Xpression Reference
June 27 2011
Harwood Lab

Contents

Synopsis

command-line
python Xpression.py option_file.txt

graphical user interface
java -jar Xpression_GUI.jar

Description

Xpression is a fast, open-source integrated pipeline that automates the analysis of relatively short read-length “next-gen” sequencing data. The raw sequencing fastq file is filtered for sequencing quality and barcoded reads if multiplexing was used to combine samples. This high-quality set of reads is then mapped to a reference genome using the burrows-wheeler aligner, bwa. Mapping quality statistics are output to a text file, providing the number of reads of good sequence quality and of those that are mappable, unique, or non-unique. The bwa alignment file and an annotated genbank file provided by the user are used to calculate expression data, which outputs raw counts as well as counts normalised to total reads (reads pM) and total reads per CDS length (reads pKM). A visualisation file in ‘wig’ format is then generated from the expression data and can be viewed using tools such as the Integrated Genome Viewer and Artemis.

Xpression is primarily coded in Python with some shell scripting and is divided into functionally intact modules for each step. The purpose and design of the pipeline is to be integrated and easy to use, but also easy to modify. It runs on Unix-like systems like MacOSX and Linux. The Windows/CYGwin environment is not supported, has not been tested, and is up the user to troubleshoot. A graphical user interface is available, allowing simple mouse-driven interaction.

System Requirements

Hardware requirements:
Benchmarking was performed on an average modern desktop computer with a 2.33 GHz quad-core CPU and 8 GB RAM running linux kernel 2.6.28-19-generic. There are no specific requirements except for the memory needed by bwa to index the reference genome.

Software requirements:
operating system
Linux or MacOSX (i.e., “Unix-like”)

core
Python 2.x, latest stable release 2.4 or any release 2.5+
bwa 0.5.7+
samtools 0.1.10+
numpy 1.0+
Biopython 1.51+
pysam 0.3+

gui
Graphical desktop environment, e.g., X
Sun/Oracle Java JRE version 1.5/5.0+

Scripts:
All provided scripts must be in the same directory.
count_mapping.awk
count_reads_per_region.py
create_gene_annotation_pickle.py
extract_reads_from_pools.py
MappingStatCalculator.py
NativeReadCounter.py
ReverseComplementReadCounter.py
splitter.sh
StatFileWriter.py
visualize_expression_data.py
Xpression.py

Commands and Arguments

Commands
Xpression arguments are passed to Xpression.py via a text file with each line containing an entry in the form --parameter=value.
As currently configured, all the Xpression scripts must be located within the same directory, and the option file provided as the only argument to the main script, Xpression.py.

Arguments
--input_file=*STR* The path to the FASTQ input sequencing file to be processed.

--reference_fasta=*STR* The path to the FASTA genome file used as a reference for read-mapping alignment. If the fasta file has not been indexed by bwa, this will automatically occur before mapping.

--reference_genbank=*STR* The path to the genbank file used for calculating gene expression. Xpression generates an internally-used Python archive (pickle) of this file and is automatically created if it is not found in the same folder as this genbank file.

--input_type=*STR* Sequencing quality formatting type, with fastq-illumina being illumina version 1.3+ and ‘fastq’ being the older fastq-sanger format. Fastq-illumina is very similar to fastq-solexa, both having an ASCII offset of 64, while the older fastq-sanger having an offset of 33. The pipeline will raise an error

while processing if this format appears to be incorrect. Values: fastq-illumina | fastq.

--allowed_mismatches=*INT* Number of allowed mismatches between a read and the reference genome. Default: 2.

--sample_barcode=*STR* Short barcode unique to the sample if it is part of a multiplexed FASTQ file.

--sample_bioseq_start=*INT* 1-based index of the start of valid biological sequence. If a barcode was used, this field is usually set to the position directly after it. This argument can be set to trim possible poor quality base-positions from the reads to be mapped. Default: 1.

--sample_prep_method=*STR* General library preparation method, assuming a portion of the resulting read is the primer used during cDNA generation and that the read is relatively short since bwa performs poorly on reads longer than ~100bp. RNAseq will allow most library prep methods to have any barcode and specificity combination. spRNAseq trims each read in a specfic way perticular to that technique. For most library construction methods, RNAseq should be selected. Values: spRNAseq | RNAseq. Default: RNAseq.

--sample_strand_specific=STR If strand specificity is maintained by library prep method, output will include positive and negative strand mapping in addition to gene and inter-gene mapping. Values: yes | no.

--sample_read_seq_direction=STR If libary prep method maintains native read orientation relative to reference genome, or if the resulting reads are actually reverse-complements of the reference strand. This argument is disregarded if --sample_strand_specific is ‘no’. Values: native | reverse_complement.

--step=*LIST* Run the step(s) listed. Individual steps can be run or rerun with steps separated by a semi-colon. Steps must be performed sequentially overall. Default: 1;2;3;4

--number_of_cpu=*INT* Number of processes to be run simultaneously while processing input fastq file. A reasonable value is 2 processes per CPU core. Default: 8

--sample_name=*STR* Name of sample used for naming output files.

--output_dir=*STR* Path to where output files are saved to. The output of each step will be saved to an individual folder.

Xpression Workflow

0. Split FASTQ file to be run in parallel. Outputs split files to *0_splitted_FASTQ*.
1. Input sequencing file filtered by various sequencing quality constraints and specified barcode if given. Outputs FASTQ file from quality/barcode filtered reads to 1*_merged_FASTQ.*
2. Align reads to genome using bwa, generate mapping statistics. Outputs bwa alignment to *2_sorted_BAM.* Mapping quality statistics are output to ‘mapping_statistic.txt’.

3. Determine mapping to genes in annotation reference. Outputs a '.csv' text file containing raw and normalised expression data for each locus found in the genbank reference file to 3_*expression_profile.*
4. Generate visualisation files for viewing in IGV or others. Outputs wiggle-formatted '.wig' expression data visualisation file to 4_*visualization.*

Xpression Performance

Hardware
.33 Ghz Intel Core2 Quad Q8200
GB RAM
Linux (kernel 2.6.28-19-generic)

Input
A single sample with a 4-nt barcode from an 8-plexed RNA-seq data test set which included 11.9 million 36-nt single-end reads.

Output
Step 1 - Extract barcodes and filter quality
- seconds to pull out 1.3 million reads having the specified barcode sequence, assess their quality and remove non-biological sequences from the reads.

Step 2 - Align quality reads to reference genome
- seconds to align 1.2 million high-quality reads against a reference genome.

Step 3 - Calculate gene expression
- ,086 seconds to count the number of reads uniquely mapped to each region of the reference genome.

Step 4 - Create expression visualisation file
- seconds to create two visualisation files.

Total elapsed time is 1,271 seconds or 21.18 minutes.

Notes
This pipeline was designed to provide a general functionality for most 'next-gen' sequencing analysis. Improvements, modifications and additional features are welcome as suggestions or the pipeline can be modified by the user to better suit specific uses or preferences. Due to some of the core software like bwa not being available for the native Windows platform, there are no plans for porting or providing support for environments such as cygwin.

License
GNU GPLv3

Citation
Phattarasukol, S., Radey, M., Lappala C., Brittnacher, M., and Harwood, C.S.. 2011. *In press*.

Author
Somsak "Sam" Phattarasukol designed and coded the pipeline.