DetectIS (v 0.1.1)

Manual

Contents

1 Installation **3**

1.1 Prerequisites to run the analysis . . . 3

1.1.1 Creating a Singularity container . . . 3

1.1.2 Nextflow . . . 4

2 Running the workflow using nextflow **4**

2.1 Structure of the configuration file . . . 4

3 Test data sets and bash script **6**

3.1 Structure of the bash script . . . 6

3.2 detectIS.pl script . . . 6

4 Result interpretation **7**

DetectIS is a pipeline designed to detect the integration sites of exogenous DNA using DNA or RNA paired-end sequencing data. A Singularity container has all the software necessary for the analysis. The workflow manager nextflow can be used to run the analysis locally or in an HPC environment.

1 Installation

You can use git to clone the detecIS repository from GitHub to your computer:

```
git clone https://github.com/AstraZeneca/detectIS.git
```

or alternatively you can download the tarball from GitHub:

```
wget https://github.com/AstraZeneca/detectIS/archive/0.1.1.tar.gz
```

and extract the files:

```
tar -xvf 0.1.1.tar.gz
```

The repository is made of three directories:

- Workflows;
- utils;
- TestDataset.

The *Workflows* directory contains the *detectIS.nf* file, necessary to run the analysis with nextflow. The directory *TestDataset* contains fastq files and reference data that can be used to test detectIS. It also contains the script *Run_detectIS.sh*, that can be used to run the analysis without using nextflow. The directory utils contains the recipe *detectIS.rec*, necessary to build the Singularity container, and the bash and Perl script used to generate simulated data sets.

The configuration file and the bash script have been written to analyse the example dataset present in the TestDataset directory and, can also be used as templates for other analyses.

1.1 Prerequisites to run the analysis

The detectIS software requires Singularity version 2.6 or higher. The installation of Nextflow, version 0.32.0.4897 or higher, is strongly advised.

1.1.1 Creating a Singularity container

Singularity is a container platform that creates and runs containers with the required software in a way that is portable and reproducible. A container is a single file that can be generated using Singularity on a laptop, and executed on HPC clusters, local university or company clusters, servers or cloud.

A Singularity container with all the necessary software is required to run the detectIS pipeline. The image can be created by using the recipe contained in the *utils* directory of the detectIS GitHub repository (https://github.com/AstraZeneca/detectIS/raw/master/utils/detectIS.rec). It is possible to generate a Singularity container from a recipe with the command:

```
sudo singularity build detectIS.simg detectIS.rec
```

Superuser privileges are necessary only to create the container but not to use it. This means you can create the container on your local pc/workstation and copy it to the system where you run the detectIS analyses (e.g. your hpc or cluster).

Singularity containers are kernel-dependent, this implies that the recipes contained in this project will not necessarily produce a container able to run on your HPC system. If none of the available recipes generates a container compatible with your system you might need to modify the recipe using an operating system with compatible kernel, please raise an issue on GitHub if this is the case and you need support for it.

1.1.2 Nextflow

Nextflow is a workflow manager that enables scalable and reproducible scientific workflows using software containers. It requires Bash 3.2 (or later) and Java 8 (or later, up to 15) and it is distributed as a self-contained executable package. See the nextflow manual for further information.

2 Running the workflow using nextflow

Once installed Singularity and Nextflow, if the container has been created and copied to the *utils* directory, you can run the nextflow workflow to analyse the test data set by using the command:

```
nextflow run Workflows/detectIS.nf -c detectIS_TestDataset.conf --with-report \
detectIS_TestDataset_nextflow_report.html
```

The file *detectIS.nf* is the workflow for the detectIS analysis and *detectIS_TestDataset.conf* is the configuration file with all the information needed for the analysis. The workflow file is made by all the instructions used by nextflow for the analysis, it can be used without any change, unless you need to change the default parameters of the Perl script (see the *detectIS.pl script* section for further information).

2.1 Structure of the configuration file

The configuration file is made of two sections: the first one contains information of the HPC/cluster used for the analysis.

```
params.project_name='Test_dataset-detectIS'
```

The variable *params.project_name* specifies the name of the project, in this case Test_dataset-detectIS.

```
process.executor='sge'
```

The variable *process.executor* specifies the executor, there are different options depending by the HPC/cluster where you run the analysis (see https://www.nextflow.io/docs/latest/executor.html# for further information). If you run the process locally you don't need to specify it, or you can set it as 'local'.

```
process.queue = 'infini.q'
```

The variable *process.queue* specifies the queue to use, this depends by the used HPC/cluster.

```
process.clusterOptions = '-S /bin/bash'
```

The variable *process.clusterOptions* specifies options specific of the used cluster, the one in the example forces the cluster to run the job with bash, in case it is not the default option.

```
process.penv = 'smp'
```

The variable *process.penv* specifies the parallel environment to use submitting a parallel task to the SGE resource manager (see https://www.nextflow.io/docs/latest/process.html#penv for further info).

```
params.scratch='/scratch/'
```

The variable *params.scratch* specifies the scratch directory.

```
singularity.enabled = true
process.container = "utils/detectIS.simg"
```

These variables are necessary to use Singularity and to specify the container file.

```
singularity.cacheDir = "/scratch/"
```

The variable *singularity.cacheDir* specifies the singularity cache.

The second part of the configuration file contains analysis specific information.

```
params.reads = "TestDataset/*_{R1,R2}.fq.gz"
```

The variable *params.reads* specifies the sequencing reads to process, the pattern used in the example is used to specify read pairs.

```
params.outdir = "TestDataset/NextflowRes/"
```

The variable *params.outdir* specifies the output directory.

```
params.cpu.minimap = 32
```

The variable *params.cpu.minimap* specifies the cpu used for the mapping with Minimap2.

```
params.host_seq="TestDataset/Cricetulus_griseus_chok1gshd.CHOK1GS_HDv1.dna.toplevel_Scaffold0.fa"
```

The variable *params.host_seq specifies* the reference fasta file of the host genome.

```
params.vir_seq="TestDataset/CelTag.fa"
```

The variable *params.vir_seq* specifies the reference fasta file of the exogenous element (plasmid, viral agent, etc.).

3 Test data sets and bash script

In the directory “TestDataset” are contained paired-end reads and reference files to run a detectIS analysis.The dataset simulates the integration of a plasmid in the genome of Chinese hamster ovary cell line (CHOK1).

3.1 Structure of the bash script

A less valid alternative to Nextflow is a bash script able to run all the steps required by the pipeline. The directory *TestDataset* contains the bash script *Run_detectIS.sh* written for this aim.

The top part of the script specifies variables specific of the analysis:

```
singimg="../utils/detectIS.simg"
```

The variable *singimg* specifies the singularity container used for the analysis

```
gref="./Cricetulus_griseus_chok1gshd.CHOK1GS_HDv1.dna.toplevel_Scaffold0.fa"
extragenome1="./CelTag.fa"
```

These variables specify respectively the host genome and the exogenous references

```
reads=( ./Sim_R1.fq.gz ./Sim_R2.fq.gz )
```

This array specifies the two fastq files to process

```
mkdir -p Res
name="./Res/SimRead"
paramscpu=4
```

The second part of the script run the alignment to the host and exogenous reference and finally the detectIS.pl script to integrate the results. Finally the markdown file produced by the script is converted to pdf and html.

3.2 detectIS.pl script

The “Workflows” directory contains the detectIS.nf file, the nextflow workflow file and the “bin” sub directory with the detectIS.pl script and the modules with all the subroutines used by it. The script processes the 4 alignment results in paf format, looking for split and/or chimeric read pairs able to identify integration sites.

Script arguments:

```
    -h1 name_mate1_gnm.paf
      (Aligment results of R1 reads on the host genome) [Mandatory argument]
    -h2 name_mate2_gnm.paf
      (Aligment results of R2 reads on the host genome) [Mandatory argument]
    -v1 name_mate1_vir.paf
      (Aligment results of R1 reads on the exogenous reference) [Mandatory argument]
    -v2 name_mate2_vir.paf
      (Aligment results of R2 reads on the exogenous reference) [Mandatory argument]
    -o name_prefix (Output prefix) [Mandatory argument]
    -mqual 1
      (Minimum mapping quality to consider hits in the host genome)
      [Not mandatory, range: 0-60 default value: 1]
    -ovlwind 0.05
      (Overlap or distance, as fraction of the read length, tollerated to detect split reads)
      [Not mandatory, range: 0-1 default value: 0.05]
    -mspr 2
      (Minimum number of split reads to identify an integration site)
      [Not mandatory, range: 1+ default value: 2]
```

The script can be executed specifying the mandatory arguments and leaving as default the other arguments. Alternatively the ovlwind can be increased and the mspr can be reduced to increase the sensitivity of the tool, for experiments executed at low coverage.

4 Result interpretation

The detectIS.pl script makes 2 final results: one text file (with the .txt extension) and one markdown file (with the .md extension). The text file can be visualized using the UNIX less command or edited by using any UNIX/MAC text editor like vim nano Emacs or in Windows, notepad. It can also be imported as spreadsheet in Excel or Open Office. The integration sites identified in the analysis are reported in rows with the following information:

1. IS: The integration site with chromosome and position of either host genome and exogenous element.
2. TotSpReads: The total number of split read pairs supporting the integration site.
3. R1R2SpReads: Number of split read pairs supporting the integration site with both read split.
4. R1SpReads: Number of split read pairs supporting the integration site having the R1 read split and the R2 read mapped within 5 read length of the integration site.
5. R2SpReads: Number of split read pairs supporting the integration site having the R2 read split and the R1 read mapped within 5 read length of the integration site.
6. ChimReads: The total number of chimeric read pairs supporting the integration site.
7. SingleSplitRead: Number of split read pairs made only one split read and the other not mapped within 5 read length of the integration site.
8. Interval: Extended interval supporting the integration site. It specifies the relative orientation of host genome and exogenous element and this information is fundamental to correctly design primers for PCR verification of the integration site.

The same information are also contained in the markdown file that can be converted to pdf and/or html. In the directory Workflows/bin is contained a script to convert all the .md file present in the directory with the results to pdf and html files.

```
cd detectIS/Workflows/bin/
bash CreatePDFandHTML.sh /MyResultDirectory
```