Trip

Version 0.5

A Perl script for the processing of raw TRIP data

Waseem Akthar, Ludo Pagie, Johann de Jong, Jelle ten Hoeve

Netherlands Cancer Institute, Amsterdam

Introduction

A TRIP experiment produces three different types of sequencing reads: (a) the normalization reads representing barcode counts in genomic DNA, (b) the expression reads representing barcode counts in cDNA and (c) the mapping reads which represent the fragments containing the barcode and the flanking genomic DNA sequence. While the normalization and expression reads are generally single reads, the mapping reads can be either single or paired-end.

Since the barcode sequences are not known a priori, the first step in the TRIP data analysis involves the identification of "genuine" barcodes by analyzing the normalization reads. The abundance of these genuine barcodes in the normalization and expression data is determined, resulting in a set of normalization counts and a set of expression counts (for each genuine barcode). Finally, the pipeline identifies the genomic positions by aligning the mapping reads using Bowtie2. Further analysis of mapping data identifies all the genomic locations associated with each genuine barcode and the proportion of top two most frequently associated locations. Ideally, a barcode should be present only at one genomic location, however; this is not always the case because of re-hopping of the transposon after starting a TRIP pool, artifacts during inverse PCR based mapping procedure and occasionally independent integrations of two reporters with identical barcodes. When executed properly, the output of the analysis consists of a data table with the sequence of all genuine barcodes, their counts in the normalization and expression data, and the most frequent location associated with each barcode. It also gives the total number of reads containing this barcode in the mapping data, the fraction of reads pertaining to the most frequently associated location, the average Bowtie2 alignment quality score of all those reads and, if applicable, also the fraction of reads pertaining to the second most frequently associated locations. In addition to the data table, the program also produces a summary table and some plots (in pdf format) showing summary of statistics.

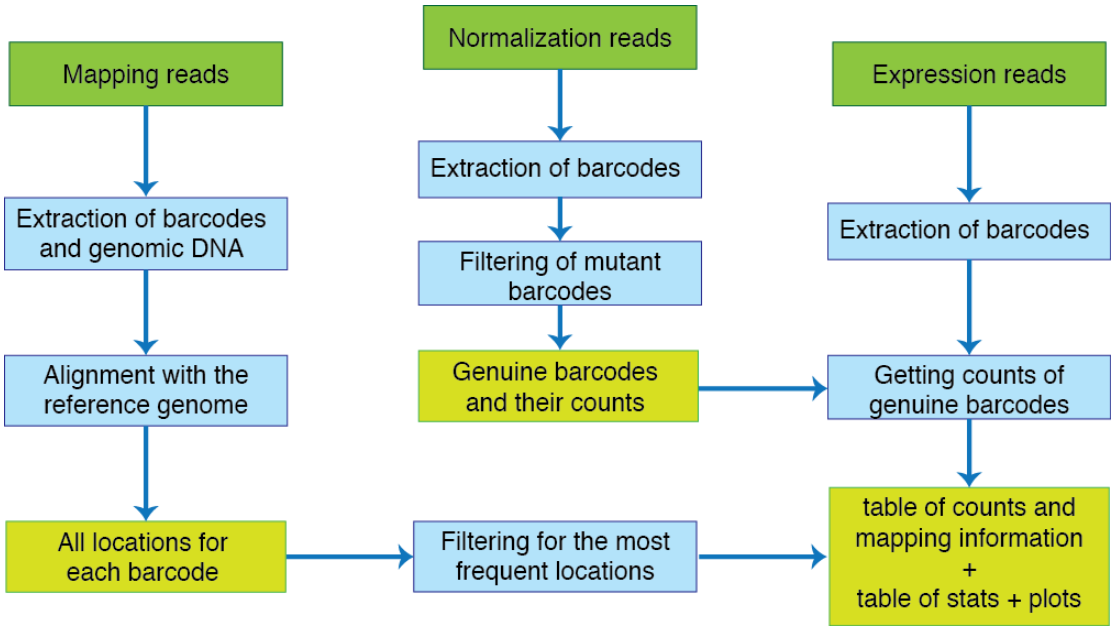

Figure 1: Flow diagram showing the data analysis pipeline for TRIP. The normalization reads are used to define the genuine barcodes. The counts of these barcodes are then determined both in normalization and expression reads. The mapping data is analyzed independently and all genomic locations associated with each of the barcodes is determined. For each barcode, the most frequently associated location, fraction of reads representing this location and related stats are determined. In the end, a table of counts and mapping information, a table of stats and a few useful summary plots are generated.

General structure of reads in TRIP data

Depending on the design of the construct used for TRIP, the reads can vary in their structure. The script presented here can work on any of the three different formats described in Figure 2. The sequence of the different constant segments and the length of the variable segments are given as arguments in the configuration file.

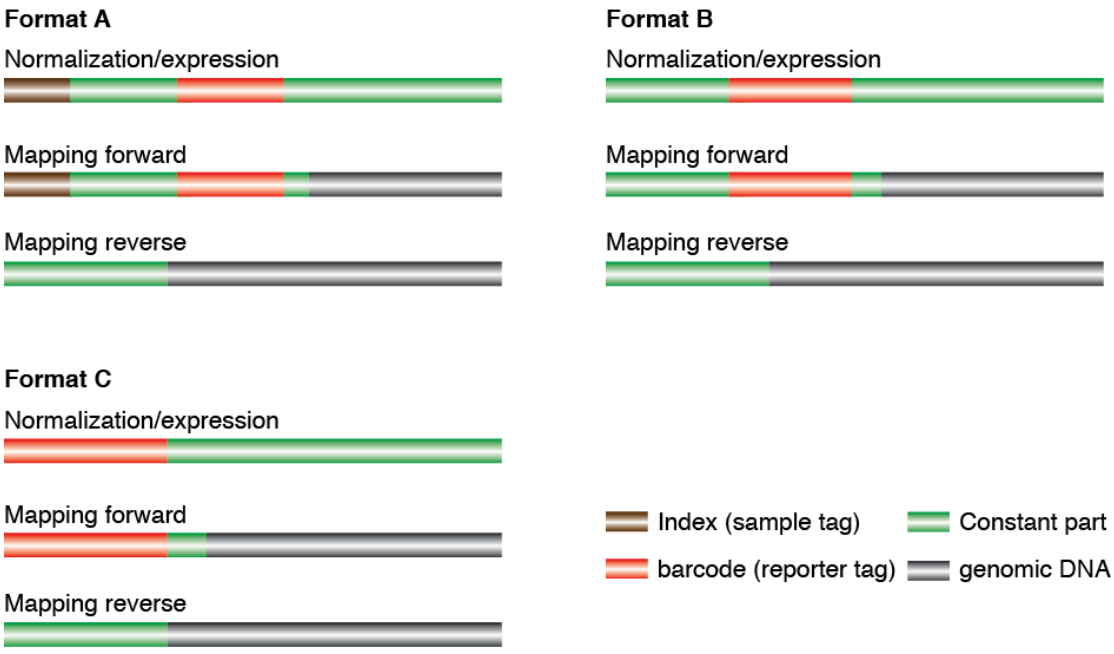

Figure 2: Different formats of read structure in a TRIP experiment. In format A, the read (forward read in case of mapping) starts with an index sequence used to identify different samples pooled on one sequencing lane. In case no index is used or indexing is done in a different way, read structure would be of format B. If the sequencing primer site is already cloned in the TRIP construct, then there is no first constant part in the forward reads and they start directly from the barcode (format C).

Getting Started with TRIP data analysis

Before getting started with the analysis of your data please make sure that you have a 32 or 64-bit computer with 8 GB of RAM (16 GB preferred) and multiple CPU cores (we recommend > 8 CPU cores).

The following softwares are installed on your machine (see installation guide for more details).

- Linux operating system. Mac OS X (10.6 Snow Leopard or more recent) with Xcode and Command Line Tools installed is also fine.
- The latest version of the R programming language (http://www.r-project.org)
- R packages ggplot2 (http://ggplot2.org) and gridExtra http://cran.r-project.org/web/packages/gridExtra/index.html.
- Bowtie 2 software (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml)
- The Bowtie2 index of the genome (of organism of interest) is built. In case of human, mouse or rat, you can download already built index from http://bowtie-bio.sourceforge.net/bowtie2/index.shtml.
- FASTX-Toolkit (http://hannonlab.cshl.edu/fastx_toolkit/)
- The Perl programming language (http://www.perl.org/get.html) version v5.12.4 or later.
- The Perl module Text::LevenshteinXS

The command line options

Following are the options available on the command line with their description.

Option	Input	Description
--normFile	*NORMALIZATION FILE*	- file containing normalization reads. Should be fastq (unzipped) *
--expFile	*EXPRESSION_FILE*	- file containing expression reads. Should be fastq (unzipped) *
--mapFor	*MAPPING_FORWARD*	- file containing forward mapping reads. Should be fastq (unzipped) §
--mapRev	*MAPPING_REVERSE*	- file containing reverse mapping reads. Should be fastq (unzipped) ¶
--config	*CONFIGURATION_FILE*	- file containing extra arguments (for an example see below) *
--out_dir	*OUTPUT_DIRECTORY*	- path to the directory where output should be saved (this directory should exist) *
--map	*[n OR b OR r OR f]*	- should mapping be done from both forward and reverse reads (b) or from forward reads only (f) or from reverse reads only (r). If 'f' only --mapFor needs to be specified otherwise (b or r) both --mapFor and --mapRev needs to be specified. The default value is undefined which means no mapping. In this case the output will be only the barcodes and their counts in normalization and expression data
--help		- prints out usage information
--verbose		- prints out running commentary during the execution of the script
--debug		- prints out the progress of the script into a file debug_file.txt in the output directory

* These options must be given otherwise the script will not work.

§ This option must be given if --map is “f”, “r” or “b”.

¶ This option must be given if --map is “r”, or “b”.

The configuration file

In addition to the arguments given on command line, the TRIP data analysis script also requires an additional set of arguments which can be conveniently provided in a text file. These arguments contain information about the read structure, the information about the genome and some additional arguments related to filtering of genuine barcodes and mapping.

Parameter		Default Value	Description
index_length	=	10	# the length of index sequence. If no index is part of the read (format B according to Figure 2) then it is 0.
barcode_length	=	16	# the length of the barcode. The program looks for barcodes which are barcode_length ± 1.
pat1	=	GTCACAAGGGCCGGCCACAACTCGAG	# the first constant part in the normalization/expression read. Only the four nucleotides (A, C, T or G) are allowed.For reads with no first constant part (format C according to Figure 2), it should be NA or na
pat2	=	TGATCCTGCAGTGTCACCTAAATCGTATGCGGCCGCGAATTCTTACTT	# the second constant part in the normalization/expression read. Only the four nucleotides (A, C, T or G) are allowed.
hd	=	2	# The maximum edit distance to filter out potential mutants. *
map_pat1	=	GTCACAAGGGCCGGCCACAACTCGAG	# the first constant part in the mapping read. Only the four nucleotides (A, C, T or G) are allowed. For reads with no first constant part (format C according to Figure 2), it should be NA or na
map_pat2	=	TGATC	# the second constant part in the forward mapping read. Only the four nucleotides (A, C, T or G) are allowed.
map_pat_rev	=	GTACGTCACAATATGATTATCTTTCTAGGGTT	# the constant part in the reverse mapping read. Only the four nucleotides (A, C, T or G) are allowed.
cores	=	6	# the number of processors to be used for Bowtie2 alignments (use less if you have fewer than 8 cores on your machine)
§ bowtie_base	=	path/mm9/mm9	# the Bowtie2 index of the genome to align the mapping reads against
max_dist_for	=	500	# the maximum distance to cluster the positions on the forward read. Typically it is kept a few hundred bases because during inversePCR of TRIP, two different closely spaced restriction sites can be independently used for mapping the same barcode.
max_dist_rev	=	20	# the maximum distance to cluster the positions on the reverse read. Typically it is kept a few bases just to combine those reads for which the alignment was slightly shifted because of sequencing errors.
min_counts	=	5	# the minimum number of reads for considering a barcode genuine. We recommend that min_counts should be at least 5.

* An hd of 2 will mean that for a frequent barcode all other (less frequent) barcodes only 2 Hamming distance apart will be discarded from the list of genuine barcodes.

§ This means that the index files (six in total: mm9.1.bt2, mm9.2.bt2, mm9.3.bt2, mm9.4.bt2, mm9.rev.1.bt2 and mm9.rev.2.bt2) for the mouse genome assembly 9 (in this case) are in the directory path/mm9.

The best way to make a configuration file is to use the template configuration file (config.txt) provided with this script and change the default values to appropriate ones. Note that white space and text after the "#" does not interfere with the interpretation of the configuration file. If you are using the constructs described in {Akhtar, 2013, 23953119} (GenBank accession number KC710227) and the read length is > 65, then the only two parameters that need to be changed are "cores" and "bowtie_base".

The sample data files and the test run

The sample data files are provided with this script in the directory sample_data. These are four sequencing files typically generated in a TRIP experiment. These files are

Filename	ReadType
normalization_reads.fastq	Normalization reads
expression_reads.fastq	Expression reads
mapping_forward_reads.fastq	Mapping reads (forward)
mapping_reverse_reads.fastq	Mapping reads (reverse)

Copy these files to a directory (for example trip_dir). Also copy the configuration file the script files trip.0.3.5.pl, trip_plot.R from the directory scripts and config.txt to trip_dir.
Type and execute the following command:

```
$ cd path/trip
```

Here path is the complete path to the directory trip_dir.

Create a sub-directory output within trip_dir

```
$ mkdir output
```

Now you are ready to execute the script. Type and execute the following command:

```
$ perl trip0.5.pl --normFile normalization_reads.fastq --expFile
expression_reads.fastq --mapFor mapping_forward_reads.fastq --mapRev
mapping_reverse_reads.fastq --config config.txt --out_dir output --map r --
verbose --debug
```

This command can also be given using the short-hand version of the arguments.

```
$ perl trip0.5.pl -n normalization_reads.fastq -e expression_reads.fastq -f
mapping_forward_reads.fastq -r mapping_reverse_reads.fastq -c config.txt -o
output -m r –v -d
```

On a machine with four CPU cores and 8 GB of RAM, this should take around half an hour. When running for the first time, it is highly recommended to run it with --debug option. This will generate a debug file debug_file.txt in the directory /trip_dir/output.

After the complete execution of the script the directory /trip_dir/output should have following set of files in it.

debug_file.txt

final_TRIP_data_table.txt

stats.txt

summary_results.pdf

You can compare the final_TRIP_data_table.txt, stats.txt and summary_results.pdf with the files provided with the script in the directory sample_output. In case the contents of these newly generated files are matching with the contents of files provided with this script, then everything is in order and you can pursue with the analysis of your own TRIP data. In case the outcome of the script is different than the sample output files provided with this script in the directory sample_output, you can report this to us together with the debug file.

Installation guide

Bowtie2

The alignment program bowtie2 should be installed on your machine and should be on the path. For installing bowtie2 go to http://bowtie-bio.sourceforge.net/bowtie2/index.shtml and download the latest release of bowtie2 for your operating system.

Type and execute the following command:

```
$ export PATH=”/path/bowtie2-2.1.0/bowtie2:$PATH”
```

Here path is the complete path to the directory bowtie2-2.1.0. In order to confirm that bowtie2 is installed and is on your path, type and execute the following command:

```
$ bowtie2
```

This should display the help page of bowtie2. If that is not the case and instead you see a message saying `bowtie2: command not found`, then edit your path file /etc/paths and manually add to this file the full path to the directory bowtie2-2.1.0.

```
$ edit /etc/paths
```

R and the R package 'ggplot2'

Download the latest version of R from http://www.r-project.org and install it on your machine according to the instructions given on the website.

Run R and type and execute the following at the R command prompt.

```
> install.packages("ggplot2")
```

You will be asked to choose a CRAN mirror. After the installation of ggplot2 is complete, type and execute the following command:

```
> install.packages("gridExtra")
```

Perl and the Perl module 'Text::LevenshteinXS'

Generally Perl is installed on Mac OS X and Linux. You can confirm this by typing and executing the following command:

```
$ perl -v
```

This should display the basic information about the Perl version installed on your machine.

In case Perl is not installed on your machine, download and install the latest version of Perl from http://www.perl.org/get.html.

To install the Perl module Text::LevenshteinXS, type and execute the following command:

```
$ sudo cpan Text::LevenshteinXS
```