ASGARD+: A New Modular Platform for Bacterial Antibiotic-Resistant Analysis

Maripaz Montero-Vargas,[1,4] Alex Saenz-Rojas,[1] Marcela Suárez-Esquivel,[2] and Lizbeth Ramirez-Carvajal[3]

[1]Advanced Computing Laboratory (CNCA) of the National High Technology Center (CeNAT-CONARE), San Jose, Costa Rica
[2]Programa de Investigación en Enfermedades Tropicales (PIET), Escuela de Medicina Veterinaria, Universidad Nacional, Heredia, Costa Rica
[3]Former affiliation: National Laboratory of Veterinary Services (LANASEVE), Ministry of Agriculture of Costa Rica, Heredia, Costa Rica
[4]Corresponding author: *mmontero@cenat.ac.cr*

Published in the Microbiology section

ASGARD+ (Accelerated Sequential Genome-analysis and Antibiotic Resistance Detection) is a command-line platform for automatic identification of antibiotic-resistance genes in bacterial genomes, providing an easy-to-use interface to process big batches of sequence files from whole genome sequencing, with minimal configuration. It also provides a CPU-optimization algorithm that reduces the processing time. This tool consists of two main protocols. The first one, ASGARD, is based on the identification and annotation of antimicrobial resistance elements directly from the short reads using different public databases. SAGA, enables the alignment, indexing, and mapping of whole-genome samples against a reference genome for the detection and call of variants, as well as the visualization of the results through the construction of a tree of SNPs. The application of both protocols is performed using just one short command and one configuration file based on JSON syntax, which modulates each pipeline step, allowing the user to do as many interventions as needed on the different software tools that are adapted to the pipeline. The modular ASGARD+ allows researchers with little experience in bioinformatic analysis and command-line use to quickly explore bacterial genomes in depth, optimizing analysis times and obtaining accurate results. © 2023 Wiley Periodicals LLC.

Basic Protocol 1: ASGARD+ installation
Basic Protocol 2: Configuration files general setup
Basic Protocol 3: ASGARD execution
Support Protocol: Results visualization with Phandango
Basic Protocol 4: SAGA execution
Alternative Protocol 1: Container installation
Alternative Protocol 2: Run ASGARD and SAGA in container

Keywords: antibiotic resistance • phylogenetic tree • pipeline automatization • single nucleotide polymorphism (SNP) • whole-genome sequencing

How to cite this article:
Montero-Vargas, M., Saenz-Rojas, A., Suárez-Esquivel, M., & Ramirez-Carvajal, L. (2023). ASGARD+: A new modular platform for bacterial antibiotic-resistant analysis. *Current Protocols*, *3,* e680. doi: 10.1002/cpz1.680

CURRENT PROTOCOLS
A Wiley Brand

Current Protocols e680, Volume 3
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.680

INTRODUCTION

Epidemiological surveillance has played an essential role in health systems worldwide in analyzing the role, distribution, and effect of antimicrobial-resistant bacteria that cause several infectious diseases (Gilchrist, Turner, Riley, Petri, & Hewlett, 2015; Simar, Hanson, & Arias, 2021). Research on the bacterial resistance mechanisms is essential for tracking pathogen outbreaks and decision making related to public health (Mäklin et al., 2021; Tümmler, 2020). In response to these needs, bioinformatic methodologies have been developed to facilitate the efficient analysis of pathogens through the use of next-generation sequencing (NGS) and whole-genome sequencing (WGS) technologies (Van Camp, Haslam, & Porollo, 2020). The massive data generated from these technologies require substantial processing to obtain valuable information, which is why the need for the standardization of bioinformatic protocols has arisen, mainly for the effective analysis of big sample batches (Shelenkov, 2021). This article presents the ASGARD+ (Fig. 1) command-line-based platform for the analysis of large WGS data sets, and a step-by-step protocol on how to install and run this tool. Basic Protocol 1 includes the installation and configuration of the tool for any Linux-based operating system. Basic Protocol 2 explains how to edit the configuration files needed to run the protocols. Next, in Basic Protocol 3, we explain how to execute the first utility of the platform, called ASGARD, which enables the analysis of multiple WGS samples at the same time; using the ARIBA tool (Hunt et al., 2017), antimicrobial resistance genes found in the samples can be annotated. As a result, this protocol produces a summary of the genes found and the necessary files to create a visualization on the Phandango web platform (Hadfield et al., 2018). Details of how to use this platform are explained in the Support Protocol. Basic Protocol 4 details how to run the second utility of the ASGARD+ platform, called SAGA. SAGA detects single nucleotide polymorphisms (SNPs) in the WGS samples using a reference genome.

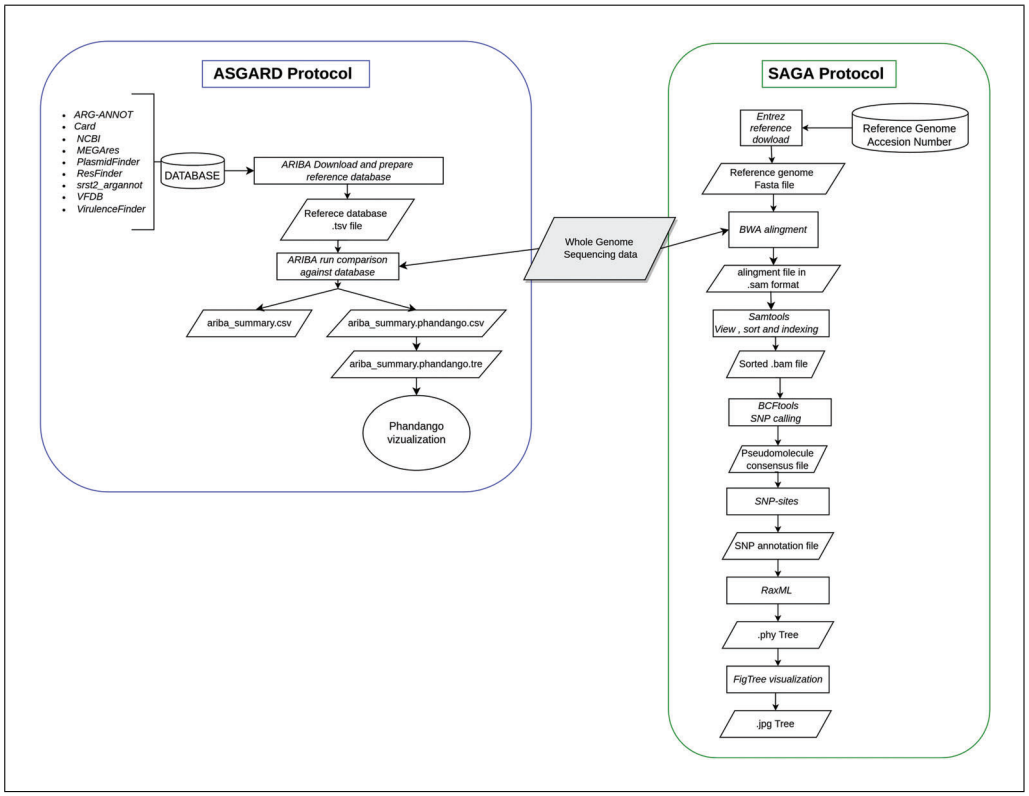

Figure 1 General workflow applied in ASGARD and SAGA protocols. This figure summarizes the general steps run in the ASGARD+ platform and which files are needed and produced in each step. In this scheme, the different shapes have different meanings: cylinder (database), rectangle (execution step), rhomboid (file), circle (web platform).

As a result of this protocol, a phylogenetic tree is obtained, and it can be used to compare the variants between the samples through its visualization in tools such as FigTree (*http://tree.bio.ed.ac.uk/software/figtree/*). Alternate Protocol 1 provides another option for installing ASGARD+ using containers, while Alternate Protocol 2 indicates how to run ASGARD+ in a container. These last two protocols provide an alternative for generating replicable configuration of the tool and its implementation. ASGARD+ aims to facilitate the processing of large data sets generated during epidemiological surveillance, and due to its easy implementation, this tool can be employed by users with no background or experience in the use of bioinformatics methodologies to analyze any number of bacterial genomes using Illumina sequences, regardless of their length.

ASGARD+ INSTALLATION

BASIC PROTOCOL 1

Similar to their experimental counterparts, bioinformatic analyses require the execution of multiple steps and procedures to achieve valid and reproducible results. Creating a controlled environment where the execution parameters can be established and documented is essential to ensure the quality of the experiments (Carriço, Rossi, Moran-Gilad, Van Domselaar, & Ramirez, 2018). Configuring and running multiple bioinformatics software can easily become complex since it requires repetitive processes that, in turn, can become points of failure. ASGARD was developed using Python 3.6 as its foundation, along with multiple built-in libraries that provide most of its functionalities; however, several other dependencies are used to broaden its capabilities. These external dependencies are based on open-source software available on the default repository of the Python package manager, pip (*https://pypi.org/*). This protocol establishes the necessary actions to replicate the environment in which the tool was developed and tested. To install ASGARD+, it is recommended to install Conda (*https://docs.conda.io/en/latest/*), which provides a simple solution to manage multiple Python versions and environments. A list of requirements is provided in the repository's main branch, which contains the minimum version required for the installation. This program was developed for Linux environments, so its operation on other platforms cannot be guaranteed. ASGARD uses multiple open-source programs to run the analysis; these requirements must be ready before the first execution of ASGARD. The version of each software is checked before each run. The steps to fulfill these requisites require root permission of the operating system. Multiple versions of these programs are available in the repositories; however, the specific versions tested are shown below.

Materials

Hardware

Computer or computational cluster with access to command line environment

Software

A Linux-based operating system—in this protocol a Debian- or Ubuntu-based OS is required, however it could be adapted to run in other Linux distributions
Bash terminal or equivalent
Miniconda installation script
Git version control system
A Python environment running version 3.6 or higher
ASGARD application files
ASGARD default workflow configuration files

1. Update the software repositories.
   i. `$ apt update`

This command fetches the latest information about packages and their versions. This information is gathered from the default repositories of the operating system which are defined in the `/etc/apt/sources.list` file.

2. Download and install the version control software git and the wget tool.

   i. `$ sudo apt install git wget`

   By running this command, it is possible to install the latest version of git, which is an open-source solution to run a distributed version control system. This allows the user to get the source code of Asgard from the main repository. wget is a free software for retrieving files using different communication protocols.

3. Download the latest version of ASGARD from the repository using the git repository.

   i. `$ git clone https://gitlab.com/CNCA_CeNAT/asgard`

   Using the git tool, clone (download) the latest version of ASGARD with the default configuration files from the official repository. Once the repository is downloaded, the user must have an `asgard` folder. This folder contains the resources and codes necessary for the implementation of this platform.

4. Install the free community edition of conda, the environment manager system, and create a new environment for ASGARD+ to run.

   i. `$ wget https://repo.anaconda.com/miniconda/Miniconda3-py38_4.10.3-Linux-x86_64.sh -O miniconda.sh`

   ii. `$ bash./miniconda.sh -b -p $HOME/miniconda && $HOME/miniconda/bin/conda init && exec bash`

   With conda, it is possible to manage package installation and Python environments to isolate the dependencies for each configuration file.

   iii. `$ conda create --name asgard+ python=3.6 -y && conda activate asgard+`

5. Locate the `asgard` directory downloaded from git repository and run the dependencies installation scripts.

   i. `$ cd asgard`

   ii. `$./script/pip-dependencies.sh`

   iii. `$./script/conda-dependencies.sh`

   iv. `$ sudo./script/debian-dependencies.sh`

BASIC PROTOCOL 2

CONFIGURATION FILES GENERAL SETUP

The ASGARD+ execution process is based on the concept of pipelines, i.e., the execution of a series of sequential steps that analyze the input information. Each one of these steps or stages generates partial results, which in turn feed the next program in the sequence. In ASGARD+, these pipelines are described in the custom AJSON type configuration files, which are based on the concepts of "JavaScript Object Notation" but add to it a series of functionalities that extend its operation. To run any pipeline using ASGARD+, it must be expressed in an AJSON file complying with the established structure. The schema and anatomy of these files can be found on the main page of the repository. By default, two main configuration files—`ASGARD.ajson` and `SAGA.ajson`—are provided, which in this case can be seen as proof of the concept of these configuration files. One of the essential parts of these files resides in the “constant” property where the paths to the input files are located and the path where the output files should be created. These files are modifiable and configurable according to the user’s needs, but to make use of these original pipelines, it is only necessary to modify the three main parameters described below.

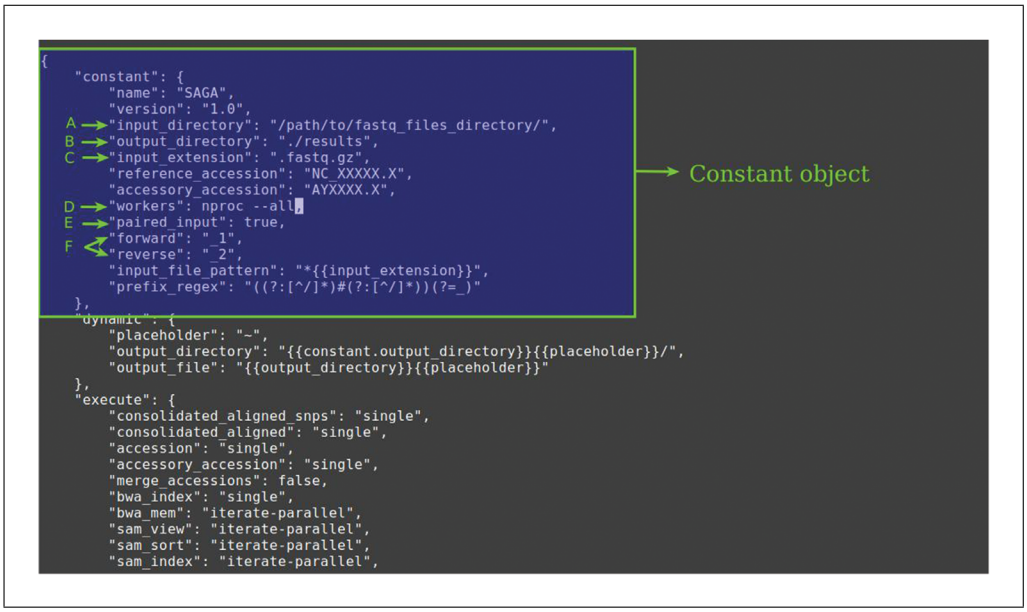

Figure 2 Structure and composition of the ASGARD+ platform configuration files. This image shows the `saga.ajson` file as an example. The constant object is highlighted in the blue box. (**A**) Parameter to indicate the directory path where the WGS files are found. (**B**) Parameter to indicate the output directory. (**C**) Parameter to indicate the extension of the fastq format. (**D**) Parameter to indicate if the input sequences are paired or not. (**E**) Parameters to indicate the nomenclature of the forward and reverse files.

Materials

Software

A simple text editor such as Vim, Nano, or Emacs
A copy of the ASGARD+ repository

1. Locate the `ASGARD+` directory.
2. Open the `config` directory. Three files should be present in this path: `schema.json`, `asgard.ajson`, `saga.ajson`.
3. Open each configuration file with a text editor and locate in the first rows the constant object (Fig. 2) as well as the `input_path`, `output_path`, and `input_extension`.
4. Replace `./input_directory/` (Fig. 2, label A) with the path to your input files.
5. Replace `./output_directory/` (Fig. 2, label B) with the path to where the output file should be generated.

   A new directory within the specified route will be generated. The new directory name should be the same as the last part of the route with a timestamp and the name of the pipeline.

6. Edit the `input_extension` in case it is needed. By default it is set to `.fastq.gz` (Fig. 2, label C).

   Depending on the sequencing technology and the facility where this process is carried out, the way in which the Fastq format extension is designated may vary. This can change between `.fastq`, `.fq`, and, if the files are compressed, the extension `.fastq.gz` or `.fq.gz` can be added. ASGARD+ takes this extension as a constant to run the protocol, so it is necessary to specify it in the constants section of the configuration file.

7. Indicate if the WGS sequences files are paired-end (have a Forward file and Reverse file) by writhing "`true`" in the "`paired_input`" parameter or "`false`" if the data is composed of a single file per sample (Fig. 2, label E).

8. Indicate with which nomenclature the forward and reverse files are distinguished in the "`forward`" and "`reverse`" parameters, respectively, if the WGS files are paired (Fig. 2, label F).

   As well as the fastq extension, the way the forward and reverse files are distinguished in the file name may vary. Forward and reverse files are usually named using "`R1`" and "`R2`," or with numbers like "`1`" and "`2`." The user must indicate how to distinguish both files so that ASGARD+ can associate the paired files with each other correctly.

9. Edit the workers number (Fig. 2, label D) to reflect the number of logical processors (CPU cores). If this value is unknown, it can be queried using the following command.

```
$ nproc --all
```

BASIC PROTOCOL 3

ASGARD EXECUTION

The first protocol to be implemented on the ASGARD+ platform is ASGARD. This is designed for the identification of antimicrobial resistance genes with the ARIBA tool, facilitating the analysis of large Whole Genome Sequencing (WGS) datasets. ARIBA works by running local assemblies on paired sequencing reads and compares them against antimicrobial resistance gene reference sequences stored in various databases, such as ARG-ANNOT (Gupta et al., 2014), CARD (McArthur et al., 2013), MEGARes (Lakin et al., 2017), NCBI BioProject (Barrett et al., 2012), Plasmidfinder (Carattoli et al., 2014; Clausen, Aarestrup, & Lund, 2018), Resfinder (Zankari et al., 2012), VFDB (Chen, Zheng, Liu, Yang, & Jin, 2016), SRST2's version of ARG-ANNOT (Inouye et al., 2014), and VirulenceFinder (Clausen et al., 2018; Joensen et al., 2014; Malberg Tetzschner, Johnson, Johnston, Lund, & Scheutz, 2020). Users can employ any of these databases for analysis according to their criteria. ARIBA is the main program in this pipeline, but other functionalities are implemented directly on ASGARD+.

Materials

Hardware

A multicore processor is not required but highly advisable since the workload can be highly compute intensive
At least 4 GB of free RAM memory for the analysis; however, it can make use of more than 20 GB at a time
Enough storage space to store the output files; this can range from 1 GB to 150 GB depending on the number of input files
An internet connection of at least 10 Mbps to download the databases

Software

A copy of the ASGARD+ repository with the configuration files edited with the updated paths
The Python environment setup as in the Basic Protocol 1
A text editor program

1. Edit the specific properties for this protocol in the asgard.ajson configuration file:
   i. `ariba_database`: The valid options include argannot, card, ncbi, megares, plasmidfinder, resfinder, srst2_argannot, vfdb_core, vfdb_full, virulencefinder.
2. Go to the ASGARD+ directory and run the program by executing the following command:

```
$ python asgard-plus.py single --config_file=./config/asgard.ajson
```

RESULTS VISUALIZATION WITH PHANDANGO

When executing the ASGARD protocol, results on the presence and absence of antimicrobial resistance genes are generated, and two files are created that can be used in the Phandango tool. This interactive web platform allows the visualization of large-scale genomics datasets through phylogenetic trees accompanied by different types of metadata such as recombination blocks, pan-genome contents, or GWAS results (Hadfield et al., 2018). For this protocol, summarizing complex data is essential in order to facilitate their exploration and obtain accurate conclusions about the behavior of bacterial populations.

Materials

Hardware

Computer or computational cluster with access to the command-line environment and internet connection

Files

Files `ariba_summary.phandango.csv` and `ariba_summary.phandango.tre` generated in the execution of the ASGARD protocol

1. Go to the Phandango website at *https://jameshadfield.github.io/phandango/#/*.
2. Drag and drop the `ariba_summary.phandango.csv` and `ariba_summary.phandango.tre` files on the web page

   Once the files are uploaded to Phandango, the phylogenetic tree produced by ARIBA will be displayed and will show the result of the presence or absence of the AMR genes (top) using color codes (right), where green means "present" and red means "absent" (Fig. 3). From this image, the user can create and include metadata files that generate a more complex visualization. Phandango also provides multiple examples in its documentation to improve the visualizations by adding additional metadata files such as genome annotation, genetic recombination data, or pan-genome data.

3. Select the settings tab at the top of the website to edit the chart configuration and control the metadata displayed to meet your needs (Fig. 4).
4. Save the visualization result by pressing "p" (Fig. 3)

   Once the user has finalized the visualization settings, the image can be exported in SVG format, which can be opened in any image editing program.

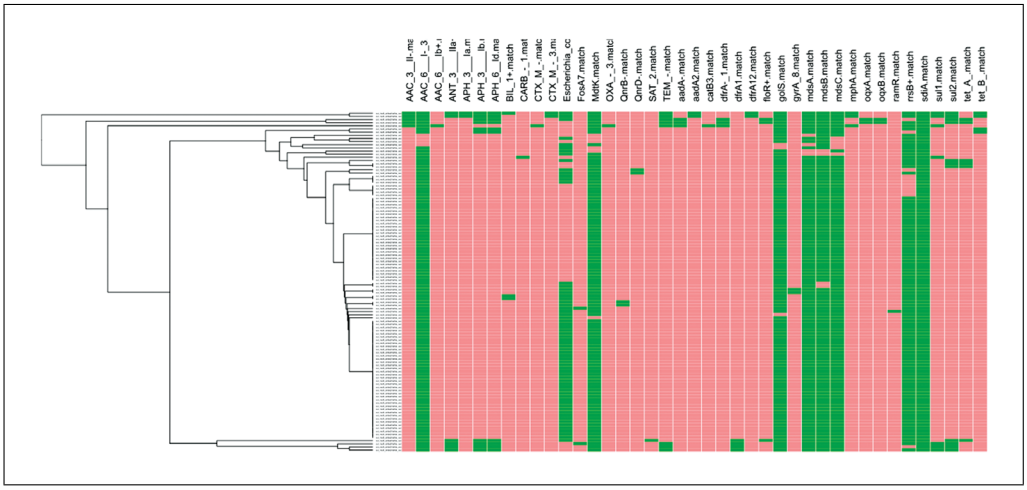

Figure 3 Visualization produced by Phandango. In this case this image was produced comparing 93 sample genomes of *Salmonella spp*, and the antimicrobial resistance genes on the CARD database.

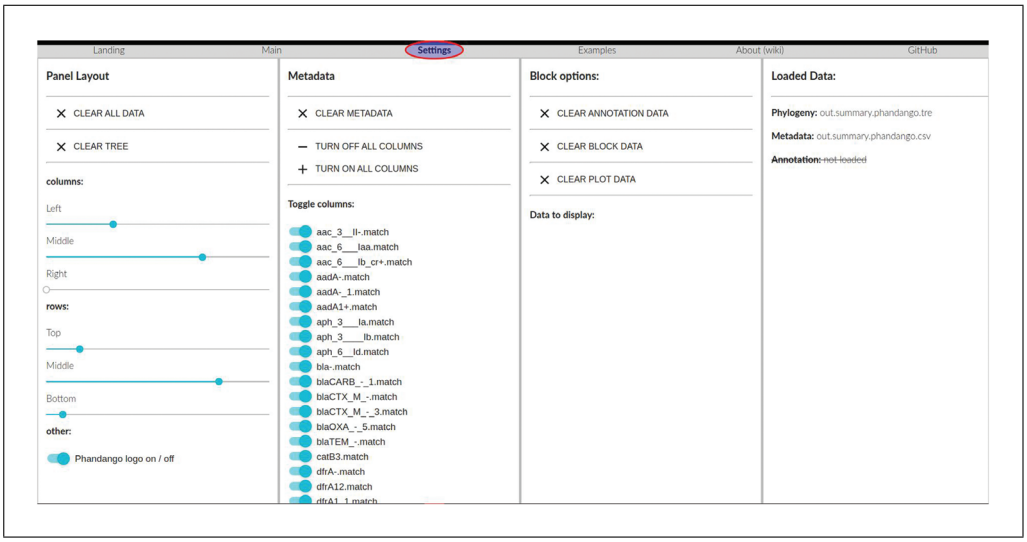

Figure 4 Settings on Phandango visualization. These options control the size of the different graphics represented in the visualization, and control the metadata displayed.

BASIC PROTOCOL 4

SAGA EXECUTION

The second protocol of the ASGARD+ platform is SAGA, which implements a pipeline for the identification of single nucleotide polymorphisms (SNPs) in bacterial genomes using a reference genome, so that genetic variability between different samples can be estimated. The pipeline allows entry of the accession number of the reference genome and accessory reference genomes from the NCBI Entrez Database and retrieval system (Barrett et al., 2012). The BWA tool (Li & Durbin, 2009) subsequently performs an alignment between each of the genomes of the samples of interest and the reference. Each resulting alignment is sorted and indexed by using Samtools (Li et al., 2009), which has a series of utilities to manipulate alignments in different formats (SAM, BAM, and CRAM). Afterwards, BCFtools (Li, 2011) takes the result of the alignments and, through a series of utilities, calls SNP variants and manipulates the files produced in VCF format. The SAGA pipeline continues to make a consensus sequence of the SNPs found in the different samples with respect to their position in the reference genome, and uses the SNP-sites tool (Page et al., 2016) to efficiently extract the SNPs from the multiple alignment. Finally, the alignment produced with the SNPs is used to build a phylogenetic tree with the RAxML tool (Kozlov, Darriba, Flouri, Morel, & Stamatakis, 2019) in order to represent the genetic relationship among the different samples. This tree of SNPs is the final result of this protocol, and can be visualized with the Figtree tool (*http://tree.bio.ed.ac.uk/software/figtree/*).

Materials

Hardware

Computer or computational cluster with access to the command-line environment and internet connection

Software

ASGARD+ and its dependencies installed
WGS data in Fastq format
Configuration file `saga.ajson` (included when downloading the ASGARD+ repository)
Figtree visualization program installed on its corresponding distribution

1. Edit the specific properties for this protocol in the `saga.ajson` configuration file:

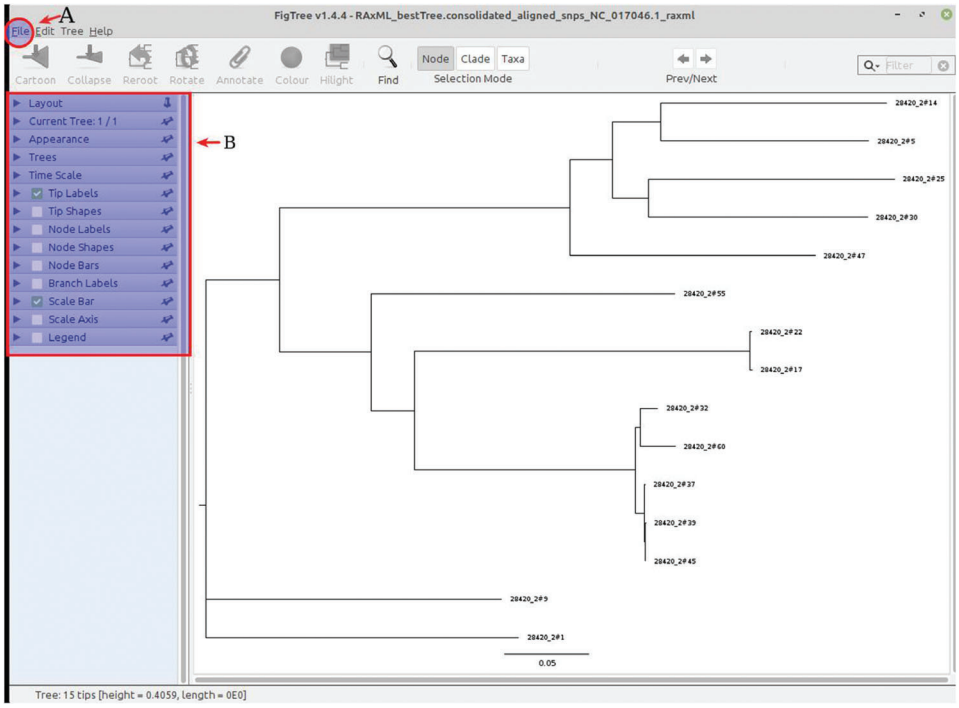

Figure 5 Figtree visualization tool for displaying the SNPs tree. The program has a top panel for general management of the tool and the visualization. (A) The File button displays the options for importing and exporting the results. (B) The control panel at the left of the screen allows the user to manipulate the general appearance of the tree. In the center of the interface is the visualization of the tree.

i. `reference_accession`: This corresponds to the reference genome accession number from the NCBI database.
ii. `accessory_accession`: This corresponds to the accessory reference genome accession number from the NCBI database. This is an optional parameter and can be removed if it is not needed.

2. Go to the ASGARD+ directory and run the program by executing the following command:

```
$ python asgard-plus.py single --config_file=./config/asgard.ajson
```

Upon executing this command, the full SAGA workflow will run. The final files produced by RAxML will be needed in further steps.

3. Visualize SNP tree on figtree.

a. Install the Figtree program from *http://tree.bio.ed.ac.uk/software/figtree/* according to the user's operating system.
b. Open Figtree interactive interface.
c. Go to the "File" button, and open the `RAxML_bestTree.consolidated_aligned_snps_ACCESION_raxml` file produced from the execution of the SAGA protocol (Fig. 5, label A).

The name of the file containing the tree varies for each execution of the protocol, so that in the name `RAxML_bestTree.consolidated_aligned_snps_ACCESION_raxml`, the word `ACCESSION` must appear with the identifier or accession number of the reference genome used.

d. Use the options panel to the right of the page (Fig. 5, label B) to modify or edit the appearance of the tree, including the layout, the tip, branch and node labels, and the rooting group.

e. Export the visualization of the tree by clicking the “File” button (Fig. 5A) and the Export button according to the desired format.

ALTERNATE PROTOCOL 1

CONTAINER INSTALLATION

One way that software development has solved installation and configuration problems has been by using containers. Containers can be viewed as a "standardized unit" of computing. This tool enables the creation of easily replicable configuration files to ensure the correct operation of a "software stack" across multiple platforms and infrastructures (Kadri, Sboner, Sigaras, & Roy, 2022). For ASGARD+, Docker was used as the main tool for the creation of these containers. This allows a simplified and standardized configuration and implementation. In order to follow the new industry standards in software development, and in an effort to simplify the setup of the ASGARD+ environment, a Dockerfile with an automated Basic Protocol 1 can be found in the same repository. Docker is an open-source platform, so it is free to use and very reliable. This protocol seeks to guide the installation and configuration process of Docker and its subsystem on Linux operating systems, specifically on Debian-based distributions.

Materials

Hardware

A multicore processor is not required, but highly advisable since the workload can be highly compute intensive
A computer with a 64-bit microprocessor; an x86 architecture is recommended
If executed on a computer with a Windows operating system, virtualization capabilities are required
At least 2 GB of free disk space to store the container image
4 GB of RAM for the Docker engine daemon
An internet connection of at least 10 Mbps

Software

A Debian-based operating system
Administrator access to the operating system

1. Update the apt repository to get the latest version of the required software:
   i. `$ sudo apt-get update`
   ii. `$ sudo apt-get install ca-certificates curl gnupg lsb-release`
2. Execute the next command to get the GPG key to ensure the integrity of the Docker software.

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
  dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

3. Install the Docker engine and the command-line tool to manage the containers using the next command:

```
$ sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli
  containerd.io
```

4. Run the test program to ensure it was installed correctly.

```
$ sudo docker run hello-world
```

A “Hello world” message should be printed in your screen

5. To be able to run Docker containers as a non-root user some extra steps are needed, if this is the case execute the following commands:
   i. `$ sudo groupadd docker`

This could output the following message "`groupadd: group 'docker' already exists`" which is acceptable.

ii. `$ sudo usermod –aG docker $USER`

iii. `$ sudo service docker restart`

6. The user should now be able to run the test without the use of root privileges. Re-Run the test program to make sure that the user can run it.

```
$ docker run hello-world
```

A "Hello world" message should be printed in your screen.

RUN ASGARD AND SAGA IN CONTAINER

ALTERNATE PROTOCOL 2

If the desired method to run ASGARD+ is by using containers, a small variation in the execution procedure should be made. The modifications in the configuration file are the same as in Basic Protocol 2, Basic Protocol 3, and Basic Protocol 4, but the Alternate Protocol 1 must be executed successfully to continue with this step. This protocol will take the Dockerfile and create an image of the environment that will build an easily reproducible environment in which ASGARD+ will run.

Materials

Hardware

A multicore processor is not , but highly advisable since the workload can be highly compute intensive
At least 4 GB of free RAM memory for the analysis; however, it can make use of more than 20 GB at a time.
Enough storage space to store the output files; this can range from 1 GB to 150 GB depending on the number of input files
If executed on a computer with a Windows operating system, virtualization capabilities are required
An internet connection of at least 10 Mbps to download the databases

Software

A copy of the ASGARD+ repository with the configuration files edited with the updated paths
The Python environment setup as in the Basic Protocol 1
A simple text editor
Docker engine with user permissions

1. Move to the ASGARD+ directory.
2. Create the image using the following command.

```
$ docker build. -t asgard-plus
```

3. Locate the path to the input files and replace the "##" symbols with the path to the input files.
4. To run the ASGARD pipeline, run the following command:

```
$ docker run -v /####/####/####/:/asgard/target -it asgard-plus:latest
   python asgard-plus.py single --config_file=config/asgard.ajson
```

5. To run the saga pipeline run the following command:

```
$ docker run -v /####/####/####/:/asgard/target -it asgard-plus:latest
   python asgard-plus.py single --config_file=config/saga.ajson
```

The output files will be in a new directory created in the same path as the input files.

COMMENTARY

Background Information

One of the biggest challenges in public health surveillance today is antibiotic resistance in bacteria responsible for infectious diseases. This problem has been identified by the World Health Organization as a global healthcare threat, and it is predicted that it could cause at least 10 million deaths per year by 2050 (Barreiro & Barredo, 2021; Ding & Yashuang, 2021; Shelenkov, 2021). The problem is aggravated due to the misuse and excessive use of broad-spectrum antibiotics not only in the clinical field but also in husbandry, which is responsible for at least half of the use of antibiotics (He et al., 2020; Van Boeckel et al., 2017). Unfortunately, the absorption of these substances in cattle is low, so around 30% to 90% is excreted into the environment through feces and urine (Sarmah, Meyer, & Boxall, 2006; Wang, Dong, Yang, Toor, & Zhang, 2013; Yue et al., 2021). Residual antibiotics that are dispersed in the environment contribute to the health problem by increasing the amount of bacteria that acquire antibiotic resistance genes, and therefore the selective pressure that accelerates the development of resistance mechanisms (Gilchrist et al., 2015; Ruan, Yu, & Feng, 2020; Yue et al., 2021). These molecular mechanisms for tolerating antimicrobial drugs vary within and across bacterial species and spread among themselves rapidly, causing the development of resistance in many ways (Van Camp et al., 2020). The constant genetic variation in the different strains makes bacterial outbreak surveillance difficult, and although traditional techniques such as PCR-based genotypic tests, multi-locus sequence typing (MLST), and amplification fragment length polymorphism (AFLP) have provided robust results, they do not provide information on the mechanisms of antibiotic resistance acquisition and spreading among microbial communities (Carriço, Sabat, Friedrich, Ramirez, & ESCMID Study Group for Epidemiological Markers (ESGEM) 2013; Coolen et al., 2021). Nowadays, the use of genomic epidemiology through the application of whole-genome sequencing (WGS) together with bioinformatics data analysis has facilitated the study of antimicrobial resistance on a large scale, allowing accurate, rapid, and cost-effective genotyping of bacterial isolates and outbreak tracking (Shelenkov, 2021; Tümmler, 2020). In response to the increased use of next-generation sequencing technologies, a large amount of WGS data has been generated, which in turn has led to the creation of public databases for microbiological typing where the information collected from epidemiological studies from all over the world is concentrated, accelerating the comparison and tracking of the different variants of species with resistance to antibiotics (Carriço et al., 2013). In response to the great availability of data, it has become necessary to create and standardize software platforms for the comprehensive analysis of this data. ASGARD+ was created from the need of a standardized protocol for the analysis of resistance genes in bacteria and the construction of phylogenies based on SNPs for the detection of pathogens of epidemiological significance. This platform allows analyzing large data sets of WGS reads in an iterative manner, enabling efficient and opportune data processing and providing researchers who have little experience in the use of bioinformatic protocols with the ease of generating results efficiently that can be visualized on platforms, such as Phandango, which can generate descriptive images of bacteria of epidemiological interest and its associated metadata in a way that facilitates the interpretation of data and decision-making in public health surveillance.

Critical Parameters

Before executing any workflow using ASGARD+, it is necessary to carry out a preliminary identification and analysis of the required software tools. To obtain reliable and repeatable results, it is essential to establish a stable environment so that both the computer configuration and software versions used are standardized in each one of the executions. The work of creating these environments is outside the focus of the ASGARD software, and these details need to be planned in advance. An alternative solution for this problem is to use containers (Docker, LXD, Singularity, etc.) so that there is a normalized environment in each execution of the workflow.

Troubleshooting

The common errors found when executing the protocols of the ASGARD+ tool and their possible solutions are listed in Table 1. It is suggested to use this as a guide in case of errors in the execution, or to contact the corresponding author.

Understanding Results

ASGARD makes use of the ARIBA tool to generate a summary of the antimicrobial

Table 1 Troubleshooting Guide for Installation and Execution of ASGARD+ Platform

Problem	Possible cause	Solution
The pipeline gets stuck in the first minutes of execution	The network does not provide access to the internet	Check the status of the network to which the computer equipment being used is connected
Missing output files after ASGARD+ protocols are finished	Possible errors with data or program installation	Review the `errors.txt` file inside the output directory in which possible program failures are recorded. Once the problem is identified, make the necessary adjustments, either in the input data or in the information required in the configuration files
No output from pipelines.	The running environment was not set up correctly	Make sure the setup scripts were executed successfully.
Unable to download files from ARIBA databases	ARIBA is not being actively maintained, so many functionalities break without notice.	Check the availability of these files from the official ARIBA git and submit a support ticket to the github. Currently some community members are providing quick fixes to the tool.
Message "`Found conflicts! Looking for incompatible packages.`" from the conda installation script	Ariba and bowtie2 have very specific dependency requirements	Verify the installation scripts are being run in a conda environment with the python version set to 3.6
The program outputs "`jsonschema.exceptions.ValidationError:`"	This is due to an ill-formatted configuration file	Check the ASGARD+ repository to review the required properties of the AJSON configuration file
The execution time greatly exceeds the estimated time	The configuration file is not configured to utilize all the resources of the platform	Check the number of logical cores present in the platform and edit the workers properties in the configuration file.
Error message "`FileNotFoundError: [Errno No such file]`"	The configuration file is not present in the selected path	Find the ASJON file inside of the ASGARD+ config directory
Multiple errors related to libgccng or glibc	Not all the required software are installed in the OS	If using Linux distributions other than Ubuntu or Debian, search the packet equivalent to the ones in the `debian-dependecies.sh script`

resistance genes of all the analyzed samples. This protocol generates three files. The first is `ariba_summary.csv`, containing a table with the results of the resistance genes found in the samples, according to the database chosen by the user. The file contains one column for each cluster and one row for each sample, which may contain "yes" or "no" depending on the presence or absence of the cluster in the sample. This file can be viewed in any spreadsheet program. The other two files, `ariba_summary.phandango.csv` and `ariba_summary.phandango.tre.`, contain the results of the analysis to be displayed on the Phandango tool. These files can be dragged and dropped on the online platforms. The resulting visualization allows the user to observe a phylogenetic tree with the genomes of the different samples or isolates, and the presence of AMR genes identified by ARIBA.

On the other hand, the SAGA protocol produces as a first result the file `accession.output_file`, which contains the indexed reference genome(s). Next, an alignment between the analyzed sequences and the reference genomes is produced in a SAM file that is later manipulated with the samtools tool to change the format to BAM, which corresponds to a smaller binary file, making

it easier for the software to manipulate the data efficiently. The BCFtools tool takes the alignment in `.bam` format and calls SNPs, producing a series of intermediate files until a VCF format file is obtained. This is a text file that contains a section with metadata and then a section separated by columns that include the different variants between nucleotides and their positions within the genome compared to the reference sequences. SAGA then consolidates the SNPs called for each of the samples, along with the references, into a multifasta file. This file is used as input by the SNP-sites tool, which locates the position of the variants and produces a new multiple alignment with the SNPs. This alignment is used by RAxML to generate a phylogenetic tree, and as a result various intermediate files are obtained that are necessary to obtain the best-scoring maximum likelihood tree, which is found in the file called `RAxML_bestTree.consolidated_aligned_snps_ACCESION_raxml`, where the word `ACCESSION` is replaced by the accession number of the reference genome. This file is used as input in the Figtree visualization interface, from which a final version of the tree can be exported in formats such as PDF, SVG, JPEG, or PNG.

Time Considerations

One of the purposes of ASGARD+ platform is to speed up the process of analyzing big data sets of WGS. Since ASGARD+ makes use of multiple online resources that are provided by apt, conda, and pip, the installation and configuration of the environment can vary greatly depending on the download speed of the internet connection. The execution of both ASGARD and SAGA protocols may vary according to the number of sample sequences and the species analyzed. One detail to be considered is that ASGARD execution time is lineal, while SAGA execution time is exponential. As an example, using a sample size of 93 WGS paired end samples of *Salmonella* spp., the execution of ASGARD takes 22 min, while SAGA takes 140 min on a server with an Intel Xeon E-2286G, which has 12 cores running at 4 Ghz with 32 Gb of DDR3 memory at 3200 Mhz.

Acknowledgments

The authors thank the National Center for High Technology (CeNAT), since the development of ASGARD+ was carried out on the Kabré supercomputer, located in this research center. The financing of this project was possible thanks to the System Funds (FEES) granted by the National Council of Rectors (CONARE) of Costa Rica.

Author Contributions

Maripaz Montero-Vargas: conceptualization, data curation, formal analysis, investigation, methodology, project administration, supervision, validation, visualization, writing original draft; **Alex Saenz-Rojas:** conceptualization, formal analysis, investigation, methodology, software, validation, writing original draft; **Marcela Suárez-Esquivel:** data curation, formal analysis, funding acquisition, investigation, methodology, supervision, validation, writing review and editing; **Lizbeth Ramirez-Carvajal:** data curation, formal analysis, funding acquisition, investigation, methodology, supervision, validation, writing review and editing.

Conflict of Interest

All authors declare that they have no conflicts of interest.

Data Availability Statement

The code that supports the protocol are openly available in Gitlab for downloading at *https://gitlab.com/CNCA_CeNAT/asgard*.

Literature Cited

Barreiro, C., & Barredo, J.-L. (2021). Worldwide clinical demand for antibiotics: Is it a real countdown? In E.C. Barreiro & J.-L. Barredo (Eds.), *Antimicrobial therapies: Methods and protocols* (pp. 3–15). Springer US. doi: 10.1007/978-1-0716-1358-0_1

Barrett, T., Clark, K., Gevorgyan, R., Gorelenkov, V., Gribov, E., Karsch-Mizrachi, I., … Ostell, J. (2012). BioProject and BioSample databases at NCBI: Facilitating capture and organization of metadata. *Nucleic Acids Research*, *40*(D1), D57–D63. doi: 10.1093/nar/gkr1163

Carattoli, A., Zankari, E., García-Fernández, A., Voldby Larsen, M., Lund, O., Villa, L., … Hasman, H. (2014). In silico detection and typing of plasmids using plasmidfinder and plasmid multilocus sequence typing. *Antimicrobial Agents and Chemotherapy*, *58*(7), 3895–3903. doi: 10.1128/AAC.02412-14

Carriço, J. A., Rossi, M., Moran-Gilad, J., Van Domselaar, G., & Ramirez, M. (2018). A primer on microbial bioinformatics for nonbioinformaticians. *Clinical Microbiology and Infection*, *24*(4), 342–349. doi: 10.1016/j.cmi.2017.12.015

Carriço, J. A., Sabat, A. J., Friedrich, A. W., Ramirez, M., & ESCMID Study Group for Epidemiological Markers (ESGEM). (2013). Bioinformatics in bacterial molecular epidemiology and public health: Databases, tools and the next-generation sequencing revolution.

Eurosurveillance, *18*(4), 20382. doi: 10.2807/ese.18.04.20382-en

Chen, L., Zheng, D., Liu, B., Yang, J., & Jin, Q. (2016). VFDB 2016: Hierarchical and refined dataset for big data analysis—10 years on. *Nucleic Acids Research*, *44*(D1), D694–697. doi: 10.1093/nar/gkv1239

Clausen, P. T. L. C., Aarestrup, F. M., & Lund, O. (2018). Rapid and precise alignment of raw reads against redundant databases with KMA. *BMC Bioinformatics*, *19*(1), 307. doi: 10.1186/s12859-018-2336-6

Coolen, J. P. M., Jamin, C., Savelkoul, P. H. M., Rossen, J. W. A., Wertheim, H. F. L., Matamoros, S. P., & van Alphen, L. B. (2021). Centre-specific bacterial pathogen typing affects infection-control decision making. *Microbial Genomics*, *7*(8), 000612. doi: 10.1099/mgen.0.000612

Ding, Z., & Ya-shuang, Z. (2021). Applications of bioinformatics in molecular epidemiology. *Chinese Journal of Disease Control & Prevention*, *25*(1), 20–24. doi: 10.16462/j.cnki.zhjbkz.2021.01.005

Gilchrist, C. A., Turner, S. D., Riley, M. F., Petri, W. A., & Hewlett, E. L. (2015). Whole-genome sequencing in outbreak analysis. *Clinical Microbiology Reviews*, *28*(3), 541–563. doi: 10.1128/CMR.00075-13

Gupta, S. K., Padmanabhan, B. R., Diene, S. M., Lopez-Rojas, R., Kempf, M., Landraud, L., & Rolain, J.-M. (2014). ARG-ANNOT, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes. *Antimicrobial Agents and Chemotherapy*, *58*(1), 212–220. doi: 10.1128/AAC.01310-13

Hadfield, J., Croucher, N. J., Goater, R. J., Abudahab, K., Aanensen, D. M., & Harris, S. R. (2018). Phandango: An interactive viewer for bacterial population genomics. *Bioinformatics*, *34*(2), 292–293. doi: 10.1093/bioinformatics/btx610

He, Y., Yuan, Q., Mathieu, J., Stadler, L., Senehi, N., Sun, R., & Alvarez, P. J. J. (2020). Antibiotic resistance genes from livestock waste: Occurrence, dissemination, and treatment. *NPJ Clean Water*, *3*(1), 1–11. doi: 10.1038/s41545-020-0051-0

Hunt, M., Mather, A. E., Sánchez-Busó, L., Page, A. J., Parkhill, J., Keane, J. A., & Harris, S. R. (2017). ARIBA: Rapid antimicrobial resistance genotyping directly from sequencing reads. *Microbial Genomics*, *3*(10), e000131. doi: 10.1099/mgen.0.000131

Inouye, M., Dashnow, H., Raven, L.-A., Schultz, M. B., Pope, B. J., Tomita, T., … Holt, K. E. (2014). SRST2: Rapid genomic surveillance for public health and hospital microbiology labs. *Genome Medicine*, *6*(11), 90. doi: 10.1186/s13073-014-0090-6

Joensen, K. G., Scheutz, F., Lund, O., Hasman, H., Kaas, R. S., Nielsen, E. M., & Aarestrup, F. M. (2014). Real-time whole-genome sequencing for routine typing, surveillance, and outbreak detection of verotoxigenic Escherichia coli. *Journal of Clinical Microbiology*, *52*(5), 1501–1510. doi: 10.1128/JCM.03617-13

Kadri, S., Sboner, A., Sigaras, A., & Roy, S. (2022). Containers in bioinformatics: Applications, practical considerations, and best practices in molecular pathology. *The Journal of Molecular Diagnostics*, *24*(5), 442–454. doi: 10.1016/j.jmoldx.2022.01.006

Kozlov, A. M., Darriba, D., Flouri, T., Morel, B., & Stamatakis, A. (2019). RAxML-NG: A fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, *35*(21), 4453–4455. doi: 10.1093/bioinformatics/btz305

Lakin, S. M., Dean, C., Noyes, N. R., Dettenwanger, A., Ross, A. S., Doster, E., … Boucher, C. (2017). MEGARes: An antimicrobial resistance database for high throughput sequencing. *Nucleic Acids Research*, *45*(D1), D574–D580. doi: 10.1093/nar/gkw1009

Li, H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, *27*(21), 2987–2993. doi: 10.1093/bioinformatics/btr509

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, *25*(14), 1754–1760. doi: 10.1093/bioinformatics/btp324

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., … 1000 Genome Project Data Processing Subgroup. (2009). The sequence alignment/map format and SAMtools. *Bioinformatics (Oxford, England)*, *25*(16), 2078–2079. doi: 10.1093/bioinformatics/btp352

Mäklin, T., Kallonen, T., Alanko, J., Samuelsen, Ø., Hegstad, K., Mäkinen, V., … Honkela, A. (2021). Bacterial genomic epidemiology with mixed samples. *Microbial Genomics*, *7*(11), 000691. doi: 10.1099/mgen.0.000691

Malberg Tetzschner, A. M., Johnson, J. R., Johnston, B. D., Lund, O., & Scheutz, F. (2020). In silico genotyping of Escherichia coli isolates for extraintestinal virulence genes by use of whole-genome sequencing data. *Journal of Clinical Microbiology*, *58*(10), e01269–20. doi: 10.1128/JCM.01269-20

McArthur, A. G., Waglechner, N., Nizam, F., Yan, A., Azad, M. A., Baylay, A. J., … Wright, G. D. (2013). The Comprehensive Antibiotic Resistance Database. *Antimicrobial Agents and Chemotherapy*, *57*(7), 3348–3357. doi: 10.1128/AAC.00419-13

Page, A. J., Taylor, B., Delaney, A. J., Soares, J., Seemann, T., Keane, J. A., & Harris, S. R. Y. (2016). SNP-sites: Rapid efficient extraction of SNPs from multi-FASTA alignments. *Microbial Genomics*, *2*(4), e000056. doi: 10.1099/mgen.0.000056

Ruan, Z., Yu, Y., & Feng, Y. (2020). The global dissemination of bacterial infections necessitates the study of reverse genomic

epidemiology. *Briefings in Bioinformatics*, *21*(2), 741–750. doi: 10.1093/bib/bbz010

Sarmah, A. K., Meyer, M. T., & Boxall, A. B. A. (2006). A global perspective on the use, sales, exposure pathways, occurrence, fate and effects of veterinary antibiotics (VAs) in the environment. *Chemosphere*, *65*(5), 725–759. doi: 10.1016/j.chemosphere.2006.03.026

Shelenkov, A. (2021). Whole-genome sequencing of pathogenic bacteria—new insights into antibiotic resistance spreading. *Microorganisms*, *9*(12), 2624. doi: 10.3390/microorganisms9122624

Simar, S. R., Hanson, B. M., & Arias, C. A. (2021). Techniques in bacterial strain typing: Past, present, and future. *Current Opinion in Infectious Diseases*, *34*(4), 339–345. doi: 10.1097/QCO.0000000000000743

Tümmler, B. (2020). Molecular epidemiology in current times. *Environmental Microbiology*, *22*(12), 4909–4918. doi: 10.1111/1462-2920.15238

Van Boeckel, T. P., Glennon, E. E., Chen, D., Gilbert, M., Robinson, T. P., Grenfell, B. T., … Laxminarayan, R. (2017). Reducing antimicrobial use in food animals. *Science*, *357*(6358), 1350–1352. doi: 10.1126/science.aao1495

Van Camp, P.-J., Haslam, D. B., & Porollo, A. (2020). Bioinformatics approaches to the understanding of molecular mechanisms in antimicrobial resistance. *International Journal of Molecular Sciences*, *21*(4), 1363. doi: 10.3390/ijms21041363

Wang, H., Dong, Y., Yang, Y., Toor, G. S., & Zhang, X. (2013). Changes in heavy metal contents in animal feeds and manures in an intensive animal production region of China. *Journal of Environmental Sciences*, *25*(12), 2435–2442. doi: 10.1016/S1001-0742(13)60473-8

Yue, Z., Zhang, J., Zhou, Z., Ding, C., Wan, L., Liu, J., … Wang, X. (2021). Pollution characteristics of livestock faeces and the key driver of the spread of antibiotic resistance genes. *Journal of Hazardous Materials*, *409*, 124957. doi: 10.1016/j.jhazmat.2020.124957

Zankari, E., Hasman, H., Cosentino, S., Vestergaard, M., Rasmussen, S., Lund, O., … Larsen, M. V. (2012). Identification of acquired antimicrobial resistance genes. *The Journal of Antimicrobial Chemotherapy*, *67*(11), 2640–2644. doi: 10.1093/jac/dks261

Internet Resources

https://gitlab.com/CNCA_CeNAT/asgard

Gitlab repository and documentation for ASGARD+.

https://jameshadfield.github.io/phandango/#/

Phandango web platform.

http://tree.bio.ed.ac.uk/software/figtree/

Figtree download.