

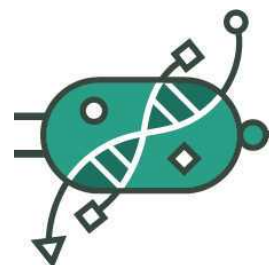
Dec 15, 2023

Version 2

Bacterial genome annotation script using BLASTN V.2

DOI

<https://dx.doi.org/10.17504/protocols.io.dm6gpjrb1gzp/v2>



Ana Mariya Anhel¹, Lorea Alejaldre¹, Ángel oñi-Moreno¹

¹Centro de Biotecnología y Genómica de Plantas, Universidad Politécnica de Madrid (UPM)-Instituto Nacional de Investigación y Tecnología Agraria y Alimentaria (INIA/CSIC), Madrid, Spain

Ángel oñi-Moreno: angel.goni@upm.es



Biocomputation Lab

CNB-CSIC

Create & collaborate more with a free account

Edit and publish protocols, collaborate in communities, share insights through comments, and track progress with run records.

Create free account

OPEN  ACCESS



DOI: <https://dx.doi.org/10.17504/protocols.io.dm6gpjrb1gzp/v2>

Protocol Citation: Ana Mariya Anhel, Lorea Alejaldre, Ángel oñi-Moreno 2023. Bacterial genome annotation script using BLASTN. [protocols.io https://dx.doi.org/10.17504/protocols.io.dm6gpjrb1gzp/v2](https://dx.doi.org/10.17504/protocols.io.dm6gpjrb1gzp/v2) Version created by **Biocomputation Lab**



License: This is an open access protocol distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Protocol status: Working

We use this protocol and it's working

Created: November 20, 2023

Last Modified: December 15, 2023

Protocol Integer ID: 91173

Keywords: Genome anotation, Bacterial, P. putida, Transposon, Transposon library, E. coli, bacterial genome annotation script, microbial bacterial genomes of pseudomonas putida kt2440, microbial bacterial genome, other bacterial genome, pseudomonas putida kt2440, results from genome amplification, genome amplification, genome sequence, gene locus of transposon insert, using blastn, read sequencing, sequencing result, blastn this protocol, line blastn, transposon insert, sequencing, blastn version, gene locus

Funders Acknowledgements:

Comunidad de Madrid

Grant ID: Y2020/TCS-6555,2019-T1/BIO-14053

MCIN/AEI

Grant ID: CEX2020-000999-S,PID2020-117205GA-I00

ERC

Grant ID: 101044360

Abstract

This protocol uses a python based script and command-line BLASTn to annotate in a final table single-read sequencing results from genome amplifications, within other output files.

Its main use in our lab (<https://biocomputationlab.com>) is to identify the location and gene locus of transposon inserts in microbial bacterial genomes of *Pseudomonas putida* *KT2440*. However, this script can be used for other bacterial genomes for which its genome sequence and annotation are available.

Script was developed and tested in python 3.9.5 with blastn version 2.9.0, sickle version 1.33 and fastqc version 0.11.9

This is a description of the LAP entry LAPu-InsertsGenAnnotation-1.0.0 located in the [LAP repository](#)

Guidelines

This script needs min 4 arguments in the following order:

1. Directory of folder containing sequencing reads
2. Reads file type (should be FASTA format even if the extension can be anything)
3. Genome file to perform blastn alignment (FASTA format)
4. Genome annotation file (.csv)

This program, by the time this guide was developed, can only be executed with Linux and macOS systems



Materials

Software

- Linux or MacOS
- Python 3.9.5
- Python packages: pandas (v2.0.0), os, subprocess, argparse, re and biopython (v1.81)
- BLAST+
- Sickle
- FastQC

Before start

To run this script command-line blastn and python 3 with packages pandas (v2.0.0), os, subprocess, argparse, re and biopython (v1.81)

Adquisition of files

1 Download reference genome file

You can download the genome of the organism that you want to compare to the reading sequences from different sources such as NCBI, GSA or even pages dedicated to the organism (i.e pseudomonas.com).

For this script to work, genome files need to be in **FASTA format** (.fasta, .fna, .ffn, .faa, .frn). Sequence alignment is based in BLASTn which requires **FASTA format** as input.

2 Download annotation file

You can download the annotation of the genome from different sources, such as NCBI, KEGG or pages dedicated to the organism.

For this script to work, the annotation file needs to be in CSV format and need to have at least the following columns with the exact names (names in **bold letter**):

- **Start** - Nucleotide number that sets the beginning of the annotated region
- **End** - Nucleotide number that sets the ending of the annotated region
- **Locus Tag** - Identifiers that are systematically applied to every gene in a genome. It has to be a unique name

Note

If the annotated file does not have those specific columns, but it has the information, the user can change the name of the columns to run the script because it should have **exactly those names**

Note

In case your annotated genome is not a CSV file but either a TSV, GFF or GTF, there are software that can convert those types of files to CSV:

- **GFF to GFT**: there are several convertors, such as *gffread* and *AGAT*. Some other converters can be found at https://agat.readthedocs.io/en/latest/gff_to_gtf.html
- **GFT to CSV**: *gft2csv* (<https://github.com/zyxue/gtf2csv>)
- **TSV to CSV**: there are a lot of online converters and Python packages that you can use to convert TSV to CSV, such as *pandas*

In future versions, those types of annotation files will be accepted, but at the time of development of this documentation only CSV files are valid

3 Download sequencing files

Sequencing files should be contained in a folder. Depending on the sequencing company, sequencing files will be in .txt, .seq, .ab1 or other format. This should be specified within the script.

Note

The sequences that sequencing companies provide usually come with other files that give the quality of those reads. If we also have these files (.ab1, .fastqc, .abi, etc), the user of this program will be able to produce results that are more trustworthy by providing these quality files to the script

4 Optional: Acquisition of map of reads for annotation of variants

The identity of each variant can be annotated by using this script if a map of the 96-well plate is provided. It can be created by hand or can be an output file of other scripts, such as LAP-PCR-OT2-1.0.0.

This map should be a CSV file that will contain the names or identities of the plate that corresponds to the reads, for instance, the identities of the plate sent to sequence in the corresponding well. It needs to have the name of the rows and columns of the plate and can only be used if the reads and map fulfil the following requirements:

1. There is only **1 map**
2. **All the reads of the directory can be tracked in that map**, i.e., the directory of reads should only include the ones in the map. An example, if the map has 12 columns and 8 rows and it is half full (48 identities), the directory of the reads cannot be more than 48 sequences.
3. **All sequences need to have an identity in the map**, i.e., the cell correspondent to a well in the map of a sequence cannot be left empty.
4. The names of the sequence files need to have the name of the cell **between underscores if the extension of the reads is txt** (for example, *readSequence_A10_example.txt*) or **between a plus and an underscore if the extension is seq** (for example, *readSequence+A10_example.seq*). Other types of extensions are not available with this function.

An example of this script with the map provided will be given at the end of this page (section Example 2)





5 Script

The last script version can be found at

<https://github.com/BiocomputationLab/LAPrepository/tree/main/LAPuEntries> (the name of this file is the user's choice). The name of the directory should be **LAPu-InsertsGenAnnotation** followed by the version.

You can also find the latest version of the script in

<https://www.laprepo.cbgp.upm.es/repository/> with the same name as in GitHub.

Software

LAP Repository

NAME

<https://biocomputationlab.com/>

DEVELOPER

www.laprepo.com

REPOSITORY

Note

LAPu-InsertsGenAnnotation-1.0.0 is only available to run in Linux and MacOS systems but efforts are being performed to make the following versions available for Windows systems as well.

For instructions on how to use this script in Windows, most of these instructions can be used, but for more specific ones, you should follow the instructions attached to the LAPu entry in www.laprepo.com

Preparing System

- 6 If you are using a **Windows system** and the **LAPu-InsertsGenAnnotation-1.0.0** you can install a Windows Subsystem for Linux (WSL) and run the script in that subsystem the same way a Linux user would do, but be aware of the nomenclature and path of the files

There are other ways to run a Unix system in Windows, like a virtual machine (<https://www.virtualbox.org/>)



You can find instructions to install a WSL on the following Windows page:

<https://learn.microsoft.com/en-us/windows/wsl/install>

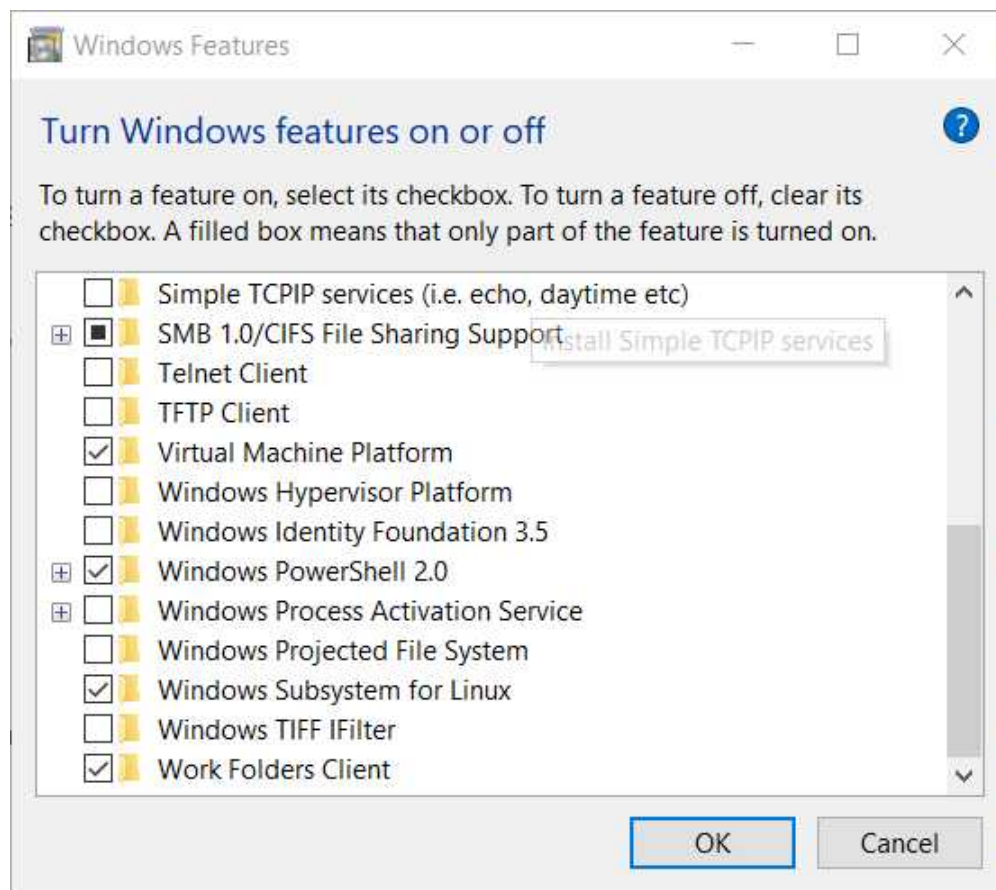
In the WSL, you can install the different requirements needed for the script to run

I will provide an example of how to install a WSL in the following sub-steps

6.1 *Make sure the Windows Subsystem in Linux Feature is activated*

Windows search bar → Apps & Features → (Scroll Down) Programs and Features → Turn Windows features on or off

In that window, you will need to have a tick in the option "Windows Subsystem for Linux" you may need to restart the computer after ticking that box so changes are made in your computer



Window with the WSL (Windows Subsystem for Linux) feature ON

6.2 *Install a Linux system*



Open a window with Windows Powershell, and you can check which distributions of Linux can be installed

Command

Check (Windows 10)

Check WSL Linux Distributions

```
wsl --list --online
```

To install one of the distributions, you can perform the following command by changing *Ubuntu-20.04* for the desired Linux system.

Command

Install Ubuntu 20.04 with the wsl command (Windows 10)

Install Ubuntu-20.04 WSL

```
wsl --install Ubuntu-20.04
```

You enter the needed data that the system will ask you, such as the UNIX username and password

```

C:\Users\Ava (RDP) > [root@ubuntu 20.04]
Installing Ubuntu 20.04 LTS
Se ha instalado Ubuntu 20.04 LTS.
Iniciando Ubuntu 20.04 LTS...
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
Go now information visit: http://aka.ms/wslusers
Enter new UNIX username: user-chgp
New password:
Retype new password:
passwd: password updated successfully
Installation successful
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Tue Nov 28 14:09:04 CET 2023

System load:  0.7%           Processes:           66
Usage of /:   0.1% of 1000.85GB  Users logged in:    0
Memory usage: 11%            IPNet address for eth0: 172.27.293.176
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo apt install

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

This message is shown once a day. To disable it please create the
/home/user-chgp/.hushlogin file.
user-chgp@ubuntu:~$

```

PowerShell screen of the installation of an Ubuntu system with wsl

6.3 Make the distribution run as default

You can make the system installed in **step 6.2** the one that is going to run when you type the command `wsl` in the PowerShell by running the following command

Command

Set the distribution installed as default for wsl (Windows 10)

Set default distribution WSL

```
wsl --set-default Ubuntu-20.04
```

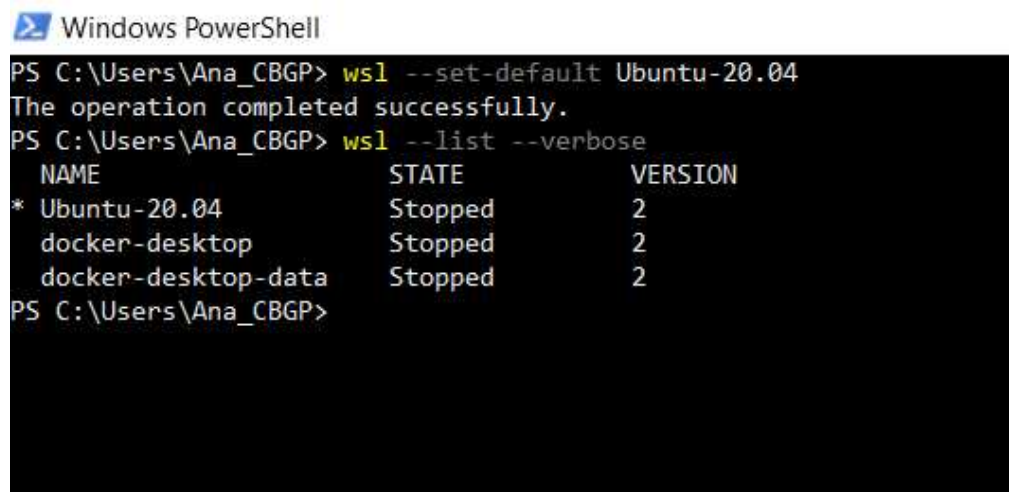
You can check which distribution is running as a default with the following command, will be the one designed or marked by an asterisk

Command

Check distributions installed WSL (Windows 10)

```
wsl --list --verbose
```

Expected result



```
Windows PowerShell
PS C:\Users\Ana_CBGp> wsl --set-default Ubuntu-20.04
The operation completed successfully.
PS C:\Users\Ana_CBGp> wsl --list --verbose
  NAME                STATE      VERSION
* Ubuntu-20.04        Stopped    2
  docker-desktop       Stopped    2
  docker-desktop-data  Stopped    2
PS C:\Users\Ana_CBGp>
```

Result of setting the Ubuntu-20.04 system as a default when running the wsl program

6.4 Run Linux System

To run the Linux system, in this example Ubuntu 20.04, you can type in the Windows PowerShell **wsl** and you will directly access the directory where you have typed this command, but in the Ubuntu system

Now you can perform the script and install the needed packages and programs in this system

7 Install Python

In most Unix systems, a default Python is installed, but you can always install more than 1 Python version on your computer

This script was developed and tested with Python 3.9.5, so it cannot be guaranteed that it works as expected in previous or later ones

In the following substeps, I will show how to install Python and set it as the default one to use in case more than 1 python version is installed on the computer.

7.1 *Linux systems*

You can install a specific version of this language with the following command. If you do not provide a version, the latest one accessible to sudo will be installed

Command

Install Specific Version Python (Linux)

```
sudo apt install python<version>
```

You can use this python-specific version by using the command `python<version>`
If you want a specific version to be the default python that will be used, you will need to change the aliases of the system.

Command

Add aliases to Linux system (Linux)

```
nano ~/.bashrc
```

Go to the last line, add the following line: **alias python='python'**, and then use *Ctrl+x* to get out. Do not forget to save before leaving.

After that, you should reload the `.bashrc` file by running the following command

**Command****Update .bashrc file (Linux)**

```
source ~/.bashrc
```

7.2 macOS systems

Note

We are going to use Homebrew to install it, so if you do not have it yet, you can install it by typing the following command in the command line

Command**Install Homebrew (macOS)**

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh  
)"
```

For more information about how to install this software, you can check the homebrew page (<https://docs.brew.sh/Installation>)

You can install a specific version of Python by typing in the command line and the following instructions. If a version is not provided, the latest python accessible in brew will be installed.

Command

Install Specific Version Python with Homebrew (macOS)

```
brew install python@<version>
```

You can use this python-specific version by using the command `python<version>`
If you want a specific python version to be used by default, you will need to change the aliases of the shell that you are using.

Command

Add aliases to macOS system in BASH (macOS)

```
nano ~/.bashrc_profile
```

Go to the last line and add the following line **alias python='python<version>'** and then use `Ctrl+x` to get out. Do not forget to save before leaving.

After that, you should reload the `.bashrc_profile` file by running the following command.

Command

Update `.bashrc_profile` file (macOS)

```
source ~/.bashrc_profile
```



Note

**If you are using Bash, you have to change the file `~/.bashrc_profile` or `~/.bashrc`
In case you are using Zsh, you need to change the file `~/.zshrc`**

Knowing which file you need to change follow the previous commands changing the file name to the correct one.

8 Install needed Python packages

This script needs the following packages: *sys*, *os*, *argparse*, *re*, *subprocess*, *pandas* and *BioPython*.

The first 5 packages are installed by default in Python but the latter 2 need to be installed

To install these packages you can perform the following command

Command

Install packages that are inside of the pip packages database

Install package Python

```
python<version> -m pip install <name_package>
```



Note

Sometimes pip is not installed by default, you can install it with the following commands

Command

Install pip for Python 3 (Linux)

```
sudo apt install python3-pip
```

Command

Install pip for Python 3 (macOS)

```
sudo easy_install pip
```

9 Install BLAST+

There are different ways to install BLAST, here we provide 3 ways in the substeps

9.1 *Download the executable from NCBI*

You can download and install the BLAST from the NCBI web (<https://www.ncbi.nlm.nih.gov/>)

Download → FTP → blast → executables → blast+ → [wanted version] → *ncbi-blast-[version]-[system].tar.gz*

Unzip it with the following command

Command**Unzip tar.gz file**

```
tar -xf [name_file].tar.gz
```

In the unzip folder, you have a bin folder that contains blastn

If you decide to install it this way, you need to run the Python script in this directory so it can be accessed by the Python file or added to the path of the system directory

9.2 *Use Anaconda*

You can install the needed commands with Anaconda if you perform the following command

Command**Install blast with Anaconda**

```
conda install -c bioconda blast
```

Note

For this option, you need to have Anaconda installed in your system

Instructions to install this software can be found on the following page

<https://docs.anaconda.com/free/anaconda/install/index.html>

9.3 *Use apt or brew install*

If you are in a Linux system, you can perform the following command to install blast+

**Command****Install BLAST+ (Linux)**

```
sudo apt install ncbi-blast+
```

In case you are in a macOS system, you can install it with the following instruction

Command**Install BLAST+ (macOS)**

```
brew install blast
```

If you do not have Homebrew in your system [➡ go to step #7.2](#) **Note** to install it

- 9.4 To make sure it has been installed in your system, you can go to the command line and type the following command *

Command**Check Version of BLASTn Command Line**

```
blastn -version
```

If this command does not raise an error, it means that BLASTn can be used from the command line

- 10 **Install FastQC** *



FastQC is a tool used for quality control of high-throughput sequence data. It provides a way to assess the quality of raw sequencing data

If you will not use the quality assessment of the script, there is no need to install this software

In the following sub-steps, the installation in different OS is provided

10.1 *Linux*

In Linux, you can install with apt the command line executable of fastqc. Just type in Bash the following command

Command

Install FastQC from command line (Linux)

```
sudo apt install fastqc
```

You can check if the installation has been successful if you type **fastqc --version** and you don't receive an error message

10.2 *MacOS*

In macOS you can install with homebrew the command line executable of fastqc, just type in Bash the following command

Command

Install FastQC from command line (macOS)

```
brew install fastqc
```

If you do not have Homebrew in your system [➡ go to step #7.2](#) Note



You can check if the installation has been successful if you type **fastqc --version** and you don't receive an error message

11 Install Sickle



Sickle is a software tool designed for quality control of high-throughput sequence data, especially for data generated by Next-Generation Sequencing (NGS) platforms. Its primary use case is trimming low-quality bases and adapter sequences from the ends of sequencing reads

If you will not use the quality assessment of the script, there is no need to install this software

In the following sub-steps, the installation in different OS is provided

11.1 Linux

In Linux you can install with apt the command line executable of sickle, just type in Bash the following command

Command

Install Sickle from command line (Linux)

```
sudo apt install sickle
```

You can check if the installation has been successful if you type **sickle --version** and you don't receive an error message

11.2 MacOS

In macOS you can install with homebrew the command line executable of sickle just type in Bash or Zhr the following command



Command

Install Sickel from command line (macOS)

```
brew install sickel
```

If you do not have Homebrew in your system [➞ go to step #7.2](#) Note

You can check if the installation has been successful if you type **sickel --version** and you don't receive an error message

Running Script

12 Choose which arguments to run the script

This script needs to be performed in a command line window using Python.

Depending on the provided arguments, the program will perform more or less actions such as a quality trimming or variant name annotation.

The script requires a minimum of 4 arguments, which we will call from now on the **positional arguments**. In addition, the program's behaviour can be changed by providing **optional arguments**. Finally, there is another type of argument that the script can accept, which is for **information usage**.

In the following sub-steps, we will define the behaviour, needs and different kinds of arguments.

12.1 *Positional arguments*

These arguments should be provided in this specific order

directoryReads

Absolute or relative path to the folder that will have the file(s) of the different sequences with a FASTA format and, optionally, the quality files of those reads.

Both files, sequence and quality, should have the same name but with a different extension.

More files can be in that folder but will be ignored.

extensionReads

Extension of the files (without the dot, i.e txt or seq) in the path provided in `directoryReads` that correspond to the sequences in FASTA format that you want to be aligned to the genome provided in *genomeSequence*.

Ensure they are the only files with this extension in that directory.

genomeSequence

Absolute or relative path to the file that will contain the genome sequence or DNA material that will be aligned with the files inside of *directoryReads*.

genomeAnnotation

Absolute or relative path to the file that contains the genome provided in *genomeSequence* annotated and needs to be in a CSV format with the characteristics noted in *Step 2*.

12.2 *Optional arguments*

These arguments do not need to be provided in this order, but some of them need to be provided together.

-out

Absolute or relative path to where the final files will be stored. If not provided, the output files will be stored in a directory called *results_annotation* in the place where the script has been run.

If the directory already exists, the program will display a warning message allowing you to stop the program from running. If you choose the program to continue, this directory will be overwritten.

-f, -filesOut

Depending on the arguments provided, you will obtain different types and numbers of files in the -out directory.

With this argument, you can control whether you obtain a **SAM file** coming from the alignment or not.

This argument only can take 2 values:

- *all* - you will obtain the SAM file and the table provided from annotating the alignments. This is the value as the default
- *table* - you will only obtain the table. No SAM file will be provided in the results directory

-t, --thresholdRange

This script takes into account the bitscore to give you the best hit.

An additional column will provide other hits in case there is a multiple alignment of the query sequence with your genome. These will be provided only if the hits are in the range of this argument, i.e, if the hit has a value minor than $(1 - \text{thresholdValue}) * \text{Bit Score of the best hit}$, this value will not be included in the multiple hit column. If more than 1 alignment has the same best score and turns out that this score is the best one, the hit provided as the best one will be randomly assigned between these top hits.

By default, this value is 0.01

Note

An example of the effects of this value

We have a sequence (seq_1) which best hits with locus_1 and a bit score of 50. seq_1 has 3 other hits, with locus_2, locus_3 and locus_4, with bit scores of 50, 45 and 10, respectively.

If we have **-t 0.01**, the **minimum bit score** that a hit needs to have to be considered a hit and added to the multiple alignment column **would be 49.5**. With this value of -t, only locus_2 would be considered as a hit.

If we have **-t 0.1**, the **minimum bit score** that a hit needs to have to be considered a hit and added to the multiple alignment column **would be 45**. With this value of -t, only locus_2 and locus 3 would be considered as a hit

If we have **-t 0.5**, the **minimum bit score** that a hit needs to have to be considered a hit and added to the multiple alignment column **would be 25**. With this value of -t, only locus_2 and locus_3 would be considered as a hit.

If we have **-t 1**, the **minimum bit score** that a hit needs to have to be considered a hit and added to the multiple alignment column **would be 0**, so all hits would be considered and locus_2, locus_3 and locus_4 would be added to the multiple alignment column.

-identity

The absolute or relative path of the file that has the names that want to be associated with the final rows that indicate the alignments.

This file should be a CSV file and needs to have the requirements or characteristics described in Step 4.

-cb, --columnsBLAST

This variable will be the absolute or relative path of the file with the names of the columns that will be reflected in the final table that are taken from the BLASTn output alignment. There should be 1 name of column per row in the file, and the name should be the same as will be named in the BLAST software, for example, *nident*.

The final table combines columns that are in the annotation file and columns that we obtain from the BLASTn alignment. With this variable, you can control which columns will be taken from the final alignment result file and will be written on the end table.

The default columns that are going to be taken have the following names: *qaccver*, *saccver*, *pident*, *length*, *mismatch*, *gapopen*, *qstart*, *qend*, *sstart*, *send*, *evaluate*, *bitscore* and *sstrand*

The meaning of these names can be found at

<https://www.metagenomics.wiki/tools/blast/blastn-output-format-6>

-ca, --columnsAnnotation

This variable will be the absolute or relative path of the file with the names of the columns that will be reflected in the final table that are taken from the file stated in the argument *genomeAnnotation*. There should be 1 name of column per row in the file.

The final table combines columns that are in the annotation file and columns that we obtain from the BLASTn alignment. With this variable, you can control which columns will be taken from the annotation file and written on the end table.

The default columns will be taken with the following names: *Locus Tag*, *Feature Type*, *Start*, *End*, *Strand*, *Gene Name*, *Product Name* and *Subcellular Localization [Confidence Class]*.

If your file does not contain 1 of these columns, you need to provide which columns you want to have in the end file with this argument, always considering the considerations provided in Step 2.

-quality

Extension (without dot i.e ab1) of the quality files attached to the sequence files in the directory given in *directoryReads*. By providing this argument, a quality check and consequent trimming of the sequences will take place before doing the alignment between reads and genome.

Only sequences that come from Sanger, Solexa or Illumina sequencing can be provided to the program. In addition, only single-end reads with *fastqc*, *ab1*, *abi*, or *qual* formats can be analyzed.

If we provide this argument, we must also provide the *-seq* argument.

Providing this argument will allow you to trim the sequences to only align with high-quality nucleotides with the genome in BLASTn. For that, a FastQC analytic HTML will be provided, and the user will decide the Q (**quality**) and length for trimming. To provide these 2 variables, you can type the numbers directly in the command window.

By default, both values would be 20 (if nothing is typed in the window and only enter is pressed)

-seq

Method of sequencing with which the sequences in the directory *directoryReads* have been sequenced.

The following arguments are accepted: *illumina*, *sanger* or *solexa*

12.3 *Information usage*

These arguments give you information but do not change the behaviour of the program.

-h, --help

This argument should be provided alone, without any other arguments.

If you provide this argument information about the program will be displayed, including how to use it and what arguments are available.

-q, --quiet

If this argument is given, minimum information will be displayed in the window while running the script.



This argument is incompatible with -v

-v, --verbose

If this argument is given, more information will be displayed in the window while performing the script than in a run where this argument is not given.

This argument is incompatible with -q

13 **Running script**

The final command should look like the following one

Command

Command to run blastn annotation script

```
python alignment_and_annotation_blastn.py [directory of sequencing  
reads] [type of file] [genome file in fasta format] [annotation file  
in csv format]
```

14 **Interact with the script**

Depending on the input and the state of the folder where the program has been executed, the program can ask for different kinds of interactions:

- Ask if you want to replace the final result folder
- Ask for the quality of trimming
- Ask for the length of trimming

15 **See results**

Depending on the input, context of the run and arguments, the program can give different outputs:

- *genomeSequence DB files (.nhr, .nin and .nsq)* - In case the organism database needed to perform a BLAST is not in the path where the genomeSequence file is,



these files will be created in the same directory as the genome sequence provided is located

- **-out directory (by default results_script_blast)** - folder where the results will be stored, and it can contain the following folders and files:
 1. **reads_fastq** - if the -quality argument is provided, this folder will have the fastq files of the sequences provided in the directoryReads. If it is not provided, this folder will be empty.
 2. **all_reads_merged.fasta** - All the sequences provided in directoryReads merged in a FASTA format.
 3. **all_reads_merged_quality.fastq** - All the sequences provided in directoryReads with their respective quality merged in a FASTQ format. For more information on the FASTQ format, you can start reading the [dedicated entry in wikipedia](#) which provides a good starting explanation of the file.
 4. **all_reads_merged_quality_fastqc.html** and **all_reads_merged_quality_fastqc.zip** - Quality analysis report of the sequences provided in directoryReads done with the software FastQC and should be checked before assigning the Q and length of trimming that will guide the dynamic trimming of the sequences done by Sickel. Both files have the same information.
 5. **all_reads_merged_quality_trimmed.fastq** - Sequences contained in all_reads_merged_quality.fastq trimmed based on the Q and length trimming variables provided by the user with their respective quality.
 6. **all_reads_merged_trimmed.fasta** - Sequences contained in all_reads_merged_quality.fastq trimmed based on the Q and length trimming variables provided by the user.
 7. **all_seq_aligned.sam** - SAM format output file of the BLAST done with the sequences provided. For more information about the SAM output, you can check the following page <https://www.metagenomics.wiki/tools/samtools/bam-sam-file-format>
 8. **all_seq_aligned.tsv** - Tabular output file of the BLAST done with the sequences provided with the name of each column and an alignment for each row.
 9. **table_reads_gene_description.csv** - Final table that will combine the columns selected from the annotation file and the columns selected in the BLAST output file in addition to columns providing info if there is a multiple alignment of that sequence to the genome and the hits (considering the value in the rangeThreshold argument). If the -map argument is provided, another 2 columns will be added with the position that the sequence holds in the map and the Identity that this has. In other words, it will hold the value that is in the cell on the file provided in the argument -map.

Example 1

12m

16 **Annotation of sequencing results from *P. putida* KT2440 with only positional arguments**

This has been done in a Windows 10 with a WSL Ubuntu 20.04



17 Acquisition of files

4m

17.1 Script

1m

Downloaded the script from the entry **LAPu-InsertsGenAnnotation-1.0.0** from [LAP repository](#) and re-named *annotation_DNAinserts.py*

 annotation_DNAinserts.py 35.6KB

17.2 Genome and Annotation Files

2m

Both files have been obtained from [Pseudomonas Genome DB](#), specifically from one of the [entries of the Pseudomonas putida KT2440 organism](#)

 Pseudomonas_putida_KT2440_110.... 6MB

 Pseudomonas_putida_KT2440_110.... 970.3KB

17.3 Reads

2m

Files return from the Sanger sequencing in the company [Stab Vida](#) of a plate coming from another protocol from this page "[High-throughput workflow for the genotypic characterization of transposon library variants](#)"

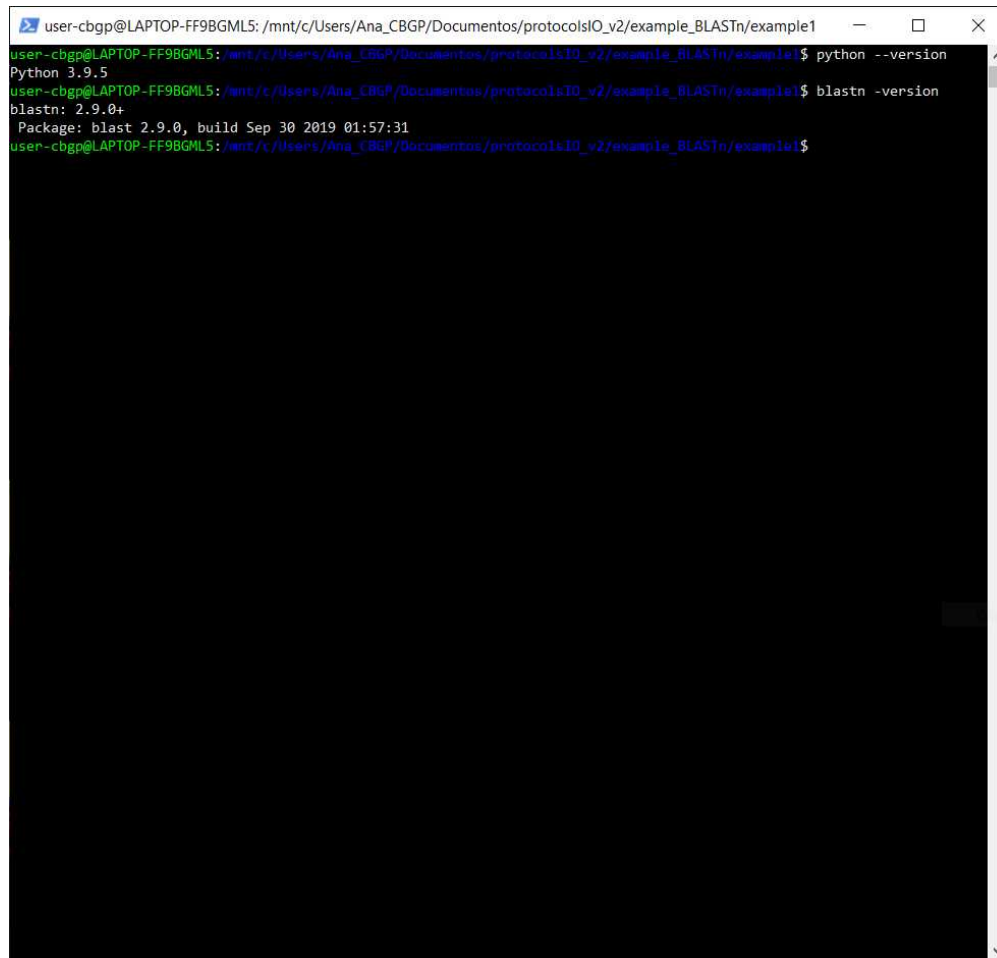
 sequencing_results_example.zip 23.4MB

18 Check requirements are meant in the system the script is going to be run

1m

Check if we have Python and BLAST installed

Checking if asking in the command line for the version of Python and blastn gives me an error or not



```
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGp/Documents/protocolsIO_v2/example_BLASTn/example1
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGp/Documents/protocolsIO_v2/example_BLASTn/example1$ python --version
Python 3.9.5
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGp/Documents/protocolsIO_v2/example_BLASTn/example1$ blastn -version
blastn: 2.9.0+
Package: blast 2.9.0, build Sep 30 2019 01:57:31
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGp/Documents/protocolsIO_v2/example_BLASTn/example1$
```

Output in the command line of Python and BLASTn versions

19 Choose arguments to give

5m

We are going to leave all the optional arguments with the default values and just give the positional ones, in other words, the minimal amount of inputs

20 Run script

2m

We are going to run the script in the same folder as every file that we are going to give to the script, so we are going to give relative paths as we can see in the image as we can see the files that the folder contained before and after executing the script

Commands that show the state of the folder (ls -l) and the run of the Python script

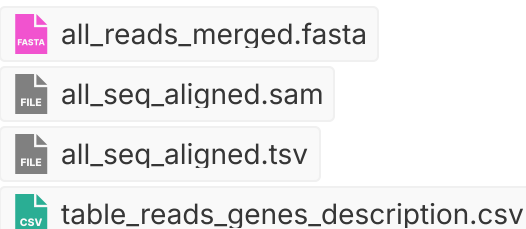


Expected result

As we can see in the figure, the files for the database to do the alignments were created, and *results_script_blast* folder was created as well

```
user-chgg@LAPTOP-FF9BQML5: /mnt/c/Users/Ana_CBG/Documentos/protocolsIO_u2/example_BLAST/example1/results_script/reads_fastq$ ls -l
total 364
-rwxr-xr-x 1 user-chgg user-chgg 144274 Nov 30 13:04 all_reads_merged.fasta
-rwxr-xr-x 1 user-chgg user-chgg 39641 Nov 30 13:04 all_seq_aligned.sam
-rwxr-xr-x 1 user-chgg user-chgg 37222 Nov 30 13:04 all_seq_aligned.tsv
-rwxr-xr-x 1 user-chgg user-chgg 512 Nov 30 13:04 README
-rwxr-xr-x 1 user-chgg user-chgg 15519 Nov 30 13:04 table_reads_genes_description.csv
user-chgg@LAPTOP-FF9BQML5: /mnt/c/Users/Ana_CBG/Documentos/protocolsIO_u2/example_BLAST/example1/results_script_blast$ cd reads_fastq/
user-chgg@LAPTOP-FF9BQML5: /mnt/c/Users/Ana_CBG/Documentos/protocolsIO_u2/example_BLAST/example1/results_script_blast/reads_fastq$ ls -l
total 0
user-chgg@LAPTOP-FF9BQML5: /mnt/c/Users/Ana_CBG/Documentos/protocolsIO_u2/example_BLAST/example1/results_script_blast/reads_fastq$
```

Content of the folder *results_script_blast* created by the alignment program



The folder *reads_fastq* is empty because we have not provided the argument -quality

Example 2

16m

21 Annotation of sequencing results from *P. putida* KT2440 with positional and optional arguments

This has been done in a Windows 10 with a WSL Ubuntu 20.04

22 Acquisition of files

8m

22.1 Script



1m

Downloaded the script from the entry **LAPu-InsertsGenAnnotation-1.0.0** from [LAP repository](#) and re-named *annotation_DNAinserts.py*

 annotation_DNAinserts.py 35.6KB

22.2 *Genome and Annotation Files*

2m

Both files have been obtained from [Pseudomonas Genome DB](#), specifically from one of the [entries of the Pseudomonas putida KT2440 organism](#)

 Pseudomonas_putida_KT2440_110.... 6MB

 Pseudomonas_putida_KT2440_110.... 970.3KB

22.3 *Reads*

1m

Files return from the Sanger sequencing in the company [Stab Vida](#) of a plate coming from another protocol from this page "[High-throughput workflow for the genotypic characterization of transposon library variants](#)"

 sequencing_results_example.zip 23.4MB

22.4 *Map*

1m

A file returned from the running of another protocol from this page, "[OT-2 PCR sample preparation protocol](#)"

This map corresponds to the same plate sent to the sequencing company and the files obtained from it.

 map_identity_plate.csv 2.8KB

22.5 *Files with columns selected*

3m

In the final table, we want to have the following columns

- **From annotation file:** Locus Tag, Start, End, Strand, Gene Name, Molecular Weight (predicted)
- **From BLAST output file:** qaccver, evalule, bitscore

 columns_selected_annotation.txt 70B

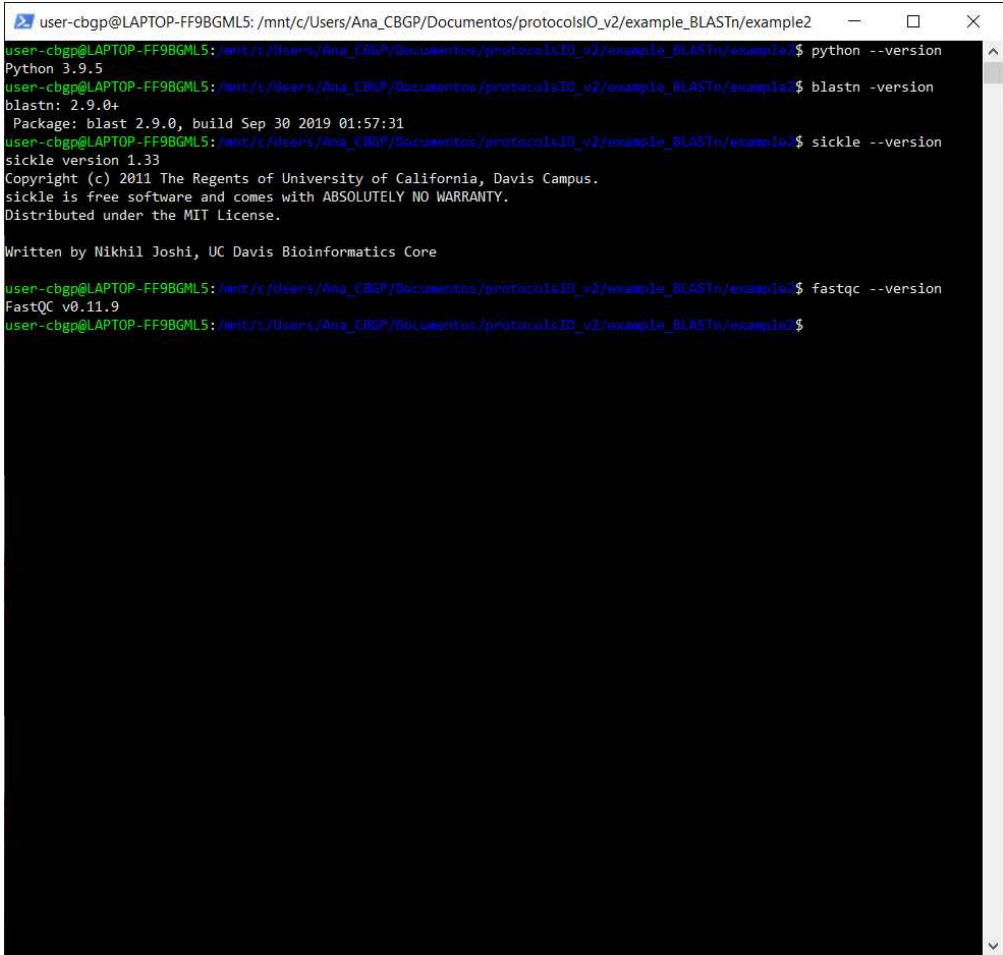
 columns_selected_blast.txt 25B

23 Check requirements are meant in the system the script is going to be run

1m

Check if we have Python, BLAST, sickle and FastQC installed

Checking if asking in the command line for the version of python, blastn, sickle, and fastqc gives me an error or not



```
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocolsIO_v2/example_BLASTn/example2$ python --version
Python 3.9.5
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocolsIO_v2/example_BLASTn/example2$ blastn -version
blastn: 2.9.0+
Package: blast 2.9.0, build Sep 30 2019 01:57:31
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocolsIO_v2/example_BLASTn/example2$ sickle --version
sickle version 1.33
Copyright (c) 2011 The Regents of University of California, Davis Campus.
sickle is free software and comes with ABSOLUTELY NO WARRANTY.
Distributed under the MIT License.

Written by Nikhil Joshi, UC Davis Bioinformatics Core
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocolsIO_v2/example_BLASTn/example2$ fastqc --version
FastQC v0.11.9
user-cbgrp@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocolsIO_v2/example_BLASTn/example2$
```

Output in the command line of Python, BLASTn, Sickle and FastQC versions

24 Choose arguments to give

5m

We want to have a verbose screen output (-v), we want the result directory to be called results_example2 (-out results_example2), we want only the table, not the SAM file (-f table), we want customized columns in the final table (-ca columns_selected_annotation.txt -cb columns_selected_blast.txt), we want that the range threshold on the bit score is 0.2 (-t 0.2), that we have a quality trimming (-quality ab1-seq sanger) and the alignments to be tracked to the map of samples (-identity map_identity_plate.csv)

25 Run script

We are going to run the script in the same folder as every file that we are going to give to the script, so we are going to give relative paths as we can see in the image as we can see the files that the folder contained before and after executing the script

[illegible]

Commands that show the state of the folder (ls -l) and the run of the Python script

As we can see, because we have given the argument -quality so we need to interact with it. After opening the file all_reads_merged_quality_fastqc.html it is decided to put as a threshold Q = 20 and length = 19, so 2 reads have been discarded from the sequencing.

Expected result

```
user-cbpg@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocols.io/v2/example_BLASTn/example2/results_example2/reads_fastq$ ls -l
total 1844
-rwxrwxrwx 1 user-cbpg user-cbpg 144274 Nov 30 14:33 all_reads_merged.fasta
-rwxrwxrwx 1 user-cbpg user-cbpg 284240 Nov 30 14:33 all_reads_merged_quality.fastq
-rwxrwxrwx 1 user-cbpg user-cbpg 775085 Nov 30 14:33 all_reads_merged_quality_fastqc.html
-rwxrwxrwx 1 user-cbpg user-cbpg 501378 Nov 30 14:33 all_reads_merged_quality_fastqc.zip
-rwxrwxrwx 1 user-cbpg user-cbpg 95058 Nov 30 14:34 all_reads_merged_quality_trimmed.fastq
-rwxrwxrwx 1 user-cbpg user-cbpg 48516 Nov 30 14:34 all_reads_merged_trimmed.fasta
-rwxrwxrwx 1 user-cbpg user-cbpg 11518 Nov 30 14:34 all_seq_aligned.tsv
drwxrwxrwx 1 user-cbpg user-cbpg 512 Nov 30 14:33 reads_fastq_compressed
-rwxrwxrwx 1 user-cbpg user-cbpg 10188 Nov 30 14:34 table_reads_genes_description.csv
user-cbpg@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocols.io/v2/example_BLASTn/example2/results_example2/reads_fastq$ cd reads_fastq/
user-cbpg@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocols.io/v2/example_BLASTn/example2/results_example2/reads_fastq$ ls
22CCRAA000_A001_premix.fastq 22CCRAA000_B09_premix.fastq 22CCRAA000_D05_premix.fastq 22CCRAA000_F01_premix.fastq 22CCRAA000_G00_premix.fastq
22CCRAA000_A02_premix.fastq 22CCRAA000_B10_premix.fastq 22CCRAA000_D06_premix.fastq 22CCRAA000_F02_premix.fastq 22CCRAA000_G10_premix.fastq
22CCRAA000_A03_premix.fastq 22CCRAA000_B11_premix.fastq 22CCRAA000_D07_premix.fastq 22CCRAA000_F03_premix.fastq 22CCRAA000_G11_premix.fastq
22CCRAA000_A04_premix.fastq 22CCRAA000_B12_premix.fastq 22CCRAA000_D08_premix.fastq 22CCRAA000_F04_premix.fastq 22CCRAA000_G12_premix.fastq
22CCRAA000_A05_premix.fastq 22CCRAA000_C01_premix.fastq 22CCRAA000_D09_premix.fastq 22CCRAA000_F05_premix.fastq 22CCRAA000_H01_premix.fastq
22CCRAA000_A06_premix.fastq 22CCRAA000_C02_premix.fastq 22CCRAA000_D10_premix.fastq 22CCRAA000_F06_premix.fastq 22CCRAA000_H02_premix.fastq
22CCRAA000_A07_premix.fastq 22CCRAA000_C03_premix.fastq 22CCRAA000_D11_premix.fastq 22CCRAA000_F07_premix.fastq 22CCRAA000_H03_premix.fastq
22CCRAA000_A08_premix.fastq 22CCRAA000_C04_premix.fastq 22CCRAA000_D12_premix.fastq 22CCRAA000_F08_premix.fastq 22CCRAA000_H04_premix.fastq
22CCRAA000_A09_premix.fastq 22CCRAA000_C05_premix.fastq 22CCRAA000_E01_premix.fastq 22CCRAA000_F09_premix.fastq 22CCRAA000_H05_premix.fastq
22CCRAA000_A10_premix.fastq 22CCRAA000_C06_premix.fastq 22CCRAA000_E02_premix.fastq 22CCRAA000_F10_premix.fastq 22CCRAA000_H06_premix.fastq
22CCRAA000_A11_premix.fastq 22CCRAA000_C07_premix.fastq 22CCRAA000_E03_premix.fastq 22CCRAA000_F11_premix.fastq 22CCRAA000_H07_premix.fastq
22CCRAA000_A12_premix.fastq 22CCRAA000_C08_premix.fastq 22CCRAA000_E04_premix.fastq 22CCRAA000_F12_premix.fastq 22CCRAA000_H08_premix.fastq
22CCRAA000_B01_premix.fastq 22CCRAA000_C09_premix.fastq 22CCRAA000_E05_premix.fastq 22CCRAA000_G01_premix.fastq 22CCRAA000_H09_premix.fastq
22CCRAA000_B02_premix.fastq 22CCRAA000_C10_premix.fastq 22CCRAA000_E06_premix.fastq 22CCRAA000_G02_premix.fastq 22CCRAA000_H10_premix.fastq
22CCRAA000_B03_premix.fastq 22CCRAA000_C11_premix.fastq 22CCRAA000_E07_premix.fastq 22CCRAA000_G03_premix.fastq 22CCRAA000_H11_premix.fastq
22CCRAA000_B04_premix.fastq 22CCRAA000_C12_premix.fastq 22CCRAA000_E08_premix.fastq 22CCRAA000_G04_premix.fastq 22CCRAA000_H12_premix.fastq
22CCRAA000_B05_premix.fastq 22CCRAA000_D01_premix.fastq 22CCRAA000_E09_premix.fastq 22CCRAA000_G05_premix.fastq
22CCRAA000_B06_premix.fastq 22CCRAA000_D02_premix.fastq 22CCRAA000_E10_premix.fastq 22CCRAA000_G06_premix.fastq
22CCRAA000_B07_premix.fastq 22CCRAA000_D03_premix.fastq 22CCRAA000_E11_premix.fastq 22CCRAA000_G07_premix.fastq
22CCRAA000_B08_premix.fastq 22CCRAA000_D04_premix.fastq 22CCRAA000_E12_premix.fastq 22CCRAA000_G08_premix.fastq
user-cbpg@LAPTOP-FF9BGML5: /mnt/c/Users/Ana_CBGP/Documents/protocols.io/v2/example_BLASTn/example2/results_example2/reads_fastq$
```

Content of the folder *results_example2* and *reads_fastq*

-  all_reads_merged.fasta
-  all_reads_merged_quality.fastq
-  all_reads_merged_quality_fastqc.html
-  all_reads_merged_quality_fastqc.zip
-  all_reads_merged_quality_trimmed....
-  all_reads_merged_trimmed.fasta
-  all_seq_aligned.tsv
-  table_reads_genes_description.csv
-  reads_fastq_compressed.zip

Protocol references

- <https://doi.org/10.1093/synbio/ysad012>
- <https://doi.org/10.1021/acssynbio.3c00397>