IntegronFinder Documentation

Release 2.0.2

B. Néron, E. Littner, M. Haudiquet, A. Perrin, J. Cury, E. P.C. Rocha

May 02, 2022

Contents

IntegronFinder is a program that detects integrons in DNA sequences. The program is available on a webserver *Galaxy Pasteur*, or by command line (IntegronFinder on github).

- You already read the *paper* and want to install it ? Click *here*
- You did not read the paper (yet) but you would like to have rapid introduction to integrons and the program? click *here*

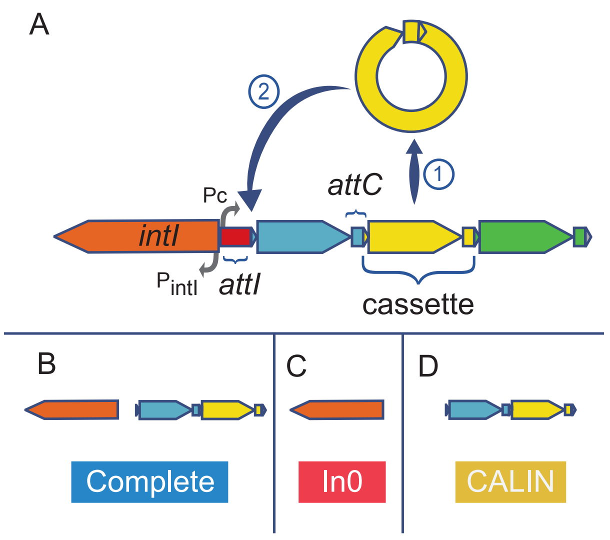
CHAPTER 1

User Guide

1.1 User Guide

1.1.1 Introduction

Integrons are major genetic element, notorious for their major implication in the spread of antibiotic resistance genes. More generally, integrons are gene-capturing platform, whose broader evolutionary role remains poorly understood. IntegronFinder is able to detect with high accuracy integron in DNA sequences. It is accurate because it combines the use of HMM profiles for the detection of the essential protein, the site-specific integron integrase, and the use of Covariance Models for the detection of the recombination site, the *attC* site.



How does it work ?

For each sequence in the input file:

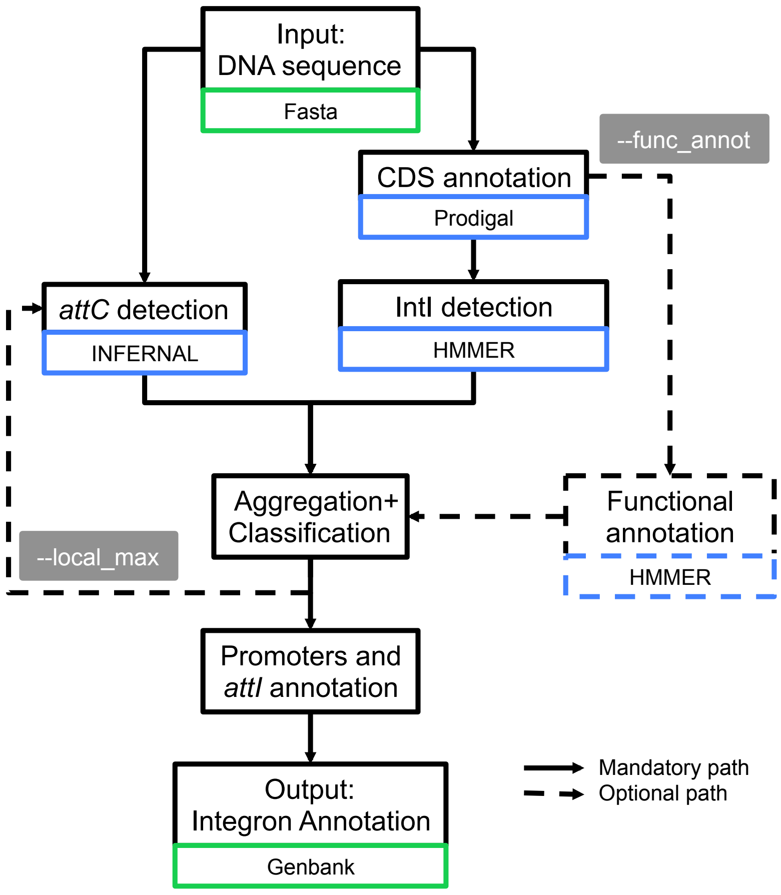
- First, IntegronFinder annotates the DNA sequence's CDS with Prodigal.
- Second, IntegronFinder detects independently integron integrase and *attC* recombination sites. The Integron integrase is detected by using the intersection of two HMM profiles:
  - one specific of tyrosine-recombinase (PF00589)
  - one specific of the integron integrase, near the patch III domain of tyrosine recombinases.

The *attC* recombination site is detected with a covariance model (CM), which models the secondary structure in addition to the few conserved sequence positions.

- Third, the results are integrated, and IntegronFinder distinguishes 3 types of elements:
  - **complete integron (panel B above)** Integron with integron integrase nearby *attC* site(s)
  - **In0 element (panel C above)** Integron integrase only, without any *attC* site nearby
  - **CALIN element (panel D above)** Cluster of *attC* sites Lacking INtegrase nearby. A rule of thumb to avoid false positive is to filter out singleton of *attC* site.

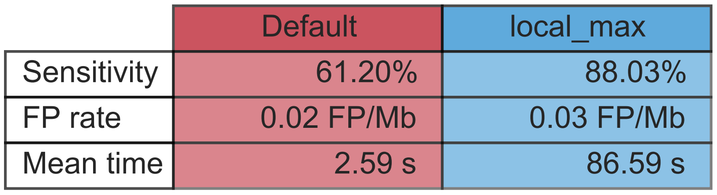
IntegronFinder can also annotate gene cassettes (CDS nearby *attC* sites) using AMRFinderPlus, a database of HMM profiles aiming at annotating antibiotic resistance genes. This database is provided but the user can add any other HMM profiles database of its own interest.

When available, IntegronFinder annotates the promoters and attI sites by pattern matching.

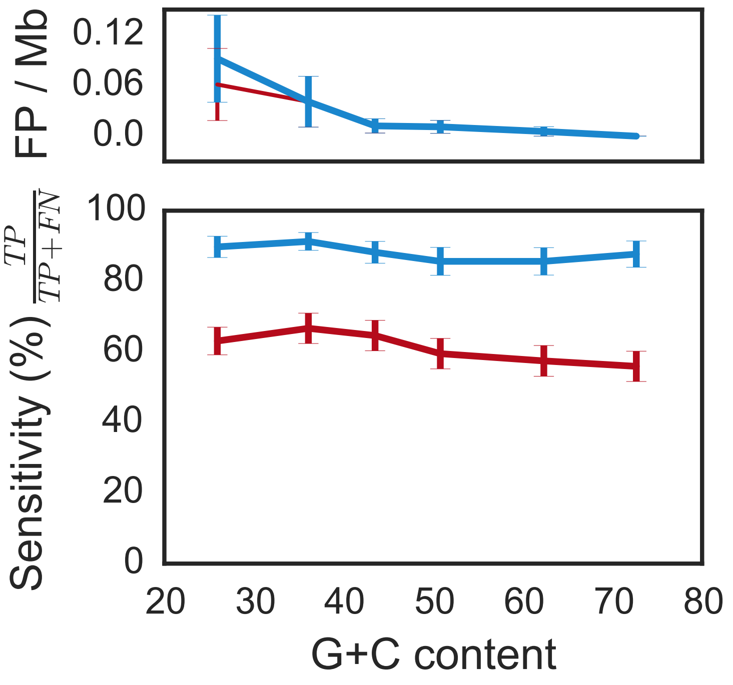


Does it work ?

Yes! The estimated sensitivity is 61% on average with the default option and goes up to 88% with the `--local_max` option. The missing *attC* sites are usually at the end of the array. The False positive rate with the `--local_max` option is estimated between 0.03 False Positive per Megabases (FP/Mb) to 0.72 FP/Mb. This leads to a probability of finding 2 consecutive false *attC* sites within 4kb between 4.10^-6 and 7.10^-9. Overall, the probability of finding an integron in a chromosome (including finding a part of it) is more than 95%. Finally, these parameters do not depend on the G+C percent of the given replicon. See the paper for more information (freely accessible).



	Default	local_max
Sensitivity	61.20%	88.03%
FP rate	0.02 FP/Mb	0.03 FP/Mb
Mean time	2.59 s	86.59 s



The time in the table correspond to the average time per run with a pseudogenome having attC sites on a Mac Pro, 2 x 2.4 GHz 6-Core Intel Xeon, 16 Gb RAM, with options –cpu 20 and –no-proteins.

Note: The time does not vary depending of the mode (default or local_max), and is about a couple of second, if the replicon does not contain any *attC* site.

1.1.2 Installation

IntegronFinder dependencies

IntegronFinder is built with Python >= 3.4, and a few libraries are needed:

- Python >=3.7,<=3.10
- Pandas >=1.1.5,<=1.4.0
- Numpy >=1.19.4,<=1.22.1
- Biopython >=1.78,<=1.79

- Matplotlib >=3.3.3,<=3.5.1
- colorlog

From version 1.5.1, integron_finder will check and install theses libraries for you.

In addition, IntegronFinder has external dependencies, which have to be installed prior the use of the program (click to access the corresponding website).

- HMMER >=3.1b2,<=3.3.2
- INFERNAL >=1.1.2,<=1.1.4
- Prodigal >V2.6.2,<=V2.6.3
- Nextflow (for parallelization)

After installation of these programs, they should be in your `$PATH` (*i.e.* you can type in a terminal `hmmsearch`, `cmsearch`, or `prodigal` and a `command not found` shall not be displayed). If you have them installed somewhere else, please refer to integron_finder's parameters to give complete path to IntegronFinder.

Installation procedure

Warning: When installing a new version (up to 2.0 included) of IntegronFinder, do not forget to *uninstall* the previous version installed !

Warning: If You upgrading from version prior to 2.0 to 2.0 be careful the python used changed for 3.x. The python 2.7 is not supported anymore. So if you installed `integron_finder` within a virtualenv you need to create a new one based on python3.

From Version 2.0

System wide installation

1. Open a terminal and hit (not recommended):

```
sudo pip install integron_finder
```

Warning: On recent Debian/Ubuntu the –user option is forced. So use of –root option give an unexpected behavior and you cannot use –prefix option at all unless you add option –system for instance

```
sudo pip install --system integron_finder
```

or

```
pip install --prefix=/tmp/test_if --system integron_finder
```

2. To get an updated version (no need to uninstall):

   ```
   sudo pip install -U integron_finder
   ```

User wide installation

1. Open a terminal and hit:

   ```
   pip install --user integron_finder
   ```

Installation in a virtualenv

The virtual environment (virtualenv) is a system to isolate a python program from the system and avoid libraries conflict. So you can install a different python or libraries version than your system in each virtualenv. So if you update the system it will not change anything for your program and *vice versa*. If you want to remove the program just remove the virtual environment.

Create a virtual environment:

```
python3 -m venv Integron_Finder
```

or on some systems:

```
virtualenv -p python3 Integron_Finder
```

activate you virtualenv:

```
source Integron_Finder/bin/activate
```

The name of the virtualenv appear in parenthesis at the beginning of the prompt. Then install integron_finder:

```
pip install integron_finder
```

To run integron finder, you have to activate (once per session) the virtual environment:

```
source  Integron_Finder/bin/activate
```

When you do not need to use integron_finder just deactivate the virtual environment. In the active terminal just type:

```
deactivate
```

The integron_finder command will disappear from the path. The name of the virtualenv disappear from the prompt.

Conda Package

From 2.0 version, Integron_Finder is available as conda package. Integron_finder is in bioconda From 2.0 version, Integron_Finder is available as [conda](https://conda.io/docs/index.html) package. Integron_finder is in [bioconda](https://bioconda.github.io/) channel. (The advantage with this solution is that it will install prodigal, hmmer, and infernal too.)

1. install conda
2. Set up channels

   ```
   conda config --add channels defaults
   conda config --add channels conda-forge
   conda config --add channels bioconda
   ```

3. install integron_finder

   ```
   conda install integron_finder
   ```

   (The advantage with this solution is that it will install prodigal, hmmer, and infernal too.)

From Version 1.5.1 and after

1. Open a terminal and hit:

   ```
   (sudo) pip install integron_finder
   ```

2. To get an updated version (no need to uninstall):

   ```
   (sudo) pip install -U integron_finder
   ```

For Version 1.5 and before

1. Download the latest release that can be installed like this (v1.5)
2. Uncompress it

3. In a shell (*e.g.* a terminal), go to the directory and run:

   ```
   (sudo) python setup.py install
   ```

Note: Super-user privileges (*i.e.*, `sudo`) are necessary if you want to install the program in the general file architecture.

Note: If you do not have the privileges, or if you do not want to install IntegronFinder in the Python libraries of your system, you can install IntegronFinder in a virtual environment. See virtualenv or if you're using Canopy, see Canopy CLI

Warning: The installer does not work with pure setuptools procedure, it does not work in egg. Unless you disable egg by using the `--root` option. `python setup.py install --root /prefix/where/to/install/integron_finder`

Uninstallation procedure

From Version 1.5.1 and after

To uninstall IntegronFinder, run in the following command:

```
(sudo) pip uninstall integron_finder
```

It will uninstall integron_finder executable

From Version 1.0 to Version 1.5

Go to the directory from where you installed IntegronFinder (e.g. Integron_Finder-1.5), and run:

```
(sudo) python setup.py uninstall
```

How to install Python

The purpose of this section is to provide some help about installing python dependencies for IntegronFinder if you never installed any python package.

As IntegronFinder has not been tested on Windows, we assume Unix-based operating system. For Windows users, the best would be to install a unix virtual machine on your computer.

Usually a python distribution is already installed on your machine. However, if you don't know how to install libraries, we recommend to re-install it from a distribution which contains pre-compiled libraries. There are two main distributions (click to access website):

- Enthought Canopy
- Anaconda

Download version 3.x which correspond to your machine, then make sure that python from these distributions is the default one (you can possibly choose that in the preference and/or during installation). Make sure Biopython is installed, otherwise, you will have to install Biopython. `pip` or `conda` are recommended as a python packages installer.

It works as follow:

```
(sudo) pip install Biopython==1.71
```

To install version 1.71 of Biopython (recommended for IntegronFinder).

Note: If you don't manage to install all the packages, try googling the error, or don't hesitate to ask a question on stackoverflow.

1.1.3 What's new ?

In Version 2.0

Here are the major changes between versions 1.x and 2.0. Essentially, it has be designed such as it becomes easier to find integrons with high confidence in huge datasets (but it works also for small datasets).

- IntegronFinder now accepts multifasta files as input.
- Only three files are created by default (see output section for details about the other possible output files):
  - A file with all integrons and their elements detected in all sequences in the input file.
  - A summary file with the number and type of integrons per sequence.
  - A file with standard output
- IntegronFinder can be run in parallel easily with a provided Nextflow script that is (almost) ready to use.
- We diversify the installation methods, so it can be easily deployed on a variety of machine. Notably, we built a docker container which will allow a smooth installation on clusters (via docker or singularity).

- CALINs are now reported when they have at least 2 *attC* sites (instead of 1 before). This value can be changed by the user with *–calin-threshold x*
- Promoters and attI sites are not detected by default to increase speed
- It is now easy to obtain multiple alignments of detected attC sites
- Improve the documentation, especially on the developer part so anyone can contribute.
- Add unit (or non regression) tests.

1.1.4 Quick start

We assume here that the program is *installed*.

You can see all available options with:

```
integron_finder -h
```

For impatient

Go to the directory containing your input file(s), or specify the path to that file and call:

```
integron_finder mysequences.fst
```

or:

```
integron_finder path/to/mysequences.fst
```

It will perform a search, and outputs the results in a directory called `Results_Integron_Finder_mysequences`.

Input and Outputs

Inputs

`integron_finder` can take as an input:

- a fasta file
- a multi-fasta file
- many (multi-)fasta files
- several replicon in gembase format

Gembase format

Integron_Finder can use *gembase* formatted protein files instead of re-annotating genes with *Prodigal*. This feature enables the user to use *Integron_Finder* with its own annotations. The *gembase* format is the typical result of *PanACoTA's* annotation step It must contain, at least, a *LSTINF* and a Protein file per replicon. Folder structure must have the following architecture.

```
gembase/
├── Genes
│   ├── ACBA.0917.00019.gen       # genome in fasta format one seq/gene
│   └── ESCO001.C.00001.C001.gen
├── LSTINFO
│   ├── ACBA.0917.00019.lst       # information in space separated␣
↪values
│   └── ESCO001.C.00001.C001.lst
├── Proteins
│   ├── ACBA.0917.00019.prt       # proteins in fasta format
│   └── ESCO001.C.00001.C001.prt
└── Replicons
    ├── ACBA.0917.00019.fna             # genome in fasta format one␣
↪seq/replicon
    └── ESCO001.C.00001.C001.fst
```

5 first sequences headers in gembase/Genes/ACBA.0917.00019.gen file

```
>ACBA.0917.00019.b0001_00001 1215 tyrS | Tyrosine--tRNA ligase | 6.1.1.
↪1 | similar to AA sequence:UniProtKB:P41256
>ACBA.0917.00019.i0001_00002 1128 anmK | Anhydro-N-acetylmuramic acid␣
↪kinase | 2.7.1.170 | similar to AA sequence:UniProtKB:Q8EHB5
>ACBA.0917.00019.i0001_00003 846 ephA | Epoxide hydrolase A | 3.3.2.10␣
↪| similar to AA sequence:UniProtKB:I6YGS0
>ACBA.0917.00019.i0001_00004 336 erpA | Iron-sulfur cluster insertion␣
↪protein ErpA | NA | similar to AA sequence:UniProtKB:P45344
>ACBA.0917.00019.i0001_00005 1005 NA | hypothetical protein | NA | NA
...
```

5 first sequences headers in gembase/Genes/ACBA.0917.00019.gen file

```
>ACBA.0917.00019.b0001_00001 1215 tyrS | Tyrosine--tRNA ligase | 6.1.1.
↪1 | similar to AA sequence:UniProtKB:P41256
>ACBA.0917.00019.i0001_00002 1128 anmK | Anhydro-N-acetylmuramic acid␣
↪kinase | 2.7.1.170 | similar to AA sequence:UniProtKB:Q8EHB5
>ACBA.0917.00019.i0001_00003 846 ephA | Epoxide hydrolase A | 3.3.2.10␣
↪| similar to AA sequence:UniProtKB:I6YGS0
>ACBA.0917.00019.i0001_00004 336 erpA | Iron-sulfur cluster insertion␣
↪protein ErpA | NA | similar to AA sequence:UniProtKB:P45344
```

(continues on next page)

(continued from previous page)

```
>ACBA.0917.00019.i0001_00005 1005 NA | hypothetical protein | NA | NA
...
```

5 first lines in gembase/LSTINFO/ACBA.0917.00019.lst

```
266     1480    C       CDS     ACBA.0917.00019.b0001_00001     tyrS   ␣
↪ | Tyrosine--tRNA ligase | 6.1.1.1 | similar to AA␣
↪sequence:UniProtKB:P41256
1560    2687    D       CDS     ACBA.0917.00019.i0001_00002     anmK   ␣
↪ | Anhydro-N-acetylmuramic acid kinase | 2.7.1.170 | similar to AA␣
↪sequence:UniProtKB:Q8EHB5
2815    3660    D       CDS     ACBA.0917.00019.i0001_00003     ephA   ␣
↪ | Epoxide hydrolase A | 3.3.2.10 | similar to AA␣
↪sequence:UniProtKB:I6YGS0
3716    4051    C       CDS     ACBA.0917.00019.i0001_00004     erpA   ␣
↪ | Iron-sulfur cluster insertion protein ErpA | NA | similar to AA␣
↪sequence:UniProtKB:P45344
4176    5180    C       CDS     ACBA.0917.00019.i0001_00005     NA     ␣
↪ | hypothetical protein | NA | NA
...
```

all sequences headers in gembase/Replicons/ACBA.0917.00019.fna

```
>ACBA.0917.00019.0001
>ACBA.0917.00019.0002
```

Integron_finder will use *LSTINF* and *Proteins* folders. Hence if you want to analyze a replicon located in a *gembase*, the command line should look like

> integron_finder –gembase data/Replicons/ACBA.0917.00019.fna

Note: If the replicon you want to analyze is not in the gembase directory, but you still want to take advantage of the *gembase* annotation, then you have to specify the *–gembase-path* option to indicate where to find it. A typical command line could be:

> integron_finder –gembase –gembase-path data/gembase/ My_replicons/ACBA.0917.00019.fna

Custom protein file

Integron_Finder allow the user to provide it's own protein file with it's own annotations instead of using *prodigal* (default). In this case you have to specified the protein file to use with the option *–prot-file* and the path to the parser to extract information from this file *–annot-parser*. A typical command line could be:

> integron_finder –prot-file <path/to/custom/protein /file> –annot-parser <my_annot_paser.py> <replicon_path>

The annotation parser must be a python file (with the *.py* extension') with one function called *description_parser* This function must take **one** argument which is the header of a fasta sequence header (without the first character >) and must return a **tuple with 4 elements**:

- sequence id: the id of the sequence (string)
- start: the beginning position of the protein on the genome (positive int)
- stop: the end position of the protein on the genome (positive int)
- strand: the strand 1 if the protein os coded on the direct strand or -1 on the reverse

Below a description_parser to parse annotation in prodigal format:

```
from typing import Tuple


def description_parser(seq_header: str) -> Tuple[int, int,␣
↪int, int]:
    """
    Extract description features from protein sequence fasta␣
↪header

    :param str seq_header: the fasta header of a sequence␣
↪(without '>' char)
    :return: features
                 - sequence id (string)
                 - start the beginning of the protein␣
↪(positive integer)
                 - stop the end position of the protein␣
↪(positive integer)
                 - strand 1 if prot is coded on direct strand␣
↪or -1 if it's on reverse
    :rtypes: tuple (str, int, int, 1/-1)
    """
    id_, start, stop, strand, *_ = seq_header.split(" # ")
    start = int(start)
    stop = int(stop)
    strand = int(strand)
    return id_, start, stop, strand
```

Outputs

By default, `integron_finder` will output 3 files under `Results_Integron_Finder_mysequences`:

- `mysequences.integrons` : A file with all integrons and their elements detected in all sequences in the input file.
- `mysequences.summary` : A summary file with the number and type of integrons per sequence.
- `integron_finder.out` : A copy standard output. The stdout can be silenced with the argument `--mute`

The amount of log in the standard output can be controlled with `--verbose` for more or `--quiet` for less, and both are cumulative arguments, eg. `-vv` or `-qq`.

Other files can be created on demand:

- `--gbk`: Creates a Genbank files with all the annotations found (present in the `.integrons` file)
- `--pdf`: Creates a simple pdf graphic with complete integrons
- `--split-results`: Creates a `.integrons` a `.summary` file per replicon if the input is a multifasta file.
- `--keep-tmp`: Keep temporary files. See *Keep intermediate files* for more.

For everyone

Note: The different options will be shown separately, but they can be used altogether unless otherwise stated.

Thorough local detection

This option allows a much more sensitive search of *attC* sites. It will be slower if integrons are found, but will be as fast if nothing is detected.

```
integron_finder mysequences.fst --local-max
```

CALIN detection

By default CALIN are reported if they are composed of at least 2 *attC* sites, in order to avoid false positives. This value was chosen as CALIN with 2 attC sites were unlikely to be false positive. The probability of a false CALIN with at least 2 attC sites within 4kb was estimated between 4.10^-6 and 7.10^-9. However, one can modify this value with the option *–calin-threshold* and use a lower or higher value depending on the risk one is willing to take:

```
integron_finder mysequences.fst --calin-threshold 1
```

Note: If `--local-max` is called, it will run around CALINs with single attC sites, even if `--calin-threshold` is 2. The filtering step is done after the search with local max in that case.

Functional annotation

This option allows to annotate cassettes given HMM profiles. As AMRFinderPlus database is distributed, to annotate antibiotic resistance genes, just use:

```
integron_finder mysequences.fst --func-annot
```

IntegronFinder will look in the directory `Integron_Finder-x.x/data/Functional_annotation` and use all `.hmm` files available to annotate. By default, there is only `NCBIfam-AMRFinder.hmm`, but one can add any other HMM file here. Alternatively, if one wants to use a database which is present elsewhere on the user's computer without copying it into that directory, one can specify the following option

```
integron_finder mysequences.fst --path_func_annot bank_hmm
```

where `bank_hmm` is a file containing one absolute path to a hmm file per line, and you can comment out a line

```
~/Downloads/Integron_Finder-x.x/data/Functional_annotation/NCBIfam-
↪AMRFinder.hmm
~/Documents/Data/Pfam-A.hmm
# ~/Documents/Data/Pfam-B.hmm
```

Here, annotation will be made using Pfam-A et NCBIfam-AMRFinder, but not Pfam-B. If a protein is hit by 2 different profiles, the one with the best e-value will be kept.

Search for promoter and *attI* sites

By default `integron_finder` look for *attC* sites and site-specific integron integrase,, If you want to search for known promoters (integrase, Pc-int1 and Pc-int3) and AttI sites in integrons elements you need to add the `--promoter-attI` option on the command line.

Keep intermediate results

Integrons finder needs some intermediate results to run completely. It includes notably the protein file in fasta (mysequences.prt), but also the outputs from hmmer and infernal. A folder containing these outputs is generated for each replicon and have name `tmp_<replicon_id>` This directory is removed at the end. You can keep this directory to analyse further each `integron_finder` steps with the option `--keep-tmp`. Using this argument allows you to rerun `integron_finder` on the same sequences without redetecting proteins and attC sites. It is useful if one wants to change clustering parameters, evalues of attC sites, or size of them. Note that it won't search for new attC sites so it is better to start with relaxed parameters and then rerun `integron_finder` with more strict parameters. See the section *for integron diggers* for more informations

For each tmp file, there are:

- `<replicon_id>.fst`: a single fasta file with the replicon_name
- `<replicon_id>.prt`: a multifasta file with the sequences of the detected proteins.
- `<replicon_id>_intI_table.res`: hmm result for the intI hmm profile in tabular format
- `<replicon_id>_intI.res`: hmm result for the intI hmm profile
- `<replicon_id>_phage_int_table.res`: hmm result for the tyrosine recombinase hmm profile in tabular format
- `<replicon_id>_phage_int.res`: hmm result for the tyrosine recombinase hmm profile in tabular format
- `<replicon_id>_attc_table.res`: cmsearch result for the attC sites covariance model in tabular format
- `<replicon_id>_attc.res`: significant (according to `evalue-attc`) attC sites aligned in stockholm format
- `integron_max.pickle`: pickle file so `integron_finder` reuse this instead of re-running the local_max part

Topology

By default, IntegronFinder assumes that

- your replicon is considered as **circular** if there is **only one replicon** in the input file.
- your replicons are considered as **linear** if there are **several replicons** in the input file.

However, you can change this default behavior and specify the default topology with options `--circ` or `--lin`:

```
integron_finder --lin mylinearsequence.fst
integron_finder --circ mycircularsequence.fst
```

If you have multiple replicon in the input file with different topologies you can specify a topology for each replicon by providing a topology file. The syntax for the topology file is simple:

- one topology by line
- one line start by the seqid followed by 'circ' or 'lin' for circular or linear topologies.

example:

```
seq_id_1 circ
seq_id_2 lin
```

You can also mix the options `--circ` or `--lin` with option `--topology-file`:

```
integron_finder --circ --topology-file path/to/topofile mysequencess.
↪fst
```

In the example above the default topology is set to *circular*. The replicons specified in topofile supersede the default topology.

Warning: However, if the replicon is smaller than `4 x dt` (where `dt` is the distance threshold, so 4kb by default), the replicon is considered linear to avoid clustering problem. The topology used to searching integron is report in the **.integrons file*

For big data people

Parallelization

The time limiting part are HMMER (search integrase) and INFERNAL (search *attC* sites). So if you have to analyze one or few replicons the user can set the number of CPU used by HMMER and INFERNAL:

```
integron_finder mysequences.fst --cpu 4
```

Default is 1.

If you want to deal with a fasta file with a lot of replicons (from 10 to more than thousand) we provide a workflow to parallelize the execution of the data. This mean that we cut the data input into chunks (by default of one replicon) then execute IntegronFinder in parallel on each replicon (the number of parallel tasks can be limited) then aggregate the results in one global summary. The workflow use the nextflow framework and can be run on a single machine or a cluster.

First, you have to install nextflow first, and *integron_finder*. Then we provide 2 files (you need to download them from the IntegronFinder github repo.)

- *parallel_integron_finder.nf* which is the workflow itself in nextflow syntax
- *nextflow.config* which is a configuration file to execute the workflow.

The workflow file should not be modified. Whereas the profile must be adapted to the local architecture.

The file *nextflow.config* provide for profiles:

- a standard profile for local use
- a cluster profile
- a standard profile using apptainer container
- a cluster profile using apptainer container

so now install nextflow. If you have capsule error like

```
CAPSULE EXCEPTION: Error resolving dependencies. while processing␣
↪attribute Allow-Snapshots: false (for stack trace, run with -
↪Dcapsule.log=verbose)
Unable to initialize nextflow environment
```

install nextflow (>=0.29.0) as follow (change the nextflow version with the last release)

```
wget -O nextflow http://www.nextflow.io/releases/v0.30.2/nextflow-
↪0.30.2-all
chmod 777 nextflow
```

for more details see: https://github.com/nextflow-io/nextflow/issues/770#issuecomment-400384617

How to get parallel_integron_finder

The release contains the workflow *parallel_integron_finder.nf* and the *nextflow.config* at the top level of the archive But If you use pip to install Integron_Finder you have not easily access to them. But they can be downloaded or executed directly by using nextflow.

to download it

```
nextflow pull gem-pasteur/Integron_Finder
```

to get the latest version or use *-r* option to specify a version

```
nextflow pull -r release_2.0 gem-pasteur/Integron_Finder
```

to see what you download

```
nextflow see Integron_Finder
```

to execute it directly

```
nextflow run gem-pasteur/Integron_Finder -profile standard --replicons␣
↪all_coli.fst --circ
```

or:

```
nextflow run -r release_2.0 gem-pasteur/Integron_Finder -profile␣
↪standard --replicons all_coli.fst --circ
```

standard profile

This profile is used if you want to parallelize IntegronFinder on your machine. You can specify the number of tasks in parallel by setting the queueSize value

```
standard {
        executor {
            name = 'local'
            queueSize = 7
        }
        process{
            executor = 'local'
            $integron_finder{
                errorStrategy = 'ignore'
                cpu=params.cpu
            }
        }
 }
```

If you installed IntegronFinder with apptainer, just uncomment the container line in the script, and set the proper path to the container.

All options available in non parallel version are also available for the parallel one. except the `--outdir` which is not available and `--replicons` option which is specific to the parallelized version. `--replicons` allows to specify the path of a file containing the replicons.

A typical command line will be:

```
./parallel_integron_finder.nf -profile standard --replicons all_coli.
↪fst --circ
```

Note: Joker as `*` or `?` can be used in path to specify several files as input.

But **do not forget** to protect the wild card from the shell for instance by enclosing your glob pattern with simple quote.

```
nextflow run -profile standard parallel_integron_finder.nf --replicons
↪'replicons_dir/*.fst'
```

Two asterisks, i.e. `**`, works like `*` but crosses directory boundaries. Curly brackets specify a collection of sub-patterns.

```
nextflow run -profile standard parallel_integron_finder.nf --replicons
↪'data/**.fa'
nextflow run -profile standard parallel_integron_finder.nf --replicons
↪'data/**/*.fa'
nextflow run -profile standard parallel_integron_finder.nf --replicons
↪'data/file_{1,2}.fa'
```

The first line will match files ending with the suffix *.fa* in the *data* folder and recursively in all its sub-folders. While the second one only match the files which have the same suffix in any sub-folder in the data path. Finally the last example capture two files: *data/file_1.fa*, *data/file_2.fa*

More than one path or glob pattern can be specified in one time using comma. **Do not** insert spaces surrounding the comma

```
nextflow run -profile standard parallel_integron_finder --replicons
↪'some/path/*.fa,other/path/*.fst'
```

The command above will analyze all files ending by *.fa* in */some/path* with *.fst* extension in *other/path*

For further details see: https://www.nextflow.io/docs/latest/channel.html#frompath

Note: The option *–outdir* is not allowed. Because you can specify several replicon files as input, So in this circumstances specify only one name for the output is a none sense.

Note: The options starting with one dash are for nextflow workflow engine, whereas the options starting by two dashes are for integron_finder workflow.

Note: Replicons will be considered linear by default (see above), here we use *–circ* to consider replicons circular.

Note: If you specify several input files, the split and merge steps will be parallelized.

If you execute this line, 2 kinds of directories will be created.

- One named *work* containing lot of subdirectories this for all jobs launch by nextflow.
- Directories named *Results_Integron_Finder_XXX* where XXX is the name of the replicon file. So, one directory per replicon file will be created. These directories contain the final results as in non parallel version.

cluster profile

The cluster profile is intended to work on a cluster managed by SLURM. If your cluster is managed by an other drm replace executor name by the right value (see nextflow supported cluster)

You can also manage

- The number of tasks in parallel with the *executor.queueSize* parameter (here 500). If you remove this line, the system will send in parallel as many jobs as there are replicons in your data set.
- The queue (or partition in SLURM terminology) with *process.queue* parameter (here common,dedicated)
- and some options specific to your cluster management systems with *process.clusterOptions* parameter

```
cluster {
    executor {
        name = 'slurm'
        queueSize = 500
    }

    process{
        executor = 'slurm'
        queue= 'common,dedicated'
```

(continues on next page)

(continued from previous page)

```
            clusterOptions = '--qos=fast'
            $integron_finder{
                cpu=params.cpu
            }
        }
}
```

To run the parallel version on a cluster, for instance on a cluster managed by slurm, I can launch the main nextflow process in one slot. The parallelization and the submission on the other slots is made by nextflow itself. Below a command line to run parallel_integron_finder and use 2 cpus per integron_finder task, each integron_finder task can be executed on a different machine, each integron_finder task claim 2 cores (cpus in nextflow terminology) to speed up the attC sites or integrase search:

```
sbatch --qos fast -p common nextflow run  parallel_integron_finder.nf -
↪profile cluster --replicons all_coli.fst --cpu 2 --local-max --gbk --
↪circ
```

The results will be the same as described in local execution.

apptainer (formely singularity) profiles

If you use the integron_finder image with the apptainer container executor, use the profile *standard_apptainer*. With the command line below nextflow will download parallel_integron_finder from github and download the integron_finder image from the docker hub and convert it to apptainer on the fly so you haven't to install anything except nextflow and apptainer.

```
nextflow run gem-pasteur/Integron_Finder -profile standard_apptainer --
↪replicons all_coli.fst --circ
```

You can also use the integron_finder apptainer image on a cluster, for this use the profile *cluster_apptainer*.

```
sbatch --qos fast -p common nextflow run  gem-pasteur/Integron_
↪Finder:2.0 -profile cluster_apptainer --replicons all_coli.fst --cpu␣
↪2 --local-max --gbk --circ
```

In the case of your cluster cannot reach the world wide web. you have to download the apptainer image

```
apptainer pull --name Integron_Finder docker pull gempasteur/integron_
↪finder:<tag>
```

the move the image on your cluster modify the nextflow.config to point on the location of the image, and adapt the cluster options (executor, queue, . . .) to your architecture

```
 cluster_apptainer {
        executor {
            name = 'slurm'
            queueSize = 500
        }

        process {
            container = /path/to/integron_finder/image
            queue = 'common,dedicated'
            clusterOptions = '--qos=fast'
            withName: integron_finder {
                cpus = params.cpu
            }
        }
        singularity {
            enabled = true
            runOptions = '-B /pasteur'
            autoMounts = false
       }
    }
}
```

then run it

```
sbatch --qos fast -p common nextflow run  ./parallel_integron_finder.
↪nf -profile cluster_singualrity --replicons all_coli.fst --cpu 2 --
↪local-max --gbk --circ
```

If you want to have more details about the jobs execution you can add some options to generate report:

Execution report

To enable the creation of this report add the `-with-report` command line option when launching the pipeline execution. For example:

```
nextflow run  ./parallel_integron_finder.nf -profile standard -with-
↪report [file name] --replicons
```

It creates an HTML execution report: a single document which includes many useful metrics about a workflow execution. For further details see https://www.nextflow.io/docs/latest/tracing.html#execution-report

Trace report

In order to create the execution trace file add the `-with-trace` command line option when launching the pipeline execution. For example:

```
nextflow run  ./parallel_integron_finder.nf -profile standard -with-
↪trace --replicons
```

It creates an HTML timeline for all processes executed in your pipeline. For further details see https://www.nextflow.io/docs/latest/tracing.html#timeline-report

Timeline report

To enable the creation of the timeline report add the `-with-timeline` command line option when launching the pipeline execution. For example:

```
nextflow run  ./parallel_integron_finder.nf -profile standard -with-
↪timeline [file name] --replicons ...
```

It creates an execution tracing file that contains some useful information about each process executed in your pipeline script, including: submission time, start time, completion time, cpu and memory used. For further details see https://www.nextflow.io/docs/latest/tracing.html#trace-report

For integron diggers

Many options are set to prevent false positives. However, one may want higher sensitivity at the expense of having potentially false positives. Ultimately, only experimental experiments will tell whether a given *attC* sites or integrase is functional.

Also, note that because of how local_max works (ie. around already detected elements), true *attC* sites may be found thanks to false *attC* sites, because false *attC* sites may trigger local_max around them. Hence, one may want to use very relaxed parameters first with the `--keep-tmp` flag to rerun the analysis on the same data while restrincting the parameters.

Clustering of elements

attC sites are clustered together if they are on the same strand and if they are less than 4 kb apart (`-dt 4000` by default). To cluster an array of *attC* sites and an integron integrase, they also must be less than 4 kb apart. This value has been empirically estimated and is consistent with previous observations showing that biggest gene cassettes are about 2 kb long. This value of 4 kb can be modified though:

```
integron_finder mysequences.fst --distance-thresh 10000
```

or, equivalently:

```
integron_finder mysequences.fst -dt 10000
```

This sets the threshold for clustering to 10 kb.

Note: The option `--outdir` allows you to chose the location of the Results folder (`Results_Integron_Finder_mysequences`). If this folder already exists, IntegronFinder will not re-run analyses already done, except functional annotation. It allows you to re-run rapidly IntegronFinder with a different `--distance-thresh` value. Functional annotation needs to re-run each time because depending on the aggregation parameters, the proteins associated with an integron might change.

Integrase

We use two HMM profiles for the detection of the integron integrase. One for tyrosine recombinase and one for a specific part of the integron integrase. To be specific we use the intersection of both hits, but one might want to use the union of both hits (and sees whether it exists cluster of attC sites nearby non integron-integrase. . .). To do so, use:

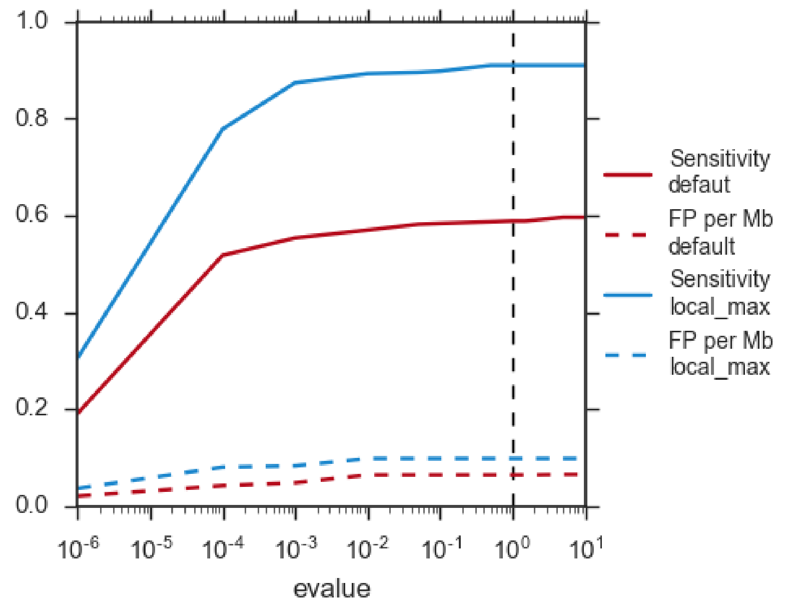
```
integron_finder mysequences.fst --union-integrases
```

attC evalue

The default evalue is 1. Sometimes, degenerated *attC* sites can have a evalue above 1 and one may want to increase this value to have a better sensitivity.

```
integron_finder mysequences.fst --evalue-attc 5
```

Here is a plot of how the sensitivity and false positive rate evolve as a function of the evalue:



Note: If one wants to have maximum sensitivity, use a high evalue (max is 10), and then integron_finder can be run again on the same data with a lower evalue. It won't work the other way around (starting with low evalue), as attC sites are not searched again.

attC size

By default, *attC* sites' size ranges from 40 to 200bp. This can be changed with the `--min-attc-size` or `--max-attc-size` parameters:

```
integron_finder mysequences.fst --min-attc-size 50 --max-attc-size 100
```

Palindromes

attC sites are more or less palindromic sequences, and sometimes, a single *attC* site can be detected on the 2 strands. By default, the one with the highest evalue is discarded, but you can choose to keep them with the following option:

```
integron_finder mysequences.fst --keep-palindromes
```

attC alignements

One can get the alignements of *attC* sites in the temporary files (use `--keep-tmp`) to have them. Under `Results_Integron_Finder_mysequences/tmp_repliconA/`

`repliconA_attc.res` one can find alignements of *attC* sites from repliconA, in Stokholm format, where R and L core regions are aligned with each others:

```
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2

ACBA.0917.00019.0001/315102-315161         GUCUAACAAUUC---
↪GUUCAAGCcgacgccgcu.................................................
↪ucgcggcgcgGCUUAACUCAAGC----GUUAGAU
#=GR ACBA.0917.00019.0001/315102-315161 PP ************...
↪******************.................................................
↪***********************....*******
ACBA.0917.00019.0001/313260-313368         ACCUAACAAUUC---
↪GUUCAAGCcgagaucgcuucgcggccgcggaguuguucggaaaaauugucacaacgccgcggccgcaaagcgcuccgGCUU
↪---GUUGGGC
#=GR ACBA.0917.00019.0001/313260-313368 PP ************...
↪*****************************************************************************
↪...*******
ACBA.0917.00019.0001/313837-313906         GCCCAACAUGGC---
↪GCUCAAGCcgaccggccagcccu.......................................
↪gcgggcuguccgucgGCUUAGCUAGGGC----GUUAGAG
#=GR ACBA.0917.00019.0001/313837-313906 PP ************...
↪***********************.......................................
↪****************************....*******
#=GC SS_cons                               <<<<<<<--------<<<-<<<<.....
↪................................................................>>>>>
↪>>---------->>>>>>>
#=GC RF                                    [Rsec=]========[=Lsec=].....
↪.................................................................
↪[Lprim]==========[Rprim]
//
```

Which you can manipulate easily with `esl-alimanip` tools provided by infernal (the following examples should work if your `cmsearch` is in your `PATH`). You can convert the same alignement in dna alphabet (cmsearch use RNA alphabet):

```
$ esl-alimanip --dna Results_Integron_Finder_mysequences/tmp_ACBA.0917.
↪00019.0001/ACBA.0917.00019.0001_attc.res
# STOCKHOLM 1.0
#=GF AU Infernal 1.1.2

ACBA.0917.00019.0001/315102-315161         GTCTAACAATTC---
↪GTTCAAGCCGACGCCGCT-------------------------------------------------
↪TCGCGGCGCGGCTTAACTCAAGC----GTTAGAT
#=GR ACBA.0917.00019.0001/315102-315161 PP ************...
↪******************.................................................
↪***********************....*******
```

(continues on next page)

(continued from previous page)

```
ACBA.0917.00019.0001/313260-313368         ACCTAACAATTC---
↪GTTCAAGCCGAGATCGCTTCGCGGCCGCGGAGTTGTTCGGAAAAATTGTCACAACGCCGCGGCCGCAAAGCGCTCCGGCTT
↪---GTTGGGC
#=GR ACBA.0917.00019.0001/313260-313368 PP ************...
↪********************************************************************************
↪...*******
ACBA.0917.00019.0001/313837-313906         GCCCAACATGGC---
↪GCTCAAGCCGACCGGCCAGCCCT---------------------------------------
↪GCGGGCTGTCCGTCGGCTTAGCTAGGGC----GTTAGAG
#=GR ACBA.0917.00019.0001/313837-313906 PP ************...
↪***********************.......................................
↪****************************....*******
#=GC SS_cons                               <<<<<<<--------<<<-<<<<.....
↪................................................................>>>>>
↪>>---------->>>>>>>
#=GC RF                                    [Rsec=]========[=Lsec=].....
↪................................................................
↪[Lprim]==========[Rprim]
//
```

You can also convert it to fasta format:

```
$ esl-alimanip --dna --outformat afa Results_Integron_Finder_
↪mysequences/tmp_ACBA.0917.00019.0001/ACBA.0917.00019.0001_attc.res
>ACBA.0917.00019.0001/315102-315161
GTCTAACAATTC---GTTCAAGCCGACGCCGCT---------------------------
----------------------TCGCGGCGCGGCTTAACTCAAGC----GTTAGAT
>ACBA.0917.00019.0001/313260-313368
ACCTAACAATTC---GTTCAAGCCGAGATCGCTTCGCGGCCGCGGAGTTGTTCGGAAAAA
TTGTCACAACGCCGCGGCCGCAAAGCGCTCCGGCTTAACTCAGGC----GTTGGGC
>ACBA.0917.00019.0001/313837-313906
GCCCAACATGGC---GCTCAAGCCGACCGGCCAGCCCT----------------------
-----------------GCGGGCTGTCCGTCGGCTTAGCTAGGGC----GTTAGAG
```

The possible outformat are:

- stockholm
- pfam
- a2m
- psiblast
- afa

1.1.5 web server

Galaxy

You can access IntegronFinder online, on the Galaxy server of the Pasteur institute

How to use it

Registration on the Galaxy server of the Pasteur institute is not required to use the tool. Yet, if you wish to keep your history, we recommend you to register.

1. Upload your sequence with **Get Data - Upload File** in the menu on the left
2. Select your file in the **Replicon file** list of Integron Finder
3. Select the options you want
4. Click on **Execute**

If you want more options:

3. Select **Show** on advanced parameters
4. Select the options you want
5. Click on **Execute**

You can see the role of the different functions in the *tutorial* page.

Results

Once the job is finished, you get your results on right panel. All files contain the log of the run which tells you how many integrons have been found for each types along with the number of *attC* sites per type. There are 4 different outputs created:

- **Raw results archive:** An archive containing all raw results.
- **Integrons annotations:** A tabular file listing all the elements and their caracteristics.
- **GenBank:** The GenBank file of the input sequence with the annotation corresponding to the elements found (integrase, *attC*, promoter, attI, etc…).
- **Graphics:** Simple representation of one or more complete integrons found. The representation is very basic and a better representation can be obtained from the GenBank file and a software (eg Geneious) to represent it.

For each of the aforementioned files, you can save them by clicking on the download button.

1.1.6 References

If you use this software, please cite:

- Identification and analysis of integrons and cassette arrays in bacterial genomes Jean Cury; Thomas Jove; Marie Touchon; Bertrand Neron; Eduardo PC Rocha. **Nucleic Acids Research**, 2016; doi: 10.1093/nar/gkw319

Please cite also the following articles:

- Nawrocki, E.P. and Eddy, S.R. (2013) Infernal 1.1: 100-fold faster RNA homology searches. **Bioinformatics**, 29, 2933-2935.
- Eddy, S.R. (2011) Accelerated Profile HMM Searches. **PLoS Comput Biol**, 7, e1002195.
- Hyatt, D., Chen, G.L., Locascio, P.F., Land, M.L., Larimer, F.W. and Hauser, L.J. (2010) Prodigal: prokaryotic gene recognition and translation initiation site identification. **BMC Bioinformatics**, 11, 119.

and if you use the function *–func_annot* which uses *NCBIfam-AMRFinder* hmm profiles:

- Haft, DH et al., Nucleic Acids Res. 2018 Jan 4;46(D1):D851-D860 PMID: 29112715

CHAPTER 2

Developer Guide

2.1 Developer Guide

This part is for developers, who want to work on IntegronFinder scripts.

2.1.1 Developer installation

If you are not part of the project, start by forking IntegronFinder repository. For that, sign in to your account on github, and go to https://github.com/gem-pasteur/Integron_Finder. Then, click on 'Fork' (under your account icon). This will create a copy of the repository, but with your username instead of 'gem-pasteur'.

create a virtual environment:

```
virtualenv -p python3 Integron_Finder
```

activate you virtualenv:

```
source Integron_Finder/bin/activate
```

then install integron_finder in developer mode:

```
pip install -e "git+https://github.com/gem-pasteur/Integron_Finder
↪#egg=integron_finder[dev]"
```

or clone your repository manually, then install it

```
mkdir src
cd src
git clone https://github.com/gem-pasteur/Integron_Finder
cd Integron_Finder
pip install -e ".[dev]"
```

It installs the requirements and create a directory in the virtualenv src/integron_finder and create links in the virtualenv. So `integron_finder` is runnable and you can modify the sources and run it again without to reinstall the project.

Note: *[dev]* allow to install extra dependencies to generate documentation, compute test coverage …

Warning: Debian/Ubuntu distribution *–user* is the default. So the *–prefix* option does not work and the *–root* option has unexpected behavior. Therefore the best solution is to use *–user* or a virtualenv.

2.1.2 Build a new release

1. activate the virtualenv:

   ```
   ./Integron_Finder/bin/activate
   ```

2. Go to the root of the project:

   ```
   cd Integron_Finder/src/Integron_Finder
   ```

3. Te build the new release:

   ```
   python -m build .
   ```

it will create a source *tar.gz* distribution and a *wheel*

2.1.3 Send changes to upstream repository

If you want to integrate your code in the upstream (main) repository, you need to create a pull request.

1. Read the Contibuting guide
2. Create a new branch with `<your branch name>` a descriptive name (e.g. 'adding-xx-feature', 'fixing-typos', etc.), so that others understand what your are working on.

3. Work on it
4. Test that your work does not break the tests. add tests corresponding to your code
5. Push your local branch on your integron_finder clone on github

   ```
   git push --set-upstream origin <your branch name>
   ```

6. ask for pull request
   - Go to your forked repository on github *https://github.com/<your_login>/Integron_Finder/pulls*
   - Click on 'New pull request'
   - Choose your repository and the branch on which you did your changes in 'head fork' (right-hand side), and choose 'gem-pasteur/Integron_Finder' with the branch on which you want to merge (probably master) in 'base fork' (left-hand side).
   - A green 'Able to merge' text should appear if git is able to automatically merge the 2 branches. In that case, click on 'Create pull request', write your comments on the changes you made, why etc, and save. We will receive the pull request.

2.1.4 Tests

IntegronFinder is provided with unit tests. You can find them in `tests` directory. You can use them to check that your changes did not break the previous features, and you can update them, and add your own tests for the new features.

Tests are done using unittest.

Running tests

To run the tests -v option is to increase the verbosity of the output:

```
python setup.py test
```

or:

```
python tests/run_tests.py -vv
```

or:

```
python tests/run_tests.py -vv tests/test_utils.py
```

to run specific tests.

If you also want to get code coverage (you need to install coverage):

```
coverage run  --source integron_finder tests/run_tests.py
```

Add `-vv` to get more details on each test passed/failed. If you want to see the coverage in html output, run (after executing the command above):

```
coverage html
```

The html coverage report will be generated in `coverage_html/index.html`.

Adding tests

If you want to create a new test file, adding a file in tests directory, must start with `test_`. Then, write your TestCase by inherits from IntegronTest and your tests using unittest framework (see examples in existing files), and *run them*.

2.1.5 Documentation

Documentation is done using `sphinx`. Source files are located in `doc/sources`. To generate the documentation you just have to run the makefile located in *doc* directory.

```
make html
```

To generate the documentation in *html* format or

```
make latexpdf
```

to generate the documentation in pdf format (for this option you need to have latex installed on your compute)

You can complete them.

2.1.6 Architecture Overview

Project files and directories

Files

COPYING The integron_finder licensing.

COPYRIGHT The integron finder copy rights holders.

MANIFEST.in What must be or should not included in the distribution.

README.md The file to red in first.

requirements.txt The requirements need to use integron_finder.

requirements_dev.txt The extra requirements to develop on integron_finder.

setup.cfg The setup.py configuration file.

setup.py The file to define how to build/install/release/test/... integron finder.

Directories

integron_finder The core of the projects contains integron_finder library The **scripts/finder** contain the main entry point.

tests Contains all needed for tests, the tests themselves, are a the top level and the name must start by `test_`. The data directory contains all data needed to perform the tests. (see *Tests* for further details)

doc Contains the documentation write in sphinx. The **source** directory contains the .rst files, whereas the **build** directory contains the generated documentation. To know how to contribute or generate documentation see *Documentation*

Singularity Contains the definition file for singularity container.

data TODO

dist This directory is generated when a distribution is created (`python setup.py sdist`).

Technical overview

The main entry point is in integron_finder/scripts/finder.py there are 3 functions

`intgeron_finder.scripts.main()` which is the real main entry point

`main` call `scripts/finder.parse_args()` which parse the commandline and generate a `config.Config` object. and do a loop over replicon and run `intgeron_finder.scripts/find_integron_in_one_replicon()`

all results are store in a directory named `Results_Integron_Finder_<replicon_file_name>` this directory is created by `intgeron_finder.scripts/find_integron_in_one_replicon()` store results in this directory or in a subdirectory call tmp_<replicon_id> these subdirectories will be keep only if `--keep-tmp` option is set, otherwise they are removed at the end of the `intgeron_finder.scripts/find_integron_in_one_replicon()`

when all replicons are computed the `main` function call `integron_finder.utils.merge_results()` to gather all results files `<replicons_id>.integtrons` and generate a unique file with these information.

to have details on `find_integron_in_one_replicon` works see *Introduction*

2.1.7 Integron_finder API Reference

annotation

attc

config

hmm

infernal

integrase

integron

prot_db

The prot_db module contains classes to handle protein file and protein description which can be either generate by Prodigal or Provide by Gembase. It also provide an interface to abstract the way to get protein sequences and descriptions

results

The *results* module contains functions to handle the final reports.

- merging results of each sequence
- generate a summary
- or filter the calin

topology

class `integron_finder.topology.`**`Topology`**(*default*, *topology_file=None*)

Class to parse and handle replicons topologies

`__getitem__`(*replicon_id*)

Parameters **`replicon_id`** (*`str`*) – The id of the replicon.

Returns the topology for the replicon corresponding to the replicon_id

`__init__`(*default*, *topology_file=None*)

Parameters

- **`default`** (*str*) – the default topology
- **`topology_file`** – the path to the file where topology for replicon are specified

`__weakref__`
list of weak references to the object (if defined)

`_parse`(*topology_file*)
Parse a topology file where topology is specified for replicons on each line a topology is specified for a replicon the syntax of each line is

```
replicon_id topology
```

the allowed value for toplogy are 'circ', 'circular', 'lin', 'linear'

Parameters **`topology_file`** (*str*) – The path to the topology file

`_parse_topology`(*topo*)
Parse a field topology in topology file the authorized values are circular, linear or circ, lin, or in uppercase

Parameters **`topo`** – the field corresponding to topology in topology file

Returns the topology in "normed" format 'circ' or 'lin'

Return type str

utils

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index