NGSEP plugin manual

Daniel Felipe Cruz  d.f.cruz@cgiar.org
Juan Fernando De la Hoz  j.delahoz@cgiar.org
Claudia Samantha Perea  c.s.perea@cgiar.org
Juan Camilo Quintero  j.c.quintero@cgiar.org
Jorge Duitama  j.duitama@cgiar.org
# Contents

NGSEP PLUGIN MANUAL .......................................................................................................................... 0

INTRODUCTION ........................................................................................................................................... 2

SYSTEM REQUIREMENTS ............................................................................................................................. 3
  COMMAND LINE ................................................................................................................................. 3
  DOWNLOADING AND INSTALLING JAVA ................................................................................................. 3
  DOWNLOADING AND INSTALLING ECLIPSE IDE .................................................................................. 4
  INCREASING ECLIPSE MEMORY ........................................................................................................... 9
    Increasing Eclipse memory in PC ........................................................................................................... 9
    Increasing Eclipse memory in Mac OS ................................................................................................. 12

NGSEP DOWNLOAD, INSTALLATION AND USE ..................................................................................... 15
  NGSEP UPDATE ....................................................................................................................................... 16
  CREATING A NEW PROJECT .................................................................................................................... 16
    Upload Files ........................................................................................................................................... 18
  TRACKING PROCESS PROGRESS ......................................................................................................... 21
    Enable NGSEP Progress Bar .................................................................................................................. 21
    Error and process tracking ................................................................................................................... 23

MAPPING READS .......................................................................................................................................... 25
  INSTALLING BOWTIE2 ........................................................................................................................... 25
  INDEX OF THE REFERENCE WITH BOWTIE2 .......................................................................................... 29
  MAP READS ............................................................................................................................................ 32
  MAP READS PARAMETERS ....................................................................................................................... 34
    Main arguments ..................................................................................................................................... 34
    Alignment options ............................................................................................................................... 36
    Sorting options ..................................................................................................................................... 41
  MULTI-MAPPING ..................................................................................................................................... 43

VARIANTS DETECTOR .................................................................................................................................. 47
  VARIANT DETECTION PARAMETERS ....................................................................................................... 49
    Main arguments ..................................................................................................................................... 49
    SNV detection ....................................................................................................................................... 51
    SV detection ....................................................................................................................................... 53
  MULTI VARIANTS DETECTOR ................................................................................................................... 54

MERGE VCF .................................................................................................................................................. 59

VARIANTS FUNCTIONAL ANNOTATOR ..................................................................................................... 65

FILTER VCF FILES ..................................................................................................................................... 69

EXPORT VCF FILES TO OTHER FORMATS .............................................................................................. 75

ONE-STEP WIZARD, AUTOMATIC PIPELINE ............................................................................................. 77
Introduction

Next Generation Sequencing (NGS) technologies have increased exponentially the understanding of the genomic structure and function of different organisms within the last decade, including the CIAT mandate crops. In order to handle the vast amount of data produced by these technologies, several bioinformatics tools have been developed to carry out different kinds of analysis. However, most of these tools are not easy to operate, integrate and customize without the technical support of experts in bioinformatics, which produces a bottleneck for several research efforts. This situation sets the need for integrated data analysis pipelines with user friendly interfaces available to the scientific community.

We have developed NGSEP (NGSTools Eclipse Plugin), an integrated framework for variants discovery from NGS data. NGSEP is based on Eclipse which is one of the leading development environments for Java. We integrated previously developed algorithms for SNV detection available in the NGSTools package with Java implementations of state-of-the-art algorithms for CNV and structural variation discovery. NGSEP provides an intuitive interface in which the user has a rich control over the files produced during the different stages of the analysis. These files follow current standard formats such as BAM and VCF, which makes NGSEP results easy to integrate with genome visualization tools. NGSEP can also be integrated with bowtie2 to allow the user to follow all the steps needed to obtain genomic variants from raw reads without scripting. NGSEP is distributed as an open source project under General Public License (GPL) to make it available to the scientific community.
System Requirements

In order to install and execute NGSEP plugin properly you must have installed the following components:

- Operative system Windows, Macintosh or Linux.
- Java (jre jdk 1.6 or higher). See instructions for how to check your current version of java or download it and install it.
- Eclipse IDE 3.7 or higher. See below for instructions of how to download and install Eclipse.
- Bowtie2 is required only for the Map Reads function. See instructions for how to download and install Bowtie2 in the Map Reads section.
- For Windows, WinRar or WinZip.
- Text editor. For Windows we recommend notepad++. You can download it in the following link: http://notepad-plus-plus.org/

Command Line

You can also run NGSEP from the command line, downloading the NGSEP library: NGSToolsApp.jar from http://sourceforge.net/projects/ngsep/files/Library/. Look at the README file for instructions on how to run the command line version of NGSEP.

Downloading and Installing Java

NGSEP is written in Java and is therefore platform independent, but the Java Runtime Environment (JRE) version 1.6 or higher (also called Java 6) has to be installed. You can check the version of your JRE, and test if it is working, using the following link:

http://www.javatester.org/version.html

Downloading Java for Windows

Download and install the latest version of java from the following link: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Downloading Java for Mac

Apple Computers supply their version of Java. Use the Software Update feature, which is available on the Apple menu, to check that you have the most updated Java version for your Mac. Additionally, make
sure that Java version 1.6 is set as first preference version. This can be changed under "Applications - Utilities - Java Preferences.app". You can also download the Mac OS X x64 Option from http://www.oracle.com/technetwork/es/java/javase/downloads/jre7-downloads-1880261.html

**Downloading and Installing Eclipse IDE**

**For 64 bit operating system**

We offer the option to download the latest release of Eclipse IDE together with NGSEP plugin and a Bowtie2 auto installer for Windows. However, this option is only available for 64 bit operating systems.

Select the Eclipse+Plugin option, and then the adequate .zip file according to your operative system. Unzip the file clicking the option extract to your work folder.
In the extracted folder called Eclipse, you will find a folder called *dropins*. In this folder you will find the NGSEP plugin.

In addition, you will find the executable file called *eclipse(.exe)*. Double click in this icon to launch Eclipse. It will immediately ask you for a work folder called *workspace*. You can select the suggested one or assign a specific one. This will be your working folder in Eclipse, where all your projects are going to be created and stored. Now Eclipse is ready to be used.

**For 32 bit operating system**

If you are using a 32 bit operating system, download first the standard Eclipse ([http://www.Eclipse.org/downloads/](http://www.Eclipse.org/downloads/)) for 32 bit and afterwards download the NGSEP plugin ([https://sourceforge.net/projects/ngsep/files/OnlyPlugin/](https://sourceforge.net/projects/ngsep/files/OnlyPlugin/)). Follow these instructions on how to download and install the NGSEP plugin and Eclipse:

Select the *OnlyPlugin* option.
Download the compressed file from the download page of Eclipse organization: http://www.Eclipse.org/downloads/.

Select your desired version of Eclipse and choose the right file according to your operative system and your system architecture (32 or 64 bits).

Unzip the file clicking the option extract to your desired folder.
In the extracted folder you will find an executable file called *Eclipse.exe*: Once you click on that file, Eclipse will be launched and immediately it will ask you for a work folder called *workspace*. You can select the suggested one or assign a specific one, this will be your working folder in Eclipse, where all your projects are going to be created. Now Eclipse is ready to be used.
Eclipse will look for your Java Virtual Machine (JVM). If it is not recognized please follow the next directions:

Once installed, you must edit the PATH variables. In windows you can access them trough: MY PC – PROPERTIES – ADVANCED OPTIONS – ENVIRONMENT VARIABLES

Click on environment variables, search for PATH Variable and edit it adding a “ ; “ plus the path for the \bin folder from the JVM at the end of the line (where you can find the executable files of the JVM), for example:

“ ; C:\Program Files\Java\jre1.6.0_20\bin “

Restart your PC so that the change will be applied, and Java will be available for all the system and therefore for Eclipse.

Increasing Eclipse memory

It is highly recommended to increase the values of memory granted for Eclipse because NGSEP runs processes that are demanding. A lack of memory will produce exceptions in some functionalities or will suddenly close the program. The most common error that reflects this issue would be Exception in thread “main” java.lang.OutOfMemoryError: Java heap space.

Increasing Eclipse memory in PC

First, locate Eclipse folder and edit a file called Eclipse.ini, which looks like the following picture:
Note: Before editing this file, make sure that Eclipse is closed; otherwise these changes will not be applied.

Inside that file, you will find the lines -Xmx___m and -Xms___m.

The -Xmx___m line indicates how much maximum memory Eclipse is allowed to use. In this example we set Xmx3500m to allow Eclipse to use up to 3500 megabytes (3.5 gigabytes) of memory. You can set this parameter up to the total memory available in your computer, although it is recommended to leave about 1Gb for other applications.

The -Xms___m line indicates how much memory Eclipse will allocate to start up. In this example we set Xms1500m. It is recommended to set this parameter to at least 512 MB.

Save and close the file and launch Eclipse again.

Note1: You can check your RAM memory in Control Panel → System and Security → Systems.
Note: The Eclipse.ini file is often hidden in your Eclipse folder. To unhide this file, go to Control Panel → Appearance and Personalization → Folder Options → View tab and select the Show hidden files, folders and drives option. Also it is recommended to unselect the option Hide extensions from known file types to be able to recognize the files by their extension.
Increasing Eclipse memory in Mac OS

In order to be able to increase these values of memory, locate Eclipse folder and right click in the icon 🏠 Eclipse, next, click the option Show Package Contents option and then go to the Contents folder.

Open contents.
Here, select the MacOS folder and open the file Eclipse.ini in the text editor of your choice.
Note: Before editing this file, make sure that Eclipse is closed; otherwise these changes will not be applied.

Inside that file, you will find the lines \texttt{-Xmx\_\_m} and \texttt{-Xms\_\_m}.

The \texttt{-Xmx\_\_m} line indicates how much maximum memory Eclipse is allowed to use. In this example we set \texttt{Xmx3500m} to allow Eclipse to use up to 3500 megabytes (3.5 gigabytes) of memory. You can set this parameter up to the total memory available in your computer, although it is recommended to leave about 1Gb for other applications.

The \texttt{-Xms\_\_m} line indicates how much memory Eclipse will allocate to start up. In this example we set \texttt{Xms1500m}. It is recommended to set this parameter to at least 512 MB.

Save and close the file and launch Eclipse again.
NGSEP download, installation and use

For installation you first need to download the NGSEP plugin (http://sourceforge.net/projects/ngsep/files/OnlyPlugin/) and paste it in a folder called dropins in the Eclipse directory. If you downloaded the Eclipse+Plugin version for 64 bits, the plugin comes already in the dropins folder in Eclipse.
After restarting Eclipse again, the NGSEP plugin will be integrated with Eclipse IDE.

**NGSEP update**

To update NGSEP, download the latest version of the plugin from the web site (http://sourceforge.net/projects/ngsep/files/OnlyPlugin/). Erase the old version and replace it pasting the new one in the *dropin* folder in the Eclipse directory. Restart Eclipse and the new NGSEP plugin will now be integrated with Eclipse IDE.

**Creating a new project**

The first thing you need to do after starting Eclipse is to create a new project. To do this, go to the task bar at the upper part of Eclipse, and select: File → New → Project, and select General → Project. Immediately a window to name the project will show up, where you can type the name of your new project.
Upload Files

The standard input files could be in BAM, SAM, GFF, VCF, FASTA or FASTQ formats, depending of the kind of analysis you want to perform:

This are very big files so instead of copying them to the workspace through Eclipse, add the input files to the project by selecting them from their corresponding directories and then placing them directly into your *eclipse project using the Windows Explorer or the Finder in Mac*. And finally, refresh the whole project folder in the Eclipse *Package Explorer*. 
Once you have pasted the selected files, you can view them in Eclipse, on the Package Explorer inside your project as shown in the following image.

We suggest creating folders according to the type of data they contain, for example folders for the references, the raw reads etc., as follows:
Now NGSEP should be working in your Eclipse. If you right click any input file, for example the .bam, you will see several options, and you should be able to locate NGSEP menu among them. If you place the mouse cursor on it, you will see the bioinformatics options that NGSEP can execute, from map reads to variants detection and statistics plots.

⚠️ **Note:** After running any process, remember always to refresh your project folder in order to be able to see your output and .log files. Press F5 or right click your project folder and select the *Refresh* option.
Tip: There is an automatic way of refreshing the package explorer, to activate this option go to the menu bar in Eclipse and select: Window → Preferences → General → Workspace → Refresh using native hooks or polling option should be checked.

Tracking Process Progress

Enable NGSEP Progress Bar

Enabling the NGSEP view in Eclipse, will allow you to see the progress bar of the NGSEP tasks.

To enable the progress bar, go to the tools bar at the upper part of Eclipse and select the following options:

1. First click on Window option in the task bar.
2. Then click on the option Show view → Other.
3. Click on the folder *General* and choose *NGSEPView*.

4. You will see a new tab next to the console and problems log.
Note: This tab contains the progress bars of NGSEP. If you haven’t triggered any process you should not see anything there, however sometimes Eclipse uses that tab to report processes of projects and its environment. Do not worry if that happens. Now with the progress bar view activated you are ready to use the different options that NGSEP offers.

Note: you could find some differences among Eclipse versions.

Error and process tracking

To check if your processes in NGSEP were successfully completed, you can check the .log files, generated by each function. The name of the log file will have the prefix of the executed process, followed by the output name and the .log extension. You can check this file at any point of the process; it will contain information on the progress of your task, or information about any possible error.

You can find your .log files, in your project folder, directly in Eclipse.

Note: Remember to refresh your project folder frequently using F5 or by right clicking at your project folder and selecting the refresh option.
Another source of error, independent of NGSEP could be tracked by looking at Eclipse’s error log file. There are two ways to check this log file. First, you can enable the error view in Eclipse, selecting Window → Show View → Error log.

Second, you can find a .log file in a folder called .metadata, located in your workspace folder. You can open it with your default text editor and check for any possible errors. This could be useful anytime that the source of error doesn’t allow opening Eclipse.

⚠️ Note: Usually the .metadata folder is hidden. In this case, you should enable to look for hidden files and folders.
Mapping Reads

This process executes the alignment or mapping process between a reference genome and reads that come from sequencers such as Illumina and 454. This process can be performed for a single sample or for multiple samples (see Multi Map Reads).

REQUIREMENTS

- Bowtie2: Open source, “ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences”. It can align reads from 35pb to thousands of bp to a reference sequence.

Installing Bowtie2

The first step to use Map reads is downloading and installing bowtie2 in your PC. In NGSEP we provide a bowtie2 auto installer for Windows operating systems of 64 bits, which makes easier the installation process. You can find the auto installer in the folder Eclipse+Plugin in http://sourceforge.net/projects/ngsep/files/Eclipse%2BPlugin/. This auto installer was created using Advanced Installer version 10.6 http://www.advancedinstaller.com.

If you are using Windows, download the .zip file, unzip it and you will find a folder named “InstallBowtie2”. To install bowtie 2 using the auto installer, double click on the Bowtie2.msi link.
Welcome to the Bowtie2 Setup Wizard

The Setup Wizard will install Bowtie2 on your computer. Click "Next" to continue or "Cancel" to exit the Setup Wizard.

Ready to Install

The Setup Wizard is ready to begin the Bowtie2 installation.

Click "Install" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.
Follow the default options, until installation is complete.

To check whether bowtie2 was successfully installed, we recommend the following test:

Look for the **Command Prompt** in Windows, by typing “cmd” in the search bar.

Then type “bowtie2-align.exe ” in the command line (“bowtie2 ” alone for mac), and you should be able to see all the options available for bowtie2 as in the following screen:
If Bowtie2 was not successfully installed, then after this test you should have the following message:

![Image of command prompt showing error message]

For a manual installation of bowtie2, if for example you have a 32bits Windows operation system, or Mac OS, follow these instructions:

Download Bowtie2

1. Download Bowtie2 in the following link: [http://bowtie-bio.sourceforge.net/bowtie2/index.shtml](http://bowtie-bio.sourceforge.net/bowtie2/index.shtml), select the Latest Release and choose from the options the adequate one for your operation system. Unzip the folder.
2. Open the folder and find the executable: bowtie2-align.exe, bowtie2-build.exe.
3. Add bowtie2 to your PATH environment variable. To do this, follow your operating system’s instructions for adding the directory to your PATH.
   1. For Windows follow these steps: Right click on Computer and choose: Properties → Advanced System Settings → Advanced Options → System Variables → Path → Edit. In the option Variable Value, add a “;” (each semicolon adds a new path for variables) and write the path where your bowtie2 folder is. For example: ;C:\Users\jcqui02ntero\Desktop\CIAT\Bowtie2\bowtie2-2.1.0\n   2. For Mac follow these steps: Open a Terminal window, Look which directories are in your path: “echo $PATH”, Go the binaries directory: “cd /usr/bin/”, Copy the files from bowtie2: “cp ~/Downloads/bowtie2-x.x/bowtie* ./” You will probably be asked for super-user access, and your password: “sudo cp ~/Downloads/bowtie2-x.x/bowtie* ./”

To check whether bowtie2 was successfully installed or not, we recommend following the steps of the test above.

**Index of the reference with Bowtie2**

NGSEP provides an easy way to generate an index of your reference genome, using Bowtie2. To perform the Map Reads function you will need an index for the reference genome; otherwise you won’t be able to execute this task.

**INPUT FILE**
• **Reference_sequence.fa**: A reference genome in FASTA format.

**OUTPUT FILES**

• **Six .bt2 files**: these files contain the index of the input reference sequence in a format that bowtie2 will use to efficiently align the reads. The common prefix of these files should be set as input for the mapping process.

• **.log file**: information about the finished process.

First, you need to upload your reference file in FASTA format (.fa) in Eclipse. Right click the file and select the **Create Index Bowtie** function in the NGSEP Menu. A window will pop-up that by default will recognize and show your input file and the name and pathway of your output file, that will be the same as the reference (Index Bowtie2 Prefix). Here you can change the input reference file, the output prefix, or both, using the browsing option at the right. We recommend you to leave the same name for your reference as for your index.

To start the process, click the **Create Index** button.
At the end of the process you will have your reference file and other 7 files with the same prefix.

⚠️ **Note:** For further analyses, when you are asked for the **reference**, you’ll have to introduce the FASTA file with the reference genome. And, when you are asked for the **Index**, you’ll have to introduce the name (prefix) you gave to your Bowtie2 Index.

For more information about the indexing process in bowtie2, we recommend these links:

Map Reads

⚠️ Note: Through this entire manual we are going to use Illumina FASTQ formats as example input files. However, other file formats and sequencing platforms are also allowed, such as FASTA, qseq and SOLiD, 454, PacBio respectively.

⚠️ Note: in Mac or Linux, NGSEP is able to process gzip compressed fastq files, whereas in windows the fastq files need to be uncompressed.

INPUT FILES

- **Reads**: These are the files generated by the sequencing platform of your election with the sequence (and quality) information of your samples. These can be single-read or paired-end reads (in which case both files are named the same but with different numbers: 1 and 2).
- **Prefix of the bowtie2 index files**: This is the prefix of the files generated as output of the option Create Index Bowtie described above. If default values are kept while creating the index, this prefix should correspond exactly to the name of the fasta file with the reference genome.

OUTPUT FILES

- **Sorted .bam**: This binary file contains the alignment of each read to the reference genome. A detailed description of the format of a bam file and its corresponding plain text version (sam) can be found in the samtools website (http://samtools.sourceforge.net/). This output bam file already contains the alignments organized by chromosome and reference position. NGSEP allows you to have a preview of the first 100 alignments in SAM format. In MAC or linux this file can also be browsed in a command line environment using samtools.
- **bai**: Binary file needed to visualize the mapped reads in the Integrative Genomics Viewer (IGV)
- **.log file**: Information about the finished process.
- **Reference.projectNGSEP**: is an automatic-generated file that will be put in your projects directory, don’t move or delete it.

Once you have your reference genome indexed, you can execute the Map Reads function in NGSEP. To begin, you will need to have your read files uploaded in the Package Explorer in Eclipse. You can select a unique file if you are working with single reads or two files in case you are using paired-end reads. Select and right click the file(s) and choose the Map Reads function in the NGSEP menu.
A window will open for you to choose the adequate parameters for your analysis:
**Map Reads parameters**

⚠️ Note: when the fields are empty, the default parameters are set.

The parameters described below are summarized from the alignment parameters from Bowtie2. You can find more detailed information about them in Bowtie2 manual, at Bowtie2’s official page (http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml).

**Main arguments**

If you are using paired-end reads, **File #1** and **File #2** fields show the path of your input files. To change the order of your input files use the browser option. For single-reads just fill in the **File #1** field.

(*) **Output File Prefix (.bam)**: Enter the name and the path where you want to save your output file; the software will automatically create the .bam extension.

(*) **Bowtie Index**: Once you have indexed your reference genome (using the *Create Index Bowtie* option described above), write down the SAME PREFIX of the index, along with the whole path. Next time that you open this screen you will see the last file you uploaded.
With the **Insert size** option you can specify the minimum and maximum fragment length for valid paired-end alignments. E.g. if Min is 60 and max is 120 and a paired-end alignment consists of two 20-bp alignments in the proper orientation with a 60-bp gap between them, that alignment is considered valid. Alignments exceeding the limits won’t be considered valid, the alignment can be very efficient for typical fragment length ranges (200-400), but the larger the difference, the slower it will run. Default values are 0 and 500.

In **Read group Id** enter a tag that is going to be used for the further classification/grouping of your samples. It will appear in your bam file. If you want, you can also change the **Sample Id** for a new classification.

By default, bowtie2 searches for distinct, valid alignments for each read. When it finds a valid alignment, it continues looking for alignments that are nearly as good or better. The best alignment found is reported (randomly selected from among best if tied). If you choose **Numbers of Alignments to report**, a new field will appear where you can define the number of alignments you want to report.

Using this option, Bowtie 2 searches for as many valid alignments for each read as you specify in the “Number of Alignments to report” field. If, for example, you choose 2, Bowtie2 will search for at most 2 distinct alignments. It reports all alignments found, in descending order by alignment score. The alignment score for a paired-end alignment equals the sum of the alignment scores of the individual mates.
Setting a number of alignments to search for has its disadvantages; Bowtie 2 does not "find" alignments in any specific order, so for reads that have more than N distinct, valid alignments, Bowtie 2 does not guarantee that the N alignments reported are the best possible in terms of alignment score. Still, this mode can be effective and fast in situations where the user cares more about whether a read aligns (or aligns a certain number of times) than where exactly it originated from.

Reporting all alignments is mutually exclusive with a number of alignments to report. And when aligning to large, repetitive genomes, it makes the search very slow, and the output files very heavy.

Alignment options

!!Note: when the fields are empty, default parameters are set.
Select the **Input** box if you know the format of your input files, and choose the adequate option. If you don’t select the Input file format, the program will assume FASTQ format.

The different file formats are listed in the chart below. For a complete description we recommend the following link:


<table>
<thead>
<tr>
<th>Input File Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FASTA</strong></td>
<td>FASTA files usually have <code>.fa</code>, <code>.fasta</code>, <code>.mfa</code>, <code>.fna</code> extension or similar. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (&quot;&gt;&quot;) symbol in the first column. The word following the &quot;&gt;&quot;) symbol is the identifier of the sequence, and the rest of the line is the description (both are optional).</td>
</tr>
<tr>
<td><strong>FASTQ</strong></td>
<td>FASTQ files usually have extension <code>.fq</code> or <code>.fastq</code>. This is the default format. This format is similar to fasta consisting of readname headers, nucleotide base calls and per-base quality scores in text form. Illumina fastq contains the READNAME, index, read number parameters with quality basis character <code>@</code>. Paired end data are presented in the orientation in which they will be aligned to a reference (5’-3’-3’-5’), which is the same orientation they were sequenced in.</td>
</tr>
<tr>
<td><strong>Illumina qseq format:</strong></td>
<td>This format is similar to FASTQ but differs in how the quality scores are calculated. QSEQ files usually end in <code>_qseq.txt</code>.</td>
</tr>
<tr>
<td><strong>Raw one-sequence-per-line:</strong></td>
<td>Reads are files with one input sequence per line, without any further information (no read names, no qualities).</td>
</tr>
</tbody>
</table>

**Phred**

![Phred 64](Phred64.png)
Phred is the code or language in which sequencer quality values for each base are given. This option allows you to choose between the Phred+64 and Phred+33 encoding formats. If you don’t select the **phred 64** option, default will be **phred 33**.

**Trim**

| Trim 5': 0 | Trim 3': 0 |

With this option you can trim low quality bases from the 5' and/or 3' end of each read. If you don’t have any information about the quality of your reads you can follow these steps:

- Continue without trimming.
- Once you have your `.bam` file you will be able to **Calculate Quality Statistics** (see below).
- In case quality is too low in any of the ends, you can return to this step and trim your reads.

Trim5': Number of bases you want to remove from 5' (left) end of each read before alignment (default: 0).

Trim3': Number of bases you want to remove from 3' (right) end of each read before alignment (default: 0).

| Give up extending after: 15 | Maximum number of times will re-seed: 2 |

In the **Give up extending after** option you can choose the number of consecutive seed extension attempts that can "fail" before Bowtie 2 moves on, using the alignments found so far. A seed extension can fail if it does not yield a new best or a new second-best alignment. Default value is 15.

With the **Maximum number of times will re-seed** option you can choose the maximum number of times Bowtie2 will "re-seed" reads with **repetitive seeds**. Default value is 2. When "re-seeding," Bowtie2 simply chooses a new set of reads (same length, same number of mismatches allowed) at different offsets and searches for more alignments. A read is considered to have repetitive seeds if the total number of seed hits divided by the number of seeds that aligned at least once is greater than 300.
With the **Length of seed ‘word’** option you can set the length of the seed to align during multiseed alignment. Smaller values make alignment slower but more sensitive. Default value is 20.

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of seed ‘word’</td>
<td>20</td>
</tr>
<tr>
<td>Interval between seed ‘words’</td>
<td>5,11,0.75</td>
</tr>
<tr>
<td>Disallow gaps within the first/last:</td>
<td>4</td>
</tr>
<tr>
<td>Include &lt;int&gt; extra ref chars:</td>
<td>15</td>
</tr>
<tr>
<td>Max number of ambiguous characters, f(read length):</td>
<td>1,0,0.15</td>
</tr>
<tr>
<td>Allowed mismatches in seed alignment:</td>
<td>0</td>
</tr>
<tr>
<td>Ignore Base Qualities</td>
<td>True</td>
</tr>
<tr>
<td>Map only to reverse strand</td>
<td>False</td>
</tr>
<tr>
<td>Map only to forward strand</td>
<td>False</td>
</tr>
</tbody>
</table>

**Interval between seed ‘words’** sets a function governing the interval between seed substrings to use during multiseed alignment. For instance, if the read has 30 characters, and seed length is 10, and the seed interval is 6, the seeds extracted will be:

```plaintext
Read: TAGCTACGCTCTACGCTATGCATGCTAAAC
Seed 1 fw: TAGCTACGCT
Seed 1 rc: AGCGTAGCTA
Seed 2 fw: CGCTCTACGC
Seed 2 rc: GCCTAGAGCG
Seed 3 fw: ACGCTATCAT
Seed 3 rc: ATGATAGCGT
Seed 4 fw: TCGATCATAA
Seed 4 rc: TTATGCATGA
```

Since it's best to use longer intervals for longer reads, this parameter sets the **interval as a function of the read length**, rather than a single one-size-fits-all number. For instance, specifying values of:

“5,1,2.5” sets the interval function f to

\[ f(x) = 1 + 2.5 \sqrt{x} \]
Where \( x \) is the read length. If the function returns a result less than 1, it is rounded up to 1. The default value is 5.11,0.75.

**Note:** Some options specify a function rather than an individual number or setting. In these cases the user specifies three parameters written this way: a function type, a constant term, and a coefficient. All separated by comma, without whitespaces, decimals separated by dots, negative numbers are allowed. The available function types are: constant (C), linear (L), square-root (S), and natural log (G). For example: G,1,5.4 would look like this \( f(x) = 1 + 5.4 \ln(x) \).

Disallow gaps within the first/last: [4]

With this option you can specify the length of the region, at the beginning and at the end of the read, you want to disallow gaps for. Default value is 4. Disallowing gaps in longer regions makes the process less accurate (leaving more reads unmapped) but faster.

Include <int> extra ref chars: [15]

This option "Pads" dynamic programming problems by the number of specified columns on either side, to allow gaps. Default value is 15.

Max number of ambiguous characters, \( f(\text{read length}) \): [L,0,0.15]

Max number of ambiguous characters sets a function governing the maximum number of ambiguous characters (usually Ns and/or .s) allowed in a read as a function of the read length. For instance, specifying “-L,0,0.15” sets the maximum N allowed to

\[
f(x) = 0 + 0.15x
\]

Where \( x \) is the read length. Reads exceeding this ceiling are filtered out. Default value is L,0,0.15.

Allowed mismatches in seed alignment: [0]

Sets the number of mismatches allowed in a seed alignment during multiseed alignment. It can be set to 0 or 1. Setting this higher makes alignment slower (often much slower) but increases sensitivity. Default value is 0.

Ignore Base Qualities

If you choose to ignore quality, input will be treated as though all read quality values are high. In other words, when a mismatch penalty is calculated, the quality value at the mismatched position will be
considered to be the highest possible, regardless of the actual value. This is also the default behavior when the input doesn't specify quality values (e.g. FASTA files).

In case you are using single-reads (not paired-end), you can choose the strand you want them to align to: If *Only to reverse* is specified, bowtie2 will not attempt to align unpaired reads to the forward (Watson) reference strand. If *Only to forward* is specified, bowtie2 will not attempt to align unpaired reads against the reverse-complement (Crick) reference strand. In paired-end mode, *Only to reverse* and *Only to forward* pertain to the fragments; i.e. specifying *Only to reverse* causes bowtie2 to explore only those paired-end configurations corresponding to fragments from the reverse-complement (Crick) strand. By default both strands are enabled.

**Sorting options**

In this tab are the options related with the unsorted `.sam` file. *Originally created in the mapping process, the software will, by default, sort this file, transform it to its lighter binary version (.bam), and delete it*. If you *Skip sorting* the `.sam` file, it will neither be sorted nor deleted. The option *Keep unsorted sam file* is designed to avoid deleting the `.sam` file after sorting and transforming it.
Finally, the **Map Reads** button will start running the process invoking Bowtie2, after validating the data entered. It won’t ask for any further confirmation. If you have the NGSEPView bar activated, you will be able to follow the progress of this task.

Once this process has finished, you will end up with three new files in your workspace: `.bam`, `.bai`, and `.log`. You will need the `.bam` file for further analysis; this file has all the reads matched against the reference. You can also find useful statistical information in the `.log` file.

⚠️ **Note:** When you execute any process, a progress bar will be displayed on the bottom, it represents the percentage of the task completed. This is important because many times this process can take from several minutes to several hours/days depending on how complex your organism is. If you want to stop the process you are able to do it by pressing the red button in the right side of the progress view, you won’t be able to continue it later. At the end of the process you will see the output files in the directory that you selected.
Multi-Mapping

INPUT FILES

- **Folder with all the read files**: These are the files generated by the sequencing platform of your election with the sequence (and quality) information of your samples. These can be single-read or paired-end reads (in which case both files should have the same prefix but different numbers: 1 and 2).
- **(*)Bowtie Index**: This is the reference sequence to which your reads should be mapped, it must be an indexed reference created by the *Create Index Bowtie* option described above.

OUTPUT FILES (one set of files will be created for each sample)

- **Sorted .bam**: bam file with all the reads organized by chromosome and reference position.
- **.bai**: Binary file needed to visualize the mapped reads in the Integrated Genome Viewer (IGV)
- **.log file**: information about the finished process.

This process is similar to Map Reads but allows performing **multiple mappings** in parallel, saving time since the execution of the process is divided and executed in parallel into the number of processors you define. Likewise, this function has the same requirements as the Map Reads function: to have Bowtie2 installed, and to index your reference genome (See Map Reads).

For this function, you will need to create a new folder or directory in Eclipse containing only your FASTQ files, whether they are paired or simple reads (they can also be mixed in the same folder). **These files need to be uncompressed** (i.e. if you have .gz, .zip, or .rar files, you need to uncompress them before uploading them in Eclipse). Right click on the directory and select one of the *Multi Mapping* functions available in the NGSEP menu, these are: *Multi Mapping Single End*, or *Paired End*. It will depend on the type of sequencing procedure done. If you select the *Single End* option, NGSEP will not attempt to match the files in pairs. If for example, you are dealing with both, single end and paired end reads, the *Paired End* option should be selected, and NGSEP will match the paired files, and the others will be left alone.
The Multi Mapping screen will display your FASTQ files organized on a table, with each pair of files (in case of complementary data) or simple files in one line. It will recognize paired files by the same name and different number (1 and 2). Select and unselect all the files for the mapping process with the option, or individual samples by selecting the check box at the left.

In the Output Directory you can select the path and directory where all your output files are going to be generated; by default is the same directory path where the process was launched.
Exception: To enter the next screen you must select at least one sample from the table.

If you want to change any of the files of your current directory, double-click the cell you want to change, and browse the directory path of your new sample.

After clicking Next, you will find another screen with the Map Reads options. Choose the adequate parameters (explained in Map Reads parameters above) for your analysis and run the process by clicking the Map Reads button.
By writing in this box, the user can define the amount of processors he wants the software to use during the mapping process. By default the software will suggest the user to use all but one of the processors available in the machine. You can still use all the processor available in your computer, but this will make it too slow to work on anything else.
Variants Detector

This is the main functionality of NGSEP, which compares a .bam file against a reference genome to detect both Single Nucleotide Polymorphisms (SNPs) and structural variants (SVs). The variants detector combines different families of algorithms for variants detection including a bayesian approach to detect SNPs and small indels, a read depth (RD) analysis to identify Copy Number Variants (CNVs), and a read pair (RP) analysis to identify large indels and inversions. This function can also identify repetitive regions within the genome based on evidence provided by reads with multiple alignments. Details of the algorithms implemented in the variants detector can be found in the papers referred at the end of this manual. Because variants detection is a very active field of bioinformatics research, it is expected that different algorithms will be added or improved as we release newer versions of NGSEP.

INPUT FILES

- **Sorted .bam file**: This file is created as the result of the alignment of short reads to a reference genome (see Map Reads above). This file needs to be sorted by chromosome and reference position (if you didn’t create this file with the Map Reads function, see Sort Alignment) in order to use the variants detector.
- **Reference genome**: This is the reference sequence to which the .bam file will be compared; it must be in FASTA format.
- **Known SVs file and/or known VCF file (optional)**: In case you want to compare your samples not only with the reference genome, but also with other, previously determined, variants.

OUTPUT FILES

- **.vcf (Variant call format)**: Tab delimited text file containing SNPs and small indels information; this is a flexible and extensible file format for variation data such as (SNPs), small INDELs, CNVs and structural variants. For more information of the file format see: http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41. In eclipse you are able to visualize the first 100 lines of this file by double clicking on it, if you want to see the totality of it you can use a Terminal, trying to open it in a common text editor will consume a huge amount of memory.
- **GFF (General Feature Format)**: Tab delimited text file containing the Structural Variants (SV) information; this format is generally used to annotate biologically relevant characteristics in a sequence. Composed by 9 mandatory fields. For more information see: http://www.sanger.ac.uk/resources/software/gff/spec.html
- **.log**: Information about the finished process.
- **HistoryFileVCF.ini**: this file is generated automatically and is located at your project directory, don’t move or delete it.

Once you have your reference genome indexed, and your .bam file sorted, you can execute the Variants Detector function in NGSEP. To begin, select one .bam file, right click on it and choose the Find Variants option from the NGSEP menu.

⚠️ **Note**: Make sure that the selected file is a Sorted Bam File otherwise the process will not work.
Variants Detector Window

![Variants Detector Window](image)

- Main arguments
- SNV detection
- SV detection
- File path
- Output File Prefix
- Reference File
- Known SVs File
- Known Variants (.vcf) File
- Skip detection of repetitive regions
- Skip read depth analysis
- Skip new CNV detection with RD
- Skip read pair analysis
- Skip detection of SNVs and small indels

Find Variant  |  Cancel
Variant Detection parameters

Main arguments

(*)File: This field shows the path of the sorted .bam file that you selected. The browser on the right lets you change the input file.

(*)Output File Prefix: This field refers to the output files you will generate with this function. The two files that will be generated will have the same output prefix for further identification. (VFC: for SNPs and Small indels, GFF: for repeats, CNVs and large indels) The browser on the right lets you change the prefix and the destination directory of your output file.

⚠️ Notice that the output directory suggested is the same of the input file as well as the name of the tested sample.

(*)Reference File: This field is for the reference genome used to map your reads. The first time that you execute this functionality this text field will be blank, you must browse for a FASTA file with the reference genome. However, for further executions this field will display the last reference used.

Known SVs (.gff) File: If you already have a .gff file with structural variants information, you can enter the path here. This is useful for analysis of WGS data if the user wants to estimate the number of copies of previously identified repetitive regions or known CNVs (CNV genotyping) using the read depth analysis.

Known Variants (.vcf) File: In this field you can enter the path of a vcf file containing the SNPs and small indels that the user needs to genotype. In this mode, NGSEP will produce genotype calls only for the variants included in this file and will produce homozygous reference genotype calls for sites in which this is the most likely genotype. See Merge VCF below for a detailed description on how this field is used during the genotyping step of the pipeline.
This section is composed by five parameters that represent the whole variant detection process. Select the option you want to skip or avoid during the variants detection process.

**Skip detection of repetitive regions:** Select this option to skip the detection of repetitive regions based on reads with multiple alignments.

**Skip read depth analysis:** Select this option to completely skip the analysis of read depth (RD) for detection and genotyping of Copy Number Variants (CNVs).

**Skip new CNV detection with RD:** Select this option to skip the detection of new Copy Number Variants (CNVs). In this mode, the read depth distribution will still be calculated but the estimation of copy number will only be performed for repeats and CNVs provided in the known SVs file or discovered by the algorithm to detect repetitive regions.

**Skip read pair analysis:** Select this option to skip the detection of large indels and inversions using the read pair (RP) information.

**Skip detection of SNVs and small indels:** Select this option to skip the detection of Single Nucleotide Polymorphisms (SNPs) and small indels. In this mode, only structural variants will be identified and then an output vcf file will not be produced.

⚠️ **Note:** If you don’t select any option from the Execution Parameters NGSEP will, by default, execute all the finding processes of the Variants Detector. On the contrary, if you select all of them, NGSEP will run the Variants Detector without producing any results.

**Sample ID:** You can give your samples any specific ID to label the header of the VFC file.

After checking the quality of your reads (see Calculate Quality Statistics below), you can determine if you want to ignore some of the first or some of the last bases from all of your reads. Ignoring low quality bases will make a more accurate search.
SNV detection

This section is to establish parameters that *can* improve the SNVs detection.

⚠️ **Note:** Some fields are filled with default values, however if you are aware about their meaning you can change them on demand according to your sample or research.

**Genomic Location** (optional): In this field, enter a specific location from the reference genome in which you want to locate SNPs. This is an example of the format accepted: ‘chr21:32,000,000-33,000,000’ (chrID:start-stop positions).

⚠️ **Note:** you must be aware of the number of chromosomes/scaffolds and range of detection.

**Heterozygosity Rate:** This field is intended to enter the prior probability of finding in every certain position heterozygous SNPs. By default this probability assumes that the sample will have about one heterozygous SNP every 1000 basepairs. This assumption is reasonable for diploid outbred species such as humans. For inbred species with only residual heterozygosity or polyploids with higher expected heterozygosity, changing this parameter accordingly should increase the accuracy of both detection and genotyping.

**Minimum Genotype Quality Score:** Indicates the minimum posterior probability to discover a novel variant against the reference or to genotype an existing variant. In genotyping mode, variants for which the predicted genotype is lower than this threshold will still be output with an undecided genotype. The
probability is encoded as a Phred Score (0-255). The default value of 40 corresponds to a minimum posterior probability of 0.9999. This parameter allows the user to control the tradeoff between sensitivity and specificity of the variants detection. Larger values reduce the error rate but also reduce the number of variants discovered or genotyped. Smaller values increase the number of variants discovered or genotyped but also increase the error rate.

**Maximum Base Quality Score**: Maximum value allowed for a base quality score. Larger values will be equalized to this value. This parameter reduces the effect of sequencing errors with high base quality scores.

**Alternative Allele Coverage**: Minimum and maximum coverage of the alternative allele to call a SNV. Default values are 0 and 0(no limit).

**Maximum Alignment Per Start Position**: Maximum number of reads allowed to start at the same reference site. This parameter helps to control false positives produced by PCR amplification artifacts. Default value is 2.

- **Print Sample Ploidy**: This generates a new header in the output VCF file with the global ploidy of the analyzed sample (ploidy parameter). This header is used during the VCF filter step to recalculate the number of samples with CNVs for each SNP or small indel.

  - Note: It is possible, that other software reading this VCF file won’t recognize this header, and therefore not accept the file as a VCF.

- **Ploidy**: Depends on the ploidy of your sample. It ranges from 1 (haploid), to n (n-ploid).

- **Ignore Lower Case References**: Ignore sites where the reference allele is lowercase, i.e. repetitive regions.

- **Include Secondary Alignments**: Consider secondary alignments while calling SNVs, and not only the best one.

- **Genotyped All Covered Sites**: Report all covered sites in the genome.
Note: Checking **Genotyped All Covered Sites** for WGS samples and even for RAD/GBS samples in organisms with medium to large genomes produces a huge file.

**Ignore XS field:** Ignores the optional field XS to decide if an alignment is unique. While bowtie2 only outputs the XS field for reads with multiple alignments, BWA-MEM and BWA-SW outputs this field for every alignment in the BAM file. Hence, this flag should always be used to process BAM files made with BWA-MEM and BWA-SW.

**SV detection**

![Variants Detector](image)

- **Genome size:** Total size of the genome to use during detection of CNVs. This should be used when the reference file only includes a fraction of the genome (e.g. a chromosome or a partial assembly).

- **Bin size:** Size of the bins to analyze read depth. Default value is 100.

- **Ignore proper pair flag:** Check this to avoid taking into account if the distance between both reads of the pair matches the expected.

- **Algorithm selection for CNV detection:** It is possible to select which algorithm will be used by NGSEP to detect the CNVs present in each sample. The list contains: CNVnator (Abyzov, et al. 2011) and EWT (Yoon, et al. 2009)

- **Min quality:** (for Structural Variants) only the variants with a higher quality than the one specified here will be reported in the final GFF file. Values are in phred score. Default value is 20.
**Max deletion length:** The maximum size of the deletions to be used for detecting breakpoints using the split-read algorithm. Default value is 1’000000.

**Seed size (split-read):** The size of the seed to be used during the search of breakpoints using the split-read algorithm. Default value is 8.

**Max % overlap repeats-CNVs:** When a new CNV is called it will be checked if it overlaps with regions previously identified as repetitive. If so, regions exceeding the given overlapping percentage will be filtered out. Default value is 100%, which means that CNVs can overlap completely with repetitive regions and still be reported.

Use the **Find Variants** button to execute

---

**Multi Variants Detector**

**INPUT FILES**

- **Folder with all the sorted .bam files:** This files are created as the result of the alignment of short reads to a reference genome (see **Map Reads** above). This file needs to be sorted by chromosome and reference position (if you didn’t create this file with the **Map Reads** function, see **Sort Alignment**) in order to use the variants detector.

- **Reference Genome:** This is the reference sequence to which the .bam files will be compared, it must be in FASTA format.

- **Known SVs file and/or known VCF file (optional):** In case you want to compare your samples not only with the reference genome, but also with other, previously determined, variants.

**OUTPUT FILES** (one set of files will be created for each sample)

- **.vcf (Variant call format):** Tab delimited text file containing SNPs and small indels information; this is a flexible and extensible file format for variation data such as (SNPs), small INDELS, CNVs and structural variants. For more information of the file format see: [http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41](http://www.1000genomes.org/wiki/Analysis/Variant%20Call%20Format/vcf-variant-call-format-version-41). In eclipse you are able to visualize the first 100 lines of this file by double clicking on it, if you want to see the totality of it you can use a Terminal, trying to open it in a common text editor will consume a huge amount of memory.

- **GFF (General Feature Format):** Tab delimited text file containing the Structural Variants (SV) information; this format is generally used to annotate biologically relevant characteristics in a sequence. Composed by 9 mandatory fields. For more information see: [http://www.sanger.ac.uk/resources/software/gff/spec.html](http://www.sanger.ac.uk/resources/software/gff/spec.html)

- **.log:** Information about the finished process.

**HistoryFileVCF.ini:** this file is generated automatically and is located at your project directory.

This process is similar to Variants Detector (See **Variants Detector**) but allows performing **Variants Detector** to multiple samples in parallel, saving time since the execution of the process is divided and executed in parallel into the number of processors you choose.
For this function, you will need to create a new folder or directory in Eclipse only with your sorted .bam files. Right click on the directory and select the *Multi Variants Detector* function in the NGSEP menu.
The Multi Variants Detector window will display your sorted bam files, each file on a line, organized on a table with the following columns: Input BAM File, Sample Id and Output Prefix. Select and unselect all the files for the finding variants process with the option, or individual samples by selecting the check box at the left.

In the Output Directory you can select the path and directory where all your output files are going to be generated; by default is the same directory path where the process was launched.

⚠️ Exception: To enter the next screen you must select at least one sample from the table.

If you want to change any of the files of your current directory, double-click the cell you want to change, and browse the directory path of your new sample.
Likewise, you can change the Sample Id for your output files and the Output Prefix by double clicking the cell you want to change, and entering a new name for these cells in a text box.
After clicking **Next**, you will find another screen with the Variants Detection options. Choose the adequate parameters (explained in **Variants Detection parameters**) for your analysis.

By writing in this box, the user can define the amount of processors he wants the software to use during the Variants detection process. By default the software will suggest the user to use all but one of the processors available in the machine. You can still use all the processor available in your computer, but this will make it too slow to work on anything else.
Merge VCF

This process is intended to merge variants from different samples into an integrated VCF file. The **PROCESS IS DIVIDED IN THREE STEPS**; the first one is intended to determine a list of variants found in at least one of the VCF files that were generated in the variants detection process, this step will generate one common VCF file with the union of variants reported by all the input files but without information about the genotypes present, just the sites where the variants were found (this happens because if a sample matches the reference genome it won’t appear).

The second step requires running again Variants Detector for all your samples (can use **Multi Variants Detector**) this time using the mentioned common file (VCF) in the *known variants file* field. This will produce a second set of VCF files which will differ slightly from the first set in the sense that it will include the sites that were indicated in the known variants file, even if they are the same as the reference genome.

Finally, join or merge those new created VCF files in one VCF for all the samples, this new VCF will contain **information about every sample in every locus**, even if they are the same as the reference genome, this will be useful to analyze the variation at each site.

**INPUT FILES**

- **VCF files**: These files are the result of the *Find Variants* process; each one corresponds to a sample of your analysis, and contains information about their differences with the reference genome.
- **HistoryFileVCF.ini**: This file was generated automatically during the variants detection process and is located at your project directory. The variants detection history file **must have at least two samples**.
- **Common variants file (for the second step)**: This file is the result of the first step and is going to be required during the second step to compile the information about all variations in your samples.

**OUTPUT FILES**

- **Variantsfile.vcf**: With information about each sample and its differences with the reference, it will be needed.
- **Variantsfile_DV.log**: Information about the first step process.
- **New VCF files for all your samples**: These files are almost like the first VCF files that were created with the *Find Variants* option, except that they have information about sites where they don’t differ from the reference sequence.
- **variantsfile2_MVCF.log**: Information about the second step process.
Once you have run the **Variants Detector** and you have all your .vcf files with the variation information from each of your samples, click on the file named *HistoryFileVCF.ini*, and choose the **Merge VCF** option from the NGSEP menu.
Select all files: Click to check all the samples.
Deselect all files: Click to uncheck all the samples.
(*)Output File: This field means the Merge VCF output file and is mandatory.

1. FIRST STEP - Determine list of variants: This option is used to join in one VCF file, the whole list of places where the samples are diverging from the reference. This doesn’t have the whole information about all samples genotypes, but all variant alleles found in at least one of the samples.

2. SECOND STEP - Merge VCF files: This option is used to join all genomic variants found in the individual VCF files (after the second variant detection) from each sample, into a single VCF file for all samples, matching each corresponding variation with their corresponding genotype in each sample.

⚠️ Note: Don’t select this option unless you already went through the whole first step, otherwise it won’t run.

FIRST STEP:

Select all files and click on the Determine list of variants button.

An example of the first output VCF file using the Determine list of variants option should look like this:
This output file is vital for the execution of the second process (Merge VCF), because this file is required to execute again the Variants Detector for all samples in order to associate the variant allele with their corresponding sample genotype (can also be Multi Variants Detector).

SECOND STEP:

After finishing the variants list determination, proceed to right click on each BAM file using the known variants field to run variants detector again as follows:
In the **Known Variants (.vcf) File** field, you should provide the VCF file you just created above with the **first step:** **Determine list of variants.**

After running the **Variant Detector** again for each sample, using the **Known Variants (.vcf) File**, you should have one output VCF file for each individual sample, each of which will look something like this:

The sample corresponding to this vcf
After performing this step with all the alignment files and having your new VCF files for each sample, proceed to right click again on the HistoryFileVCF.ini file, and select the **Merge VCF** option on the **NGSEP menu**. In the Merge VCF window select all the new VCF files and click on the **Merge VCF files** button.

**Final Result from Merge VCF:**

Finally, the output file after using **Merge VCF** should be a **VCF file** that includes all the samples, with their corresponding genotypes, and should look like this:
Variants Functional Annotator

INPUT FILES

- **VCF file**: This file contains the information about the variation detected among the samples, it can be a single sample, or the compilation of all the samples.
- **Reference genome**: This is the reference sequence; it must be in FASTA format.
- **GFF file**: Tab delimited text file, generally used to annotate biologically relevant characteristics in a sequence, composed by 9 mandatory fields containing gene annotations related with the given genome. There are several versions (.gff-.gff3) and all of them are accepted. However, NGSEP expects a description of gene models following the GFF3 format, which is more specific than GFF in the sense that it specifies how a gene and its corresponding transcripts, exons and CDS should be reported (see [http://www.sequenceontology.org/gff3.shtml](http://www.sequenceontology.org/gff3.shtml) for details).

OUTPUT FILES

- **_Annotated.vcf**: This file contains the information about the variation found in the samples analyzed, with the functional annotation corresponding to the genome region.
- **.log**: Information about the finished process.

Once you have obtained a .vcf file, through the **Variant Detector** or through the **Merge VCF** function, right click on the file, and choose the **Variants Functional Annotation** option from the NGSEP Menu.
Window of the Variants Functional Annotator

(*VCF) Variants File: This field is for the path of the VCF file that was selected. The browser on the right can also be used to change the input file.
(*) Gene Annotation File: This field is for the basic input of annotations. At the beginning it will be blank; you have to browse a GFF file with the genome annotations. For further executions the field will display the last file used.

(*)FASTA) Genome Reference: This field is for the reference genome that will be used to compare your data. The first time that you execute this functionality this text field will be blank, you must browse for a FASTA file with the reference genome. For further executions the field will display the last reference used.

(*VCF) Output File: This field is for the path and name where the output file will be located; we recommend using the same project directory.

The Upstream and Downstream options allow you to define the distance from the annotated gene, in which a variation should be marked as upstream or downstream of the gene.

Use the Functional Annotation button to execute. To close the window click Cancel.

Final Result for Variants Functional Annotator:

At the end of this process the output file will be a VCF file holding the information about the samples variations and the genes being affected.
<table>
<thead>
<tr>
<th>Gene</th>
<th>Chromosome</th>
<th>Start</th>
<th>End</th>
<th>Strand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gene1</td>
<td>1</td>
<td>1000</td>
<td>5000</td>
<td>Forward</td>
<td>Sample data</td>
</tr>
<tr>
<td>Gene2</td>
<td>2</td>
<td>2000</td>
<td>7000</td>
<td>Reverse</td>
<td>Another sample</td>
</tr>
<tr>
<td>Gene3</td>
<td>3</td>
<td>3000</td>
<td>8000</td>
<td>Forward</td>
<td>Yet another sample</td>
</tr>
</tbody>
</table>
Filter VCF Files

This function implements different filters on annotated VCF files with genotype information. It creates a new VCF file with variants passing the filtering criteria.

INPUT FILE

- **VCF file**: This file contains the information about the variation detected among the samples, normally, it is the compilation of all the samples analyzed.

OUTPUT FILES

- **Filtered VCF file**: A new file containing information about all the samples present in the original VCF, but only with a subgroup of the variations detected, depending on your filtering criteria. It can be much smaller than the original VCF.
- **.log**: Information about the finished process.

Once you merged your VCF files into one single file with all the genotypes information (from **Merge VCF**, see above), right click on the VCF file, and choose the **VCF Filter** option from the NGSEP menu.
(*)File: This is the path of the input file that you selected. You can also use the browser on the right if you want to change the input file. We recommend having all the input files in the project directory.

(*)Output File: Name and path where you want your output file; we also recommend using the same project directory. Default option for your filtered output is the same directory with the name of your VCF file plus “_filter”.

Main Arguments:

Filter Regions From File: For this option you need to upload a file specifying the genomic regions in which variants should be excluded or filtered out from the VCF. The format of this file should be tab delimited and it should contain three columns:

- First column: Sequence name (chromosome or scaffold)
- Second column: First position in the sequence.
- Third column: Last position in the sequence.

Both positions are assumed to be 1-based. The file should look like this:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ChrI</td>
<td>1259000</td>
<td>1326000</td>
</tr>
<tr>
<td>ChrI</td>
<td>2571000</td>
<td>3225000</td>
</tr>
<tr>
<td>ChrI</td>
<td>1</td>
<td>4730</td>
</tr>
<tr>
<td>ChrII</td>
<td>43500</td>
<td>44000</td>
</tr>
<tr>
<td>ChrII</td>
<td>963800</td>
<td>1358300</td>
</tr>
<tr>
<td>ChrV</td>
<td>496493</td>
<td>496512</td>
</tr>
<tr>
<td>ChrVI</td>
<td>2385</td>
<td>5042</td>
</tr>
<tr>
<td>scaffold0341</td>
<td>1717</td>
<td>1725</td>
</tr>
<tr>
<td>scaffold3237</td>
<td>108433</td>
<td>108435</td>
</tr>
<tr>
<td>scaffold4692</td>
<td>1</td>
<td>13000</td>
</tr>
</tbody>
</table>

Select Regions From File: For this option you need to upload a file specifying the genomic regions from which the variants should be included in the VCF file. The format of this file should be the same as the one described in Filter Regions From File in the above section.

Minimum distance between variants: With this option, you can specify a value for the minimum physical distance between the variants displayed on your VCF file. You can choose any integer value. Default value is 5.
Minimum and Maximum minor allele frequency (MAF) over the samples in the VCF file: You can use this option to filter variants by the minimum frequency at which the least common allele occurs in the population, commonly known as MAF. This value ranges from 0 to 0.5, being values near 0 very rare alleles, while 0.5 very common variations.

Minimum number of samples genotyped to retain variant: This option is useful to filter out variants with a lot of missing data. With this option, you keep only variants that have genotypic information in at least the number of samples you specify. Default value is 1, meaning no filter at all.

Maximum number of samples with copy number variation: (CNV) in the region where the variant is located

Minimum genotyping quality score: You can exclude all genotypes with a quality score below the threshold specified by this option. The GQ score is a Phred-scaled confidence that the true genotype is the one provided in the called genotypes, it can be found in the GQ field for each genotype call in the VCF. Values range from 0 to 255. Default value is 40

Minimum coverage (to keep a genotype call): This option uses the coverage information of each variant in the VCF file, to keep or filter the genotype call of the variant. Default value is 1, where no filter is applied.

Keep only biallelic SNVs: Select this option if you want to keep only biallelic SNPs, filtering out any other kind of variants (INDELs or SNPs with more than one alternative alleles in the population).
**Filter Invariant Sites:** Select this option if you want to keep only variants in which at least two alleles are identified in the population (polymorphic variants).

**Filter Invariant Alternative:** Select this option if you want to filter variants in which only one alternative allele is observed in the population, keeping variants in which at least two alleles are identified in the population (polymorphic variants) or variants fixed for the reference allele.

**Filter Invariant Reference:** Select this option if you want to filter variants in which only the reference allele is observed in the population, keeping variants in which at least two alleles are identified in the population (polymorphic variants) or variants fixed for an alternative allele.

**GC Content Filter:**

![Image of GC Content Filter](image)

**Filter for GC-content of the region surrounding the variant:** You can filter by the minimum and maximum percentage of GC of the 100bp region surrounding the variants. This requires uploading the reference file in FASTA format in order to gain information of the flanking area. Upload the reference file in the field “Reference” and change the minimum and maximum value of GC Content according to your needs. Default values are 40-65.

**Functional Filter:**
**Gene name:** Filter annotated VCF files by a specific gene or transcript ID. Type the gene or transcript ID that you want to include in your VCF file, exactly as it appears annotated in your VCF file.

**Functional Role:** Filter an annotated VCF file according to the type of functional annotation of the variants. Choose one or more of the following options to be included in the resulting VCF: Intron, Intergenic, FivePrimeUTR, ThreePrimeUTR, Upstream, Downstream, NCRNA, Synonymous, Missense, Nonsense, Frameshift or ExonJunction.
Sample Selection:

Filter by including or excluding selected samples from your VCF: For this option you need to upload a file specifying the samples ID's you want to Include (select) or Exclude (filter) from the VCF. This file should be a one column, one sample ID per line, text file, and should look like this:

![Notepad image showing sample IDs]

Final Result for Filter VCF:

At the end of the process you will see a file similar to the input, but smaller, only with the information that passed your filtering criteria.
Export VCF files to other formats

This function is designed to convert the genotype calls from VCF format to other formats for commonly used genetic software packages, to perform different kinds of analysis. Available options to date are: Structure, Hapmap, Haploview, EigenSoft, FASTA, Spagedi, Emma, Flapjack, Matrix, Plink (.map and .ped), PowerMarker, Darwin, Treemix and JoinMap.

**INPUT FILES**

- **VCF file**: This file contains the information about the variation detected among the samples, it could be a single sample, or the compilation of all the samples.
- **Populations File (only for Treemix)**: text file with population assignments for each sample included in the VCF (See format below).

**OUTPUT FILES**

- **Formatted file for exporting data to other software**: Depending on your choice, the files with your samples information to be used in other analysis.
- **VC.log**: Information about the finished process.

Once you have the VCF file with the variation information. Right click on it and choose the **VCF converter** option from the NGSEP menu. Choose the file format you want to export your VCF into; you can choose as many file formats as you want simultaneously.
In addition to known formats for commonly used software, we provide an additional format option Print Matrix, which prints a matrix of genotypes in a simple ACGT format, which can be imported and visualized in Excel.

When you select the Print Treemix option, a text box appears for you to introduce another file.

**Populations File:** Enter the name and path of the file with the Population information of each sample in the field Populations File. The populations file is a tab-delimited text file with two columns: sample ID and population ID. Sample ID must be the same as in the VCF file and population ID can be a numeric or alphanumeric name without whitespaces. The file should look like this, with no header:

```
10 ARG66  LRLAC
11 ARG64  LRLAC
12 ARG65  LRLAC
13 ARG67  LRLAC
14 B92030  AFR
15 BOL1   LRLAC
16 BOL2   LRLAC
17 BOL3   LRLAC
18 BRA1003  LRLAC
19 BRA1010  LRLAC
20 BRA1016  LRLAC
21 BRA1031  LRLAC
```

This file is required only to export the VCF to the Treemix format.

When you select the Print JoinMap option, two text boxes appear for you to type the ID of both samples.

⚠️ **Note:** FASTA conversion does not use IUPAC codes, heterozygous SNPs are changed to N

⚠️ **Note:** Plink is only designed for humans, therefore it will only work for 22 sequences (chromosomes). If a sample exceeds that number of sequences, we strongly recommend you to reduce the number of sequences to 22 and remove all remaining scaffolds.

⚠️ **Note:** To generate dendograms in Tassel, it is better to use the HapMap format.

⚠️ **Note:** The JoinMap format is currently designed to convert only to outbreeding full sib families or CP population type.
One-Step Wizard, Automatic Pipeline

This feature is designed to perform an automated analysis of a population, from the raw reads to the final population variants file. This enables an unsupervised and automatic completion of the whole pipeline.

INPUT FILES

- **Folder with all the read files**: These are the files generated by the sequencing platform of your election with the sequence (and quality) information of your samples. These can be single-read or paired-end reads (in which case both files should have the same prefix but different suffix numbers: 1 and 2).
- **(*)Bowtie Index**: This is the reference sequence to which your reads should be mapped, it must be an indexed reference created by the *Create Index Bowtie option* described above.
- **Reference genome**: This is the reference sequence to which the .bam file will be compared; it must be in FASTA format.

OUTPUT FILES

- **Sorted BAM**: BAM file with all the reads of a sample organized by chromosome and reference position.
- **BAI file**: Binary index of each BAM file.
- **VCF files for initial variant discovery**: Tab delimited text file containing SNPs and small indels information in a sample.
- **GFF (General Feature Format)**: Tab delimited text file containing the Structural Variants (SV) information for each sample.
- **Variantsfile.vcf**: List of all variant sites present in the population, in VCF format.
- **VCF file for sample genotyping (**gt.vcf)**: These files are almost like the first VCF files except that they contain information about sites where they don’t differ from the reference genome.
- **Population VCF file**: Final file, containing all the samples genotyped for each variant site in the population. This is the file to be used in downstream analyses.
- **log files**: information about each finished process.

*One file will be created for each sample*

* In eclipse you are able to visualize the first 100 lines of this file by double clicking on it, if you want to see the totality of it you can use a Terminal, trying to open it in a common text editor will consume a huge amount of memory.

To start, right click on the folder containing all the reads and choose the **Wizard** option from the NGSEP menu, either for **Single End** or **Paired End**, depending on your experiment.
This will open a new window to select the samples you want to include in the analysis, and the output directories for the final files.

The number of processors will determine how fast the process runs. But selecting all the processors may leave your computer unable to run any other application. The Final vcf Prefix will be the name for the last file containing the genotype information from the whole population. You have also the option to stop the whole pipeline if an error arises in the processing of any sample.

⚠️ Note: Running many processes at a time can make things faster, it will analyze as many samples at a time, as processors you assigned. But beware, analyzing large samples, or with a big amount of variants, will probably cause your process to exceed the memory overhead allocated to Eclipse. Look the steps above for allowing Eclipse to take more memory. A reasonable estimation of the
memory can be done, by assigning 4Gb per sample, if you are expecting less than 10’000.000 SNPs, if you are expecting a larger number of SNPs, then 7Gb per sample would be fine. In other words, for running this pipeline in two cores, you will need something around 8Gb of memory.

Click on **Next** to continue with the parameters setting. A new window will appear with the following layout:

![NGSEP Wizard](image)

Here, parameters must be set for the two steps of the process: mapping (**Map**) and variants detection (**VD**). First, the Mapping parameters must be set as explained above in the **Map Reads** section.

![NGSEP Wizard](image)

**Alignment** and **Sorting** options are the same as explained before.
The main arguments for the Variants Detection are also the same as when doing it manually step by step.

From the SNV options, the new extra field allows the user to obtain a more sensitive genotyping of the samples, if it allows the use of the minimum Genotype Quality Score only in the first step of variants
discovery. In this case, the first run on NGSEP (discovery), analyzing sample per sample, will have a GQ threshold to declare a position as variant. Then in the second run (genotyping), the information for each sample won’t be filtered by this criterion, obtaining a more complete population VCF file. Filters on GQ can be applied later with the VCF Filter option.

The final tab shows the options previously explained in the SV parameters of the Variants Detector.

To get the fully automatized pipeline going, simply click on the Go Wizard button, and keep track of the progress bar.
Optional Functions

Calculate Quality Statistics

This process compares the reads held in the .Bam file according to the reference genome, and indicates the number of sequencing errors for each position of the reads as one set. It should have a homogenous distribution around one.

INPUT FILES

- **Sorted .bam file**: This file is created as the result of the alignment of short reads to a reference genome (see Map Reads above). This file needs to be sorted by chromosome and reference position (if you didn’t create this file with the Map Reads function, see Sort Alignment) in order to calculate quality statistics.
- **Reference genome**: This is the reference sequence; it must be in FASTA format.

OUTPUT FILES

- **ReadPos.stats**: This is a tab delimited file, with five columns; the first one has the nucleotide position inside the reads, the second the amount of mismatches in multiple alignments, the third the amount of mismatches in unique alignments, the fourth the number of total multiple alignments and the fifth the number of total unique alignments. At the end there is a summary.
- **.png file**: This is an image file that contains the plot for the selected column. 300dpi resolution.
- **PS.log**: Information about the finished process.

Once you have your sorted .bam file, right click on it and choose the Calculate Quality Statistics option from the NGSEP menu.
Calculate Quality Statistics window

(*)File: this field contains the path of the selected input file (this sorted.bam file can be the output file of the “Sort Alignment” function of NGSEP). The browser on the right can also be used to change the input file.

(*)Reference File: This field is for the reference genome that is going to be used to compare the reads from the .bam file. The first time that you execute this functionality this text field will be blank, you must browse for a FASTA file with the reference genome. For further executions the field will display the last reference used.

(*)Output File Prefix: This field is to enter the name and path where you want your output file; we recommend using the same project directory. By default the system will always suggest the project location; however you can select another one if you want.

Graphical Output: A histogram will be generated from the .stats file, in the x-axis will be the position inside the read (first column), and in the y-axis can be either the percentage of mismatches in multiple alignments or in unique alignments.
Use the button with the label **Statistics** to execute.

When you execute the Calculate Quality Statistics, a progress bar will be displayed on the bottom representing the percentage of completed process. This process can take several minutes depending on how many reads you have. To stop the process, press the red button at the right. In the end of the process you will see the output files in the directory in the folder that you selected.

**Final Result from Calculate Quality Statistics:**

At the end of this process you will have generated two files with the same prefix but with different extension. The first file (.stats) holds the statistics from unique and multiple alignments with a summary at the end. And the second one (.png) is the plot. To open the statistics you can use any text editor and for opening the plot you can use any visual program, the resolution of this image is 300dpi.
Plot Quality Statistics

This function generates a plot based on the quality statistics file previously generated with the Calculate Quality Statistics function (see above).

**INPUT FILE**

- **ReadPos.stats**: This is a tab delimited file, with five columns; the first one has the nucleotide position inside the reads, the second the amount of mismatches in multiple alignments, the third the amount of mismatches in unique alignments, the fourth the number of total multiple alignments and the fifth the number of total unique alignments. At the end there is a summary.

**OUTPUT FILE**

- **.png file**: This is an image file that contains the plot for the selected column. 300dpi resolution.

Once you have generated the ReadPos.stats file, right click on it and choose the Plot Quality Statistics option from the NGSEP menu.

**Screen Plot Quality Statistics**

<table>
<thead>
<tr>
<th>(*)File:</th>
<th>/.../EclipseWorkspace/NGSEPtests/Mapping/CBS6412_read_bowtie2_sortedReadPos.stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>(*)Output File:</td>
<td>/.../EclipseWorkspace/NGSEPtests/Mapping/CBS6412_read_bowtie2_sortedReadPos</td>
</tr>
</tbody>
</table>

- **Multiple alignments**
- **Unique alignments**

*File*: this field contains the path of the selected input file (The output file of the “Calculate Quality Statistics” function). The browser on the right can also be used to change the input file.
(*)Output File: This field is to enter the name and path where the output file will be; we recommend using the same project directory. By default the system will always suggest the project location; however you can select another one if you want.

Graphical Output: A histogram will be generated from the .stats file, in the x-axis will be the position inside the read (first column), and in the y-axis can be either the percentage of mismatches in multiple alignments or in unique alignments.

Use the Plot Quality Statistics button to execute

Final Result for Plot Quality Statistics:

At the end of this process you will have generated a png file, the resolution of this image is 300dpi. To open it you can use any visual program. The x axis represents the Read Position (From 5’to 3’), and the Y axis the Percentage of non-reference calls; the lower, the better. If this percentage gets very high at the beginning or at the end of the reads, consider trimming your reads for other analyzes.
Calculate Coverage Statistics

This process compares a reference genome with a sample, looking the number of reads in the sample covering each position in the reference genome. It should have a normal distribution.

**INPUT FILE**

- **Sorted .bam file**: This file is created as the result of the alignment of short reads to a reference genome (see Map Reads above). This file needs to be sorted by chromosome and reference position (if you didn’t create this file with the Map Reads function, see Sort Alignment) in order to calculate coverage statistics.

**OUTPUT FILES**

- **Coverage.stats**: This is a tab delimited file, with three columns; the first one has the coverage of each base, the second one the amount of positions in the reference genome with a given coverage (just for multiple alignments), and the third one is the same, but for unique alignments. At the end there are the remaining bases.
- **.png file**: This is an image file that contains the plot for the selected column. 300dpi resolution.
- **CS.log**: Information about the finished process.

Once you have your sorted .bam file, right click on it and choose the Calculate Coverage Statistics option from the NGSEP menu.
Calculate Coverage Statistics Window

(*)File: This field contains the path of the selected input file (this sorted.bam file can be the output file of the “Sort Alignment” function of NGSEP). The browser on the right can also be used to change the input file.

(*)Output File: This field is to enter the name and path where you want your output file; we recommend using the same project directory. By default the system will always suggest the project location; however you can select another one if you want.

Graphical Output: A histogram will be generated from the .stats file, in the x-axis will be the coverage for each reference position (first column), and in the y-axis can be either the amount of reference positions with that coverage in multiple alignments (second column) or in unique alignments (third column).

Use the button with the label Statistics to execute.

When you execute the Calculate Coverage Statistics, a progress bar will be displayed on the bottom representing the percentage of completed process. This process can take several minutes depending on how many reads you have. To stop the process, press the red button at the right. In the end of the process you will see the output files in the folder that you selected.
Final Result from Calculate Coverage Statistics:

At the end of this process you will have generated two files with the same prefix but with different extension. The first file (.stats) holds the coverage statistics from unique and multiple alignments. And the second one (.png) is the plot. To open the statistics you can use any text editor and for opening the plot you can use any visual program. The plot will show a normal distribution around the mean coverage value, the resolution of this image is 300dpi.
Plot Coverage Statistics

This function generates a plot based on the coverage statistics file previously generated with the **Calculate Coverage Statistics** function (see above). It considers unique and multiple alignments. It should have a normal distribution centered on the expected coverage value.

**INPUT FILE**

- **Coverage.stats**: This is a tab delimited file, with three columns; the first one has the coverage of each base, the second one the amount of positions in the reference genome with a given coverage (just for multiple alignments), and the third one is the same, but for unique alignments. At the end there are the remaining bases.

**OUTPUT FILE**

- **.png file**: This is an image file that contains the plot for the selected column. 300dpi resolution.

Once you have generated the **Coverage.stats** file, right click on it and choose the **Plot Coverage Statistics** option from the NGSEP menu.

**Screen Plot Coverage Statistics**
(**)File: this field contains the path of the selected input file (The output file of the “Calculate Coverage Statistics” function). The browser on the right can also be used to change the input file.

(**)Output File: This field is to enter the name and path where the output file will be; we recommend using the same project directory. By default the system will always suggest the project location; however you can select another one if you want.

**Graphical Output:** A histogram will be generated from the `.stats` file; in the x-axis will be the coverage for each reference position (first column), and in the y-axis can be either the amount of reference positions with that coverage in multiple alignments (second column) or in unique alignments (third column).

Use the Plot Coverage Statistics button to execute

**Final Result for Plot Coverage Statistics:**

At the end of this process you will have generated a png file, the resolution of this image is 300dpi. To open it you can use any visual program. The X axis represents the coverage and the Y axis the amount of reference positions.
VCF Summary Statistics Calculator

This functionality will produce a text file compiling useful information about the variation and genotypes present in the samples analyzed. It also includes alignment and annotation summaries.

INPUT FILE

- **VCF file**: This file contains the information about the variation detected among the samples, it can be a single sample, or the compilation of all the samples.

OUTPUT FILES

- **_SummaryStats.txt**: A text file with basic statistical information divided in 6 sections: Summary of the variants detected, the MAF Distributions, Samples Genotyped Distributions, SNP, INDEL and Other variants counts per sample.
- **_SummaryStats_ST.log**: Information about the finished process.

Once you have obtained a .vcf file through the **Merge VCF** function, right click on the file, and choose the **VCF Summary Statistics** option from the NGSEP Menu.
(*) Input VCF File: This field is for the path of the VCF file that was selected. The browser on the right can also be used to change the input file.

(*) Output File: This field is for the path and name where the output file will be located; we recommend using the same project directory.

The **Minimum number of samples genotyped** defines the minimum number of samples that are required to be genotyped to trust the calculations of population parameters included in the VCF file. A variant having less than this minimum number of individuals genotyped will not be considered for the calculation of the MAF distribution and the counts of number of variants with the minor allele. However, these variants will still be counted for general statistics such as number of variants genotyped or number of variants in coding (or missense, not sense, etc.) regions. This value ranges from 0, where all the variations are reported, to the same number of samples analyzed, in which case, with a low coverage, many variations will be lost.

Use the **Summary Statistics** button to execute. Click Cancel to close the window.

**Final Result for VCF Summary Statistics Calculator:**
This process will generate a text file holding the information about the samples variations and the genotyping statistics in the previously named sections.
Plot VCF Summary Statistics

This function generates a series of plots based on the VCF Summary Statistics file previously generated with the VCF Summary Statistics Calculator function (see above).

INPUT FILE

- **SummaryStats.txt**: This is a tab delimited file containing several tables and statistical information that can be plotted; It is divided in 6 sections: Summary of the variants detected, the MAF Distributions, Samples Genotyped Distributions, SNP, INDEL and Other variants counts per sample.

OUTPUT FILE

- **.png files**: depending on the selections made on the window, one or more image files will be generated. Each file contains one plot for one of the selected tables. All plots have a 300dpi resolution.

Once the _SummaryStats.txt file has been generated, right click on it and choose the Plot VCF Summary Statistics option from the NGSEP menu.
(*) VCF Summary Statistics File: This field is for the path of the VCF Summary Statistics file that was selected. The browser on the right can also be used to change the input file.

(*) Plot Name Prefix: This field is for the path and name prefix that the output files will have; all files will have a different suffix, assigned by the program.

Variants Accumulation: Checking this option will open three further possibilities, at least one of them must be checked in order to plot a graph. Each selection will produce two plots, one containing the cumulative amount of fully genotyped variations (SNPs, indels, other variants) for each sample included, and the second representing the percentage of missing data from a [samples X variations] matrix with each new sample included, in this plot, having all the samples genotyped will necessarily represent a full matrix (0% missing data).

SNP Calls per Sample: This option will generate a plot containing the total number of coding SNPs for each sample, showing the proportion of synonymous, missense and nonsense substitutions. X-axis will contains the samples numerically ordered as in the original VCF. Only works for annotated VCFs.
Allele Frequency Distribution: This option will produce a plot showing the Minor Allele Frequency distribution of each kind of genotyped SNPs in the VCF. Total SNPs include coding and not coding. Only works for annotated VCFs.

Select as many options as you want, and use the Plot button to execute. Click Cancel to close the window.

Final result from Plot VCF Summary Statistics:

This process will generate several images plotting the information contained in the VCF summary statistics file, which describe the variation and genotyping statistics for each sample and each type of variation present in the vcf.
**VCF Diversity Calculator**

This module produces basic diversity statistics for each variant in a VCF file.

**INPUT FILES**

- **VCF file:** This file contains the information about the variation detected among the samples, it can be a single sample, or the compilation of many samples.
- **(Optional) Population Information File:** text file with population assignments for each sample included in the VCF (See format below).

**OUTPUT FILES**

- **_diversity.txt:** A text file with the genomic coordinates of each variant plus the following statistics separated by semicolon:
  1. Number of samples genotyped
  2. Expected heterozygosity
  3. Observed heterozygosity
  4. F-statistic (Weir and Cockerham’s 1984)
  5. Minor allele frequency (MAF)
  6. Chi-square value of departure from HWE
  7. Uncorrected p-value of the Chi-square test for departure from HWE

  If the file with population assignments is provided, this module will output one column of statistics for the whole group and one column for each population.

- **_DC.log:** Information about the finished process.

Once you have a .vcf file obtained through the **Merge VCF** function or containing a single sample, right click on it, and choose the **VCF Diversity Calculator** option from the NGSEP menu.
In the field (*VCF File* you can see the path of the input file that you selected. You can also use the browser on the right in case you want to change the input file.

Enter the *Output File* name and path where you want your output file. We recommend using the same project directory. Default option for your output is the same directory with the name of your VCF file plus "\_diversity".

**Optional:** Enter the name and path of the file with the Population information of each sample in the field *Population Information File*. The populations file is a tab-delimited text file with two columns: sample ID and population ID. Sample ID must be the same as in the VCF file and population ID can be a numeric or alphanumeric name without whitespaces. The file should look like this, with no header:
This file is optional. If you don’t upload any file in the field, you still will have the diversity statistics of the overall population from your VCF file.
Genotype Imputation (Beta version)

This Function allows imputation of “missing genotypes from unphased multilocus SNP genotype data” in a VCF. It uses a Hidden Markov Model, which takes into account the parents of the population and the Linkage Disequilibrium between several SNVs in a haplotype. This version of this module is only able to impute genotypes in biparental or multiparental inbred families in which the heterozygosity is low and all parents are included in the population. Only biallelic SNPs are imputed and included in the output vcf file, the current implementation does not produce heterozygous imputed genotypes. We expect to include more general cases and improve the imputation accuracy in future versions of NGSEP.

INPUT FILE

- **VCF file**: This file contains the information about the variation detected among the samples, including the parentals of the population.

OUTPUT FILES

- **_assignments.txt**: A text file showing a table, where each line is a SNP and each column is a sample. Each cell tells where did the genotype of that sample in that SNP came from (parentals).
- **_impute.vcf**: The new VCF file with the previously missing genotypes filled with the inferred ones.
- **_IG.log**: Information about the finished process.

![Image of Genotype Imputation Interface](image-url)
(\*)VCF File: In this field you can see the path of the input file that you selected. You can also use the browser on the right in case you want to change the input file.

(\*)Output File: This is the name and path where you want your output file. We recommend using the same project directory.

Number of Clusters: Maximum number of groups in which local haplotypes will be clustered. See Scheet and Stephens (2006) for details of the HMM implemented in the fastPHASE algorithm. For bi-parental or multi-parental breeding populations, even if the parents are selected from the list, this number should be provided. This option allows taking into account cases of populations in which some of the parents are missing. Default value is 20.

Average centiMorgans per Kbp: Type here an estimate of the average number of centiMorgans per Kbp on euchromatic regions of the genome. This value is used by the model to estimate initial transitions between the states of the HMM. Typical values of this parameter are 0.001 for humans, 0.004 for rice and 0.35 for yeast populations. In future versions, the estimated recombination rate per site should be implemented. Default value is 0.001.

Fixed Transitions: If set, transition probabilities in the HMM will not be updated during the Baum-Welch training phase. We don’t recommend checking this box unless the average number of centiMorgans per Kbp allows a reasonable initial estimation of the transition probabilities.

Parents ID list: Select from this list the parents of the population. This should only be used for bi-parental or multi-parental breeding populations.
**Sample Deconvolution**

This option allows building individual fastq files for different samples from a single file containing the reads for a whole sequencing lane in which several samples were barcoded and sequenced. Up to this point only single read data can be deconvoluted.

**INPUT FILES**

- **FASTQ file:** The original FASTQ file containing several samples sequenced in the same lane, where each sample has a uniquely assigned ID. It must be uncompressed.
- **Index File:** A file containing the information of each sample, its lane, flow cell, barcode and ID.
- **Output Directory:** The directory where you want to store all your separate FASTQ files. It must be previously created.

**OUTPUT FILES**

- **Several FASTQ files:** A FASTQ file for each sample ID present in the index file. Each file will contain all the reads corresponding to an ID. The reads will be trimmed using the information of the restriction site of the nuclease used to prepare the samples.
- **_SD.log:** Information about the process, all the samples present in the file with their barcode, and the amount of reads corresponding to each sample.
(**)Flow Cell:** Select one of the flow cells of the index file. This is an alphanumeric coded ID.

(**)Lane:** Select one of the lanes in the flow cell. Normal Illumina flow cells have 1-8 lanes.

**Trim sequence: (optional)** This optional field can generally be used to trim reads with any sort of partial contamination, retaining only the segment that belongs to the sequenced sample. The sequence included in this field will be searched on each read and, if found, the read will be trimmed to the basepair just before the place where this sequence is found. For example, in some GBS experiments, some of the sequenced fragments could be smaller than the read length and then part of the sequence adapter could be sequenced towards the 3’ end of several reads. The first few basepairs of the adapter sequence could be set here to identify and trim the adapter portion of the read. IUPAC codes can be used to indicate degenerate basepairs.

Output uncompressed files: By default, NGSEP will generate gzip compressed fastq files for each sample, NGSEP is able to handle compressed files when mapping, so leaving files compressed won’t be of trouble ahead. However, the default behavior in Windows is to produce uncompressed files.
Read Depth Comparator

This function implements the algorithm from Xie & Tammy, 2009, for the comparison between two samples of the read depth by window in the reference genome. It takes two alignment files and a reference genome, splits the genome into windows, and for each window compares the read depth between samples.

INPUT FILES

- **Two sorted .bam files**: These files are created as the result of the alignment of short reads to a reference genome (see Map Reads above). These files need to be sorted by chromosome and reference position (if you didn’t create these files with the Map Reads function, see Sort Alignment) in order to be able to compare read depth between them.

OUTPUT FILES

- **List of CNV ratio per window**: A file containing the list of windows (one per line) of the genome that have a significant CNV ratio.
- **RDcompar.log**: Information about the finished process.

If you are willing to find novel CNVs by comparing two samples, for example “test” and “control”, or “ill” and “healthy”, or simply “Sample A” and “Sample B”, you can select both alignment files for your samples (bam files) and choose, from the NGSEP menu, the Read Depth Comparator option.
(*)Sample 1/2 (.bam): These are the bam files to be compared, they will be the ones you selected from the project, but they can be changed in the window to match other bam files. Both bam files must be mapped to the same reference genome.

(*)Reference Genome: This is the same reference genome to which the reads were mapped, and the one that generated the bam files.

(*)Output File: The prefix of the file that will contain the list of windows and the CNV ratio in each one, and the prefix for the plot file.

Window size: This will be the window size to be used during the read depth comparison; larger window sizes may have lower p-values, but will lose resolution.

Perform GC correction: Select if you want to perform a GC correction for each window. Use this if different technologies with different known GC bias generated your reads.

Max p-value to report: This acts as a filter; only the windows with a lower p-value than the one specified will be reported.
Print whole genome output: This eliminates the filter and reports all the bins in the genome. This option may be useful for plotting.

Final Result for Read Depth Comparator:

A file listing one window per line and 7 columns: The chromosome ID, initial and final position of the window, number of reads in each window for sample 1 and sample 2, the raw ratio of reads (sample 1 / sample 2), and the p-value associated with that ratio:

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Start</th>
<th>End</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Raw Ratio</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBS64122</td>
<td>200</td>
<td>1874</td>
<td>4647.75</td>
<td>0.482748187953680769792</td>
<td>2.925890608476783E-13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1000</td>
<td>724.51</td>
<td>519.28</td>
<td>1.6703349176515052</td>
<td>9.30044861892412E-9</td>
</tr>
<tr>
<td></td>
<td>1001</td>
<td>1200</td>
<td>261.33</td>
<td>135.70</td>
<td>2.3893283238379077E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>1400</td>
<td>833.93</td>
<td>198.58</td>
<td>4.283671738419155E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1400</td>
<td>1600</td>
<td>835.45</td>
<td>259.94</td>
<td>2.081447385255878E-10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>1800</td>
<td>496.72</td>
<td>293.56</td>
<td>2.049863685048086E-10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1800</td>
<td>2000</td>
<td>378.25</td>
<td>241.01</td>
<td>1.875008957493562E-12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>2200</td>
<td>325.59</td>
<td>227.62</td>
<td>1.7152237375479493E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2200</td>
<td>2400</td>
<td>319.16</td>
<td>177.04</td>
<td>2.180815309206511E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2400</td>
<td>2600</td>
<td>340.72</td>
<td>222.92</td>
<td>1.83221080631884E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2600</td>
<td>2800</td>
<td>388.67</td>
<td>253.82</td>
<td>1.8362706425579513E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2800</td>
<td>3000</td>
<td>430.63</td>
<td>258.15</td>
<td>2.900366487679444E-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>3200</td>
<td>480.21</td>
<td>219.11</td>
<td>1.975489536853184E-12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3200</td>
<td>3400</td>
<td>395.21</td>
<td>247.58</td>
<td>1.914215622817835E-9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3400</td>
<td>3600</td>
<td>395.21</td>
<td>247.58</td>
<td>1.914215622817835E-9</td>
<td></td>
</tr>
</tbody>
</table>
Sort Alignment

This option sorts the Bam file. This process is required because sequencers such as Illumina, 454 and Sanger among others, produce files that match randomly in the genome. Sort Alignment uses internally Picards Tools, a library which already contains an option for this purpose. The Map Reads and Multi-Mapping options described above automatically perform the sort alignment of the reads. However, if you start from unsorted .bam files, you must use this option to sort them.

INPUT FILE

- **.bam**: This is a tab delimited text file which consists in an optional header section and an alignment section. The header begins with @ while the alignment lines don’t. Each aligned line has 11 optional information fields that make it flexible. For a detailed description of a bam file we suggest the following reference [http://samtools.sourceforge.net/SAMv1.pdf](http://samtools.sourceforge.net/SAMv1.pdf) in the SAM tools web page.

OUTPUT FILES

- **Sorted.bam**: .bam file with all the reads is organized by chromosome and reference position. You are able to visualize the first 100 lines of this file by double clicking on it.

If you have an unsorted .bam file, right click on it and choose the Sort Alignment option from the NGSEP menu. A window will open:
The first field is the (*)File (.bam), this field contains the path of the selected input file; the browser on the right can also be used to change the input file. The next one is (*)Output file: This text field holds the same input name and path of the input, with the addition of the word “_sorted” just before the extension. You can also change the output destiny directory and file name. Our advice is to use the same directory because further processes will require them.

Use the Sort Alignment button to execute.

**Final Result for Sort Alignment:**

At the end of the process you will see a file similar to the input, but organized and ready to continue with the pipeline.
Citing and supporting packages

A manuscript with the description of the latest modules of NGSEP is available in Nucleic Acids Research:


http://nar.oxfordjournals.org/content/42/6/e44.full

Details of variant detection algorithms implemented in NGSEP can be found in the following publications:

**SNV detection**


**CNV detection (Read depth analysis)**


**Genotype imputation**


**Read Depth comparison**


**Large indels detection (Mate-pair analysis)**

We implemented a new model to integrate paired-end and split-read analysis for detection of large indels. Benchmarking with other tools is in progress.

NGSEP is also supported by the following open source software packages:


FAQs

**Why is my VCF empty?**

A quite feasible cause for this situation is due to the usage of unsorted BAM files in processes like Variants Detector among others. **For all functionalities in NGSEP that requires a BAM file, this must be sorted.** Currently the Map reads function can generate the sorted BAM file directly, or you can sort your unsorted BAM file using the Sort Alignment function.

**White spaces issue (Files and directory names)**

Currently some NGSEP functions are presenting problems when output files names or directories contain white spaces. This could throw many kinds of exceptions that you can verify in the log file.

For example

Using the Map Reads option, if your input files look like this:

- File #1: /home/directory05/workspace/Sample71 SupPar/Samplen71_1.fq
- File #2: /home/directory05/workspace/Sample77SupPar/Samplen71 2.fq

You can have the following exception:

```
Exception: Extra parameter(s) specified: "SupPar/Samplen71_1.fq", "2.fq"
```

In this case both parameters are wrong.

For File 1: Sample71(space)SupPar
For File 2: Samplen71(space)2.fq

We are working to solve this problem, for now we recommend to **avoid to use spaces in your output files names and directories.**

**No Java virtual machine was found using Eclipse+Plugin option (Windows).**

One common exception is that after you downloaded the Eclipse+Plugin zip file, and while executing the Eclipse.exe, this message is shown: “A Java Runtime Environment (JRE) or Java Development Kit (JDK) must be available in order to run Eclipse. No Java virtual machine was found ...... javaw.exe in your current path”. This could be due to issues in the Java virtual machine and for almost any cause the solution that we suggest is to reinstall the JRE (Java Runtime Environment) ([http://www.oracle.com/technetwork/es/java/javase/downloads/jre7-downloads-1880261.html](http://www.oracle.com/technetwork/es/java/javase/downloads/jre7-downloads-1880261.html)). Here, you need to choose the Windows x64 (.exe) option (please keep in mind that we only offer Eclipse+Plugin for 64 bit operating systems).
If you are using a 32 bit operating system you must download first the standard Eclipse (http://www.Eclipse.org/downloads/) for 32 bit and then download just the plugin (https://sourceforge.net/projects/ngsep/files/OnlyPlugin/) and paste it in the dropins folder of your Eclipse directory.