Rational Function Complexity Manual

Timothy Daley Andrew Smith

January 3, 2013

Chapter 1

Quick Start

The Rational Function Complexity package is aimed at predicting the yield of distinct reads from a genomic library from an initial sequencing experiment. The estimates can then be used to examine the utility of further sequencing, optimize the sequencing depth, or to screen multiple libraries to avoid low complexity samples.

The two main programs are c_curve and lc_extrap. c_curve samples reads without replacement from the given sequenced read file to estimate the yield of the experiment and the subsampled experiments. These estimates are used construct the complexity curve of the experiment. lc_extrap uses rational function approximations of Good & Toulmin's [1] non-parametric empirical Bayes estimator to predict the yield of future experiments, in essence looking into the future for hypothetical experiments. lc_extrap is used to predict the yield and then c_curve can be used to check the yield from the larger experiment.

1.1 Installation

1.1.1 Download

Rational Function Complexity is available at
`http://smithlab.cmb.usc.edu/software/`.

1.1.2 System Requirements

Rational Function Complexity runs on Unix-type system with GNU Scientific Library (GSL), available at `http://www.gnu.org/software/gsl/` and GNU Complilation Collection (GCC) (if you would like to compile it yourself), available at `http://gcc.gnu.org/`. If the input file is in bam format, bamtools is required, available at `https://github.com/pezmaster31/bamtools`. If the input is a text file of counts or is in bed format, bamtools is not required. It has been tested on Linux and Mac OS-X.

1.1.3 Installation

Download the source code and decompress it with

```
 $ tar -jxvf RationalFunctionComplexity.tar.gz
```

Enter the RationalFunctionComplexity/ directory and run

```
$ make all
```

If the input is in bam format and the root directory of bamtools is $bamtools, instead run

```
$ make all BAMTOOLS_ROOT=$bamtools
```

If compiled successfully, the executable files are available in **RationalFunctionComplexity/**.

1.2 Using Rational Function Complexity

1.2.1 Basic usage:

To generate the complexity plot of a genomic library from a sorted read file in .bed format, use the program *c_curve*. Use -o to specify the output name.

```
$ c_curve -o output.txt input.bed
```

To estimate the future yield of a genomic library using an initial experiment in .bed format, use the program *lc_extrap*. The required options are -o to specify the output of the yield estimates and the input file, which is either a .bed file sorted by chromosome, end position, start position, and strand or a .bam file sorted with bamtools or samtools sort function. If the input is in bam format, then the flag -B must be included. Additional options are available and are detailed below.

```
$ lc_extrap -o yield.txt input.txt
```

1.3 File Format

Input files are sorted mapped read files in bed or bam format, or a text file consisting of one column giving the observed read counts. To ensure compatibility with other Smith Lab software, the programs require that bed files are sorted by chromosome, end position, start position, and strand. This can be achieved by using the command line function sort as follows:

```
sort -k 1,1 -k 3,3n -k 2,2n -k 6,6 input.bed > input.sort.bed
```

.bam format read files should be sorted by chromosome and start position. This can be done with either samtools (available at `http://samtools.sourceforge.net/`) or bamtools sort functions.

For more general applications we allow the input to be a text file of observed read counts, one count per line. Such a text file can typically be constructed by command line arguments. Take for example an unmapped sequencing experiment in fastq format. We shall use the first 20 bases of each unmapped read as the unique molecular identifier. A command line argument to construct the counts would then be

```
awk '{if (NR%4==2) print substr($0,1,20);}' input.fastq | sort | uniq -c

| awk '{print $1}' > counts.txt
```

Chapter 2

Detailed usage

2.1 c_curve

c_curve is used to compute the expected complexity curve of a mapped read file by subsampling without replacement and counting the distinct reads. Output is a text file with two columns. The first gives the total number of reads and the second the corresponding number of distinct reads.

-o, -output Name of output file, default prints to screen

-v -verbose Prints more information

-B, -bam Input file is in bam format

-V, -vals Input is a text file of read counts

2.2 lc_extrap

lc_extrap is used to compute the expected yield for theoretical larger experiments and bounds on the number of distinct reads in the library and the associated confidence intervals, computed by bootstrapping. Output is a test file with four columns. The first is the total number of reads, second gives the corresponding average expected number of distinct reads, and the third and fourth give the lower and upper limits of the confidence interval. Specifying verbose will print out the histogram of the input file.

-o, -output Name of yield output file, defaults prints to screen.

-e, -extrapolation_length The maximum number of total reads to compute yield estimates for. The default is 10 billion reads.

-s, -step The step size between yield estimates. Default is 1 million reads.

-b, -bootstraps The number of bootstraps used to compute the confidence intervals, default is 100.

-c, -cval Level for confidence intervals. Default is 0.95.

-v, -verbose Print more information.

-B, -bam Input file is in bam format.

-V, -vals Input is a text file of read counts.

Bibliography

[1] I. J. Good and G. H. Toulmin. The number of new species, and the increase in population coverage, when a sample is increased. *Biometrika*, 43:45–63, 1956.