Falush D, Keane JA, Parkhill J. 2015. Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics* 31(22):3691–3693.

Rodriguez-Valera F, Ussery DW. 2012. Is the pan-genome also a pan-selectome? *F1000Res*. 1:16.

Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. 2016. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics* 32(17):i487–i493.

Tettelin H, Masignani V, Cieslewicz MJ, Donati C, Medini D, Ward NL, Angiuoli SV, Crabtree J, Jones AL, Durkin AS, et al. 2005. Genome analysis of multiple pathogenic isolates of *Streptococcus agalactiae*: implications for the microbial "pan-genome." *Proc Natl Acad Sci U S A*. 102(39):13950–13955.

Tettelin H, Riley D, Cattuto C, Medini D. 2008. Comparative genomics: the bacterial pan-genome. *Curr Opin Microbiol*. 11(5):472–477.

Vernikos G, Medini D, Riley DR, Tettelin H. 2015. Ten years of pan-genome analyses. *Curr Opin Microbiol*. 23:148–154.

Zhao Y, Sun C, Zhao D, Zhang Y, You Y, Jia X, Yang J, Wang L, Wang J, Fu H, et al. 2018. PGAP-X: extension on pan-genome analysis pipeline. *BMC Genomics* 19(1 Suppl):36.

Zhao Y, Wu J, Yang J, Sun S, Xiao J, Yu J. 2012. PGAP: pan-genomes analysis pipeline. *Bioinformatics* 28(3):416–418.