The PARMA toolkit - useful tools for NGS data analysis

Overview

The PARMA toolkit provides tools for the analysis of NGS data, especially for (PAR-)CLIP sequencing reads. The most important tool is the mapping tool which embeds the PARMA algorithm for read alignment and applies a best practice pipeline for PAR-CLIP read mapping. The following tools are available in the PARMA toolkit Version 0.5 alpha:

Table 1: Overview of tools accessible through the PARMA toolkit.

Tool	Description
map	Utilizes the PARMA algorithm to map a given sequencing read dataset against a reference sequence; optionally combines mapping against genomic and transcriptomic sequences
comb	Combines the results of genomic and transcriptomic read alignments; recalculates genomic mapping positions for transcriptomic hits
error	Calculates the error profile (mismatches and indels) for an aligned read dataset compared to the reference sequence
clust	Clusters an aligned PAR-CLIP read dataset to obtain RBP-bound genomic regions. Is able to filter T-C conversion sites that are annotated as SNPs in an appropriate database
simulate	Creates a simulated PAR-CLIP read dataset based on observations made for PAR-CLIP sequencing reads
benchmark	Calculates accuracy of an aligned simulated PAR-CLIP dataset
setup	Setup options for the PARMA toolkit, e.g. setting the path to the PARMA algorithm

Getting started

The PARMA toolkit can be downloaded as a pre-compiled jar (java executable) including all dependent libraries (except CPAN Math::Random and samtools, see below).

```
git clone https://github.com/akloetgen/PARMA_tk.git

cd PARMA_tk/bin/

java -jar parma.jar
```

For optimal use of the PARMA toolkit, Java 7, the PARMA algorithm (https://github.com/akloetgen/PARMA), embedded in the open source aligner BWA, samtools (https://github.com/samtools/samtools) and the Perl CPAN Math::Random package (http://search.cpan.org/~grommel/Math-Random-0.71/) are required.

The PARMA algorithm should be included in the PATH environment, otherwise the PARMA toolkit is not able to access the algorithm. Alternatively, you can set up the path to the PARMA installation using the following command:

```
java -jar parma.jar setup --parma myPATH_TO_PARMA
```

The reference sequence index is calculated with the BWA algorithm (so far, only BWA version 0.7.8 is supported). If you are usually using another BWA Version than 0.7.8 and don't want to change your PATH environment, you can also use the PARMA implementation for the BWA location as follows:

```
java -jar parma.jar setup --bwa myPATH_TO_PARMA
```

Alternatively, the source code of the PARMA toolkit can be downloaded and compiled, but additional libraries are required:

- HTSjdk-1.128.jar (http://samtools.github.io/htsjdk/)
- bzip2.jar (http://www.kohsuke.org/bzip2/)
- log4j-1.2.17.jar (https://logging.apache.org/log4j/1.2/download.html) (a newer version 2.1 is available but not yet supported)
- jmathplot.jar (http://code.google.com/p/jmathplot/)

The PARMA toolkit

The basic command for executing the PARMA toolkit is as follows:

```
java -jar parma.jar MODE [options]
```

where *MODE* is one of the tools from Table 1. To print an overview of the available tools, just execute the jar-file without any further options. A more detailed description of every tool can be printed by executing the tool without further options, e.g. as follows for the mapping tool:

```
java –jar parma.jar map
```

which will print the instructions for the mapping tool.

We also provide example files in the subfolder "examples" for tools of the PARMA toolkit and an execution script (bin/examples.sh and bin/examples_remove_temp.sh) which applies every tool to those example files. This will also help to understand the file formats necessary for the individual tools.

Workflow for mapping

First, a BWT-index and a fasta-index for the reference genome sequence have to be created using the index function of the BWA algorithm and samtools as follows:

```
bwa index REFERENCE

samtools faidx REFERENCE
```

Afterwards, the PARMA mapping tool can be executed as follows:

```
java -jar parma.jar map -q INPUT -r REFERENCE -p THREADS -o OUTPUT --refine
```

To allow mapping against multiple databases, the command just needs the indexed transcript reference filename as additional input:

```
java -jar parma.jar map -q INPUT -r REFERENCE -p THREADS -o OUTPUT -t
TRANSCRIPT_REFERENCE --refine
```

where *TRANSCRIPT_REFERENCE* is a multiple fasta file containing sequences of known transcripts for a given organism. For this multiple fasta file, a BWT-index has to be created in a first step, too. It is important that the fasta header of the *TRANSCRIPT_REFERENCE* looks as follows (which could be downloaded e.g. from Ensembl BioMart):

```
>Gene_ID|Transcript_ID|Chr|Exon_start_site1;Exon_start_site2;…|Exon_end_site1;Exon_end_site
2;…|Strand
```

Combine tool

The combination of results of a genomic reference mapping and the results of a transcriptomic reference mapping in the PARMA toolkit is possible using the combine tool. Therefore, the two alignment files must be stored in a BAM-format and are used as input for the tool, as follows:

```
java -jar parma.jar comb GENOMIC_MAPPING TRANSCRIPT_MAPPING OUTPUT
```

The result is saved in the *OUTPUT* file in a BAM-format. Note, that the *TRANSCRIPT_MAPPING* needs a specific format for the fasta-header for each transcript sequence as described above.

Error profile tool

The calculation of the error profile for a given sequence read dataset is possible using the error profile tool of the PARMA toolkit. Therefore, a reference-based read alignment has to be calculated (and stored in a BAM-file) and can be used as input for the error profile tool:

```
java -jar parma.jar error MAPPING REFERENCE MAX_READ_LENGTH
```

Clustering tool

A first postprocessing analysis for (PAR-)CLIP data is the pile-up of aligned reads into clusters representing the RBP-bound regions in the genome. This can be done using the clustering tool which also excludes T-C conversion sites that are annotated as SNP loci in an appropriate SNP database. The additional parameter *MIN_COVERAGE* is necessary to already pre-filter the list of clusters for those that contain at least *MIN_COVERAGE* sequencing reads.

```
java -jar parma.jar clust MAPPING REFERENCE OUTPUT SNP_DB MIN_COVERAGE
```

PAR-CLIP read simulation

As common sequencing read simulators are not applicable to simulate realistic PAR-CLIP reads, we provide a PAR-CLIP read simulator based on PAR-CLIP read specific properties. The results are saved to *OUTPUT_PREFIX*.fastq in the common fastq-format. To achieve sequencing reads for which the genomic positions can be tracked, the header line of the transcript-fasta file should have the following format:
>*TRANSCRIPT_ID*|*CHR*|*TRANSCRIPT_START*|*TRANSCRTIP_END*

The following command executes the PAR-CLIP read simulator:

```
java -jar parma.jar simulate TRANSCRIPTS OUTPUT_PREFIX ERRROR_PROFILE T2C_PROFILE
T2C_POSITION_PROFILE QUALITIES INDEL_PROFILE BOUND_PROB
```

If you get an error such as the following:

```
Can't locate Math/Random.pm in @INC (@INC contains: /etc/perl)

BEGIN failed--compilation aborted at createSimulatedPARCLIPDataset.pl line 5.
```

please make sure the CPAN Math::Random package for Perl is installed correctly and specify the path to the package via the -I option to the simulation tool, such as follows:

```
java -jar parma.jar simulate TRANSCRIPTS OUTPUT_PREFIX ERRROR_PROFILE T2C_PROFILE
T2C_POSITION_PROFILE QUALITIES INDEL_PROFILE BOUND_PROB -I
PATH_TO_MATH_RANDOM
```

Note, that the header of the transcriptome fasta is slightly different to the one used for the transcript mapping (will be updated soon, so that 1 file is enough):

```
>TRANSCRIPT_ID|CHR|TRANSCRIPT_START|TRANSCRIPT_END
```

Benchmarking read alignments of a simulated PAR-CLIP dataset

After a simulated PAR-CLIP read set was aligned against a reference sequence, this tool can assess the alignment accuracy of the respective aligner on PAR-CLIP reads:

```
java -jar parma.jar benchmark MAPPING OUTPUT SIM_READS_FILE
```

Setup

To set up some properties, the setup mode can be executed. So far, the paths to different aligners can be set unless they are already in the environment path:

```
java -jar parma.jar setup --parma PATH_TO_PARMA
```

If any further questions arise or if you note a bug, please contact us: andreas.kloetgen@hhu.de