The **preseq** Manual

Timothy Daley Chao Deng Terence Li Andrew Smith

August 8, 2020

Contents

1 Quick Start

The **preseq** package is aimed to help researchers design and optimize sequencing experiments by using population sampling models to infer properties of the population or the behavior under deeper sampling based upon a small 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 four main programs are `c_curve`, `lc_extrap`, `gc_extrap`, and `bound_pop`. `c_curve` interpolates the expected complexity curve based upon a hypergeometric formula and is primarily used to check predictions from `lc_extrap` and `gc_extrap`. `lc_extrap` uses rational function approximations of Good & Toulmin's [**?**] non-parametric empirical Bayes estimator to predict the library complexity of future experiments, in essence looking into the future for hypothetical experiments.

`gc_extrap` uses a similar approach as `lc_extrap` to predict the genome coverage, i.e. the number of bases covered at least once, from deeper sequencing in a single cell or low input sequencing experiment based on the observed coverage counts. An option is available to predict the coverage based on binned coverage counts to speed up the estimates. `gc_extrap` requires mapped read or bed format input, so the tool `to-mr` is provided to convert bam format read to mapped read format.

`bound_pop` uses a non-parametric moment-based approach to conservatively estimate the total number of classes in the sample, also called the species richness of the population that is sampled.

2 Installation

Download

preseq is available at `http://smithlabresearch.org/software/preseq/` or `https://github.com/smithlabcode/preseq`. Only clone the repo if you are planning to modify or reuse the code. Most users should download a release. The README.md file in the GitHub repo explains how to download a release, and not simply the source code files.

System Requirements

preseq runs on Unix-type systems. If the input file is in BAM format, the HTSLib API is required (http://www.htslib.org/download/) but is included in all binaries and source code. If the input is a text file of counts in a single column or is in BED format, SAMTools is not required. It has been tested on Linux and Mac OS-X.

Installation: Instructions on how to install preseq are included in the README.md file in root of the source tree for preseq. If you have problems installing preseq, please contact Andrew Smith at `andrewds@usc.edu`

3 Using preseq

Basic usage

To generate the complexity curve of a genomic library from a read file in BED or BAM format or a duplicate count file, use the function `c_curve`. Use `-o` to specify the output name.

```
$ ./preseq c_curve -o complexity_output.txt input.bed
```

To predict the complexity curve of a sequencing library using an initial experiment in BED format, use the function `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, start position, end position, and strand or a BAM file sorted with the samtools sort function. Additional options are available and are detailed below.

```
$ ./preseq lc_extrap -o future_yield.txt input.bed
```

For a low input sequencing experiment the genomic coverage is highly variable and uncertain function of sequencing depth. Some regions may be missing due to locus dropout or preferentially amplified during whole genome amplification. `gc_extrap` allows the level genomic coverage from deep sequencing to be predicted based on an initial sample. The input file format need to be a mapped read (MR) or BED, sorted by chromosome, start position, and end position. Additional options are available and are detailed below.

```
$ ./preseq gc_extrap -o future_coverage.txt input.mr
```

4 File Format

Sorted read files in BED or BAM 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. The programs require that BED files are sorted by chromosome, start position, end position, and strand. This can be achieved by using the command line function sort as follows:

```
sort -k 1,1 -k 2,2n -k 3,3n -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 the SAMTools sort function. If the input is in BAM format, then the flag `-B` must be included.
If the input is paired end, the option `-P` can be set. In this case concordantly mapped reads and disconcordantly mapped fragments are counted. This means that both ends of a disconcordantly mapped read will each be counted separately. If a large number of reads are disconcordant, then the default single end should be used or the disconcordantly mapped reads removed prior to running **preseq**. In this case only the mapping location of the first mate is used as the unique molecular identifier [**?**].

Text files of observed read counts

For more general applications **preseq** allows the input to be a text file of observed read counts, one count per line. To specify this input, the option `-V` must be set.
Such a text file can typically be constructed by command line arguments. Take for example an unmapped sequencing experiment in FASTQ format. To predict the complexity, the unique molecular identifier needs to use only the observed sequence. For instance, a unique molecular identifier used may be the first 20 bases in the observed sequence. 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
```

More complicated unique molecular identifiers can be used, such as mapping position plus a random barcode, but are too complicated to detail in this manual. For questions with such usage, please contact us at `andrewds@usc.edu`

Mapped read format for `gc_extrap`

`gc_extrap` does not allow for input files to be in BAM format. We have found that some mappers give inconsistent SAM flags for paired end reads, preventing efficient merging of reads in the proper order. We provide the tool `to-mr` to convert SAM or BAM format files to MR format. The MR or BED format file needs to be sorted by chromosome, start, and end position before input into `gc_extrap`.

5 Detailed usage

c_curve

`c_curve` is used to compute the expected complexity curve of a mapped read file with a hypergeometric formula [**?**]. 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
-s, -step	The step size for samples. Default is 1 million reads
-v -verbose	Prints more information
-B, -bam	Input file is in BAM format
-P, -pe	Input is a paired end read file
-H, -hist	Input is a text file of the observed histogram
-V, -vals	Input is a text file of read counts

lc_extrap

`lc_extrap` is used to generate the expected yield for theoretical larger experiments and bounds on the number of distinct reads in the library and the associated confidence intervals, which is computed by bootstrapping the observed duplicate counts histogram. Output is a text 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 counts histogram of the input file.

-o, -output	Name of output file. Default prints to screen
-e, -extrap	Max extrapolation. Default is 1×10^{10}
-s, -step	The step size for samples. Default is 1 million reads
-n, -bootstraps	The number of bootstraps. Default is 100
-c, -cval	Level for confidence intervals. Default is 0.95
-x, -terms	Max number of terms for extrapolation. Default is 100
-v -verbose	Prints more information
-B, -bam	Input file is in BAM format
-P, -pe	Input is a paired end read file
-H, -hist	Input is a text file of the observed histogram
-V, -vals	Input is a text file of read counts
-Q, -quick	Quick mode, option to estimate yield without bootstrapping for confidence intervals
-D, -defects	Defects mode, estimates the complexity curve without checking for instabilities in the curve. Should only be used on datasets that fail estimation without defects.

gc_extrap

For single cell or low input sequencing experiments `gc_extrap` is used to extrapolate the expected number of bases covered at least once for theoretical larger experiments. Input format is required to be in mapped read format and we have provided the tool `to-mr` to convert bam format files to mr. Output is a text file with four columns. The first is the total number of sequenced and mapped bases, second gives the corresponding expected number of distinct bases covered, and the third and fourth give the lower and upper limits of the confidence interval. Specifying verbose will print out the coverage counts histogram of the input file.

-o, -output	Name of output file. Default prints to screen
-w, -max_width	max fragment length, set equal to read length for single end reads
-b, -bin_size	bin size. Default is 10
-e, -extrap	Maximum extrapolation in base pairs. Default is 1×10^{12}
-s, -step	The step size in bases between extrapolation points. Default is 100 million base pairs
-n, -bootstraps	The number of bootstraps. Default is 100
-c, -cval	Level for confidence intervals. Default is 0.95
-x, -terms	Max number of terms for extrapolation. Default is 100
-v -verbose	Prints more information
-D, -bed	Input file is in BED format without sequence information
-Q, -quick	Quick mode, option to estimate genomic coverage without bootstrapping for confidence intervals

bound_pop

`bound_pop` is a method for estimating species richness, the total number of species or classes in the sampled population. Input format is the same as `lc_extrap`. Default output is a three column text file, with the first column containing the estimated species richness and the second and third containing the estimated lower and upper confidence intervals. If `bound_pop` is run in quick mode, then the output is two columns. The first column will contain the estimated species richness and the second column will contain the dimension or order of the approximation.

-o, -output	Name of output file. Default prints to screen
-p, -max_num_points	Maximum number of points to use in the quadrature estimator. Default is 10, corresponding to 20 entries of the counts histogram being used.
-t, -tolerance	Numerical tolerance for convergence of QR algorithm. Default is $1\text{E} - 20$.
-n, -bootstraps	The number of bootstraps. Default is 100.
-c, -clevel	Level for confidence intervals. Default is 0.95.
-v -verbose	Prints more information.
-B, -bam	Input file is in BAM format.
-P, -pe	Input is a paired end read file.
-H, -hist	Input is a text file of the observed histogram.
-V, -vals	Input is a text file of read counts.
-Q, -quick	Quick mode, option to estimate species richness without bootstrapping for confidence intervals.

6 lc_extrap Examples

Usage and output of `c_curve` is similar, so the following examples are of `lc_extrap` and its different options.

Using a sorted read file in BED (or BAM with the `-B` flag) format as input

```
$ ./preseq lc_extrap -o future_yield.txt input.bed
```

TOTAL_READS	EXPECTED_DISTINCT	LOGNORMAL_LOWER_95%CI	LOGNORMAL_UPPER_95%CI
0	0	0	0
1000000.0	955978.6	953946.4	958015.1
2000000.0	1897632.0	1892888.4	1902387.5
3000000.0	2829410.5	2819146.4	2839712.0
4000000.0	3751924.0	3732334.5	3771616.2
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
9999000000.0	185394069.4	76262245.8	450694319.0

This example uses a sorted read file in BED format from an initial experiment generated from single sperm cells. As noted above, the default step size between yield estimates is 1 million, the default confidence interval level is 95%, and the default extrapolation length is 10 billion.

Using a sorted read file in BED format as input, including the verbose option

```
$ ./preseq lc_extrap -o future_yield.txt input.bed -v
```

As `lc_extrap` is running, information will print to screen that gives a read counts histogram of the input file which truncates after the first bin value that has zero observations. Included here is the first 10 lines of what would be observed:

```
TOTAL READS     = 536855
DISTINCT READS  = 516200
DISTINCT COUNTS = 48
MAX COUNT       = 269
COUNTS OF 1     = 511413
MAX TERMS       = 100
OBSERVED COUNTS (270)
1       511413
2       2202
3       597
.
.
```

Using a sorted read file in BED format as input, with options

```
$ ./preseq lc_extrap -e 15000000 -s 500000 -b 90 -c .90 -o future_yield.txt input.bed
```

TOTAL_READS	EXPECTED_DISTINCT	LOGNORMAL_LOWER_90%CI	LOGNORMAL_UPPER_90%CI
0	0	0	0
500000.0	481098.5	480329.1	481869.1
1000000.0	956070.6	954493.7	957650.2
1500000.0	1428183.4	1425461.7	1430910.2
2000000.0	1897886.0	1892501.7	1903285.7
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
14500000.0	12932529.0	12056525.8	13872180.9

Notice the slight changes, with the step sizes of the extrapolation now at 500,000 as specified, and the maximum extrapolations ending at 15,000,000. The confidence intervals are now at a level of 90%.

Using a histogram or read counts as input

`lc_extrap` allows the input file to be an observed histogram. An example of the format of this histogram is as followed:

```
1       1.68166e+07
2       4.55019e+06
3       1.93787e+06
4       1.07257e+06
5       708034
6       513134
7       384077
8       282560
9       206108
10      146334
```

The following command will give output of the same format as the above examples.

```
$./preseq lc_extrap -o future_yield.txt -H histogram.txt
```

Similarly, both `lc_extrap` and `c_curve` allow the option to input read counts (text file should contain ONLY the observed counts in a single column). For example, if a dataset had the following counts histogram:

```
1      4
2      3
3      1
```

Then, the corresponding input file of just read counts could be as such:

```
1
2
1
1
3
2
2
1
```

Command should be run with the `-V` flag (not to be confused with `-v` for verbose mode):

```
$./preseq lc_extrap -o future_yield.txt -V counts.txt
```

7 gc_extrap Example

gc_extrap is designed for coverage extrapolation in single cell whole genome sequencing experiments. For illustrative purposes we will examine an MDA whole genome sequencing experiment, SRA accession SRR1777281. This experiment has 5.76 million paired end 101 base pair reads. We mapped the experiment with bowtie2 v0.0-beta7 under default parameters. This resulted in 3.63 million concordantly mapped fragment pairs and 2.1 million disconcordantly mapped fragments.
The first step is to convert the sorted bam file to mr format and sort it.

```
$ ./to-mr -o SRR1777281_bwt2.mr -L 10000 SRR1777281_bwt2.sort.bam
$ sort -k 1,1 -k 2,2n -k 3,3n SRR1777281_bwt2.mr > SRR1777281_bwt2.sort.mr
```

The resulting mapped read has 813 million bases (note that bases covered by two fragments of the same read are only counted once) and 410 million covered bases.
As a default, gc_extrap divides the genome into 10 base pair non-overlapping bins. In default mode, the running time of gc_extrap was under 12 minutes.

```
LOADING READS
MAPPED READ FORMAT
TOTAL READS         = 5726883
BASE STEP SIZE      = 1e+08
BIN STEP SIZE       = 1e+07
TOTAL BINS          = 8.1325e+07
BINS PER READ       = 14.2006
DISTINCT BINS       = 4.09582e+07
TOTAL BASES         = 8.1325e+08
TOTAL COVERED BASES = 4.09582e+08
MAX COVERAGE COUNT  = 79775
COUNTS OF 1         = 3.13723e+07
OBSERVED BIN COUNTS (79776)
1       3.13723e+07
2       6.97799e+06
3       1.72101e+06
.
.
```

The output is as follows:

TOTAL_BASES	EXPECTED_COVERED_BASES	LOWER_95%CI	UPPER_95%CI
0	0	0	0
100000000.0	64522380.0	63075879.1	66002053.1
200000000.0	123422455.0	120747705.7	126156454.1
300000000.0	178054120.0	174319619.1	181868626.2
400000000.0	229008295.0	224352188.5	233761032.3
500000000.0	276727080.0	271265011.6	282299130.1
.	.	.	.
.	.	.	.
999900000000.0	1826891418.6	1621208958.1	2058668772.4

To run gc_extrap at single base resolution, the option -b 1 is required. This results in a significant increase in the running time of the algorithm. For this case the running time was 113 minutes.

```
$ ./preseq gc_extrap SRR1777281_bwt2.sort.mr -o SRR1777281_bwt2_1bp_gc_extrap.txt -b 1 -v

LOADING READS
MAPPED READ FORMAT
TOTAL READS         = 5726883
BASE STEP SIZE      = 1e+08
BIN STEP SIZE       = 1e+08
TOTAL BINS          = 8.13236e+08
BINS PER READ       = 142.003
DISTINCT BINS       = 4.09454e+08
TOTAL BASES         = 8.13236e+08
TOTAL COVERED BASES = 4.09454e+08
MAX COVERAGE COUNT  = 80028
COUNTS OF 1         = 3.13527e+08
OBSERVED BIN COUNTS (80029)
1       3.13527e+08
2       6.98288e+07
3       1.722e+07
.
.
```

TOTAL_BASES	EXPECTED_COVERED_BASES	LOWER_95%CI	UPPER_95%CI
0	0	0	0
100000000.0	64680427.0	64284593.4	65078698.0
200000000.0	123709880.0	122978429.6	124445680.9
300000000.0	178447901.0	177427065.8	179474609.6
400000000.0	229488768.0	228216803.2	230767822.1
500000000.0	277279369.5	275788078.2	278778724.8
.	.	.	.
.	.	.	.
999900000000.0	1838021604.0	1682515315.7	2007900543.3

8 bound_pop Example

We examine T-Cell β repertoire (TCRβ) sampling data downloaded from `http://mitcr.milaboratory.com/datasets/aging2013/` and viewed through MiTCR (`http://mitcr.milaboratory.com/`). (**note: link seems to be dead**) The first column corresponds to the observed TCRβ counts. This can be summarized in a duplicate counts histogram, easily formed by using the `hist` function in R.
We will first examine the experiment L1_10_M9. The first ten entries of the duplicate counts histogram is as follows:

```
1       433541
2       56447
3       13030
4       4120
5       2841
6       1981
7       1338
8       970
9       814
10      632
```

There are 861,156 total observed TCRβ sequences and 520,315 distinct TCRβ sequencing. We

```
$ ./preseq bound_pop -H L1_10_M9_hist.txt -o L1_10_M9_estim_species_richness.txt
```

log_mean_estimated_unobs	log_lower_ci	log_upper_ci
2976677.3	2862770.9	3095116.0

In quick mode, the output is as follows:

quadrature_estimated_unobs	n_points
2969646.3	2

preseq estimates that there are at least two million unobserved TCRβ sequences in the sample. This implies that the vast majority of the TCRβ sequences are unobserved in this sample.

9 preseq Application Examples

Screening multiple libraries

This section provides a more detailed example using data from different experiments to illustrate how **preseq** might be applied. Because it is important to avoid spending time on low complexity samples, it is important to decide after observing an initial experiment whether or not it is beneficial to continue with sequencing. The data in this example comes from a study (accession number SRA061610) using single cell sperm cells amplified by Multiple Annealing and Looping Based Amplification Cycles (MALBAC) [**?**] and focuses on three libraries coming from different experiments from the study (SRX205369, SRX205370, SRX205372).
These libraries help show what would be considered a relatively poor library and a relatively good library, as well as compare the complexity curves obtained from running `c_curve` and `lc_extrap`, to show how `lc_extrap` would help in the decision to sequence further. The black diagonal line represents an ideal library, in which every read is a distinct read (though this cannot be achieved in reality). The full experiments were down sampled at 5% to obtain a mock initial experiment of the libraries, as shown here, where we have the complexity curves of the initial experiments generated by `c_curve`:

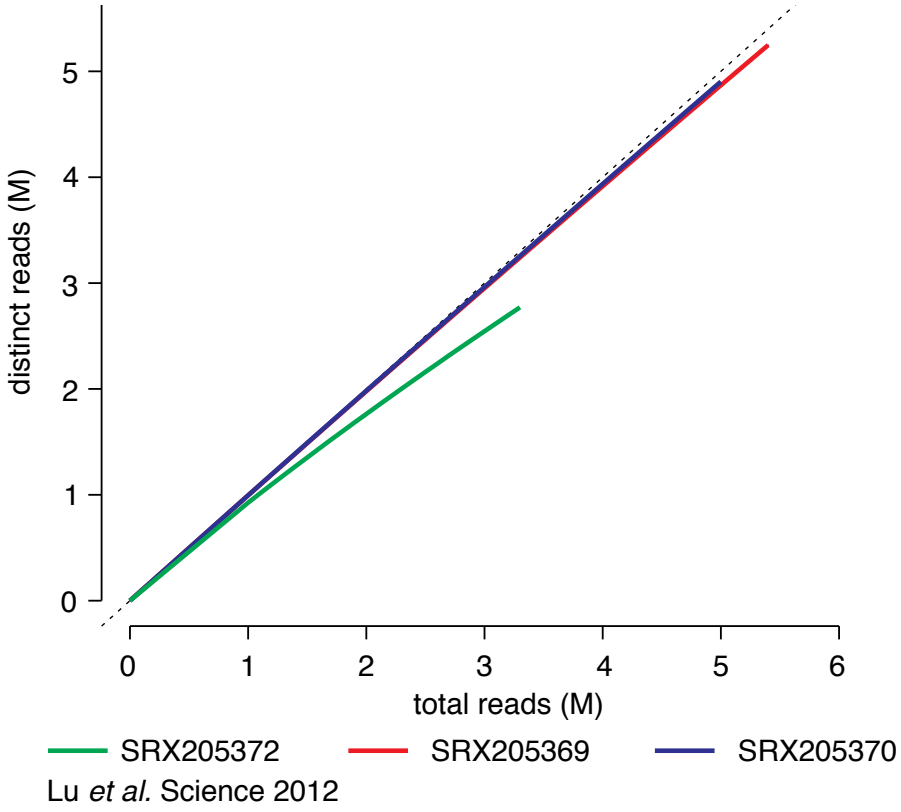

Figure 1: Initial observed complexities

With such a relatively small amount of reads sequenced, it is hard in the first stages of a study to guess at whether it is not worth sequencing a library further, as all three libraries seem to be relatively good. This is a comparison of the full experiment complexity curves and the extrapolated complexity curves created using information from the initial experiments above as input. The dashed lines indicate the complexity curves predicted by `lc_extrap`, and the solid lines are the expected complexity curves of the full experiments, obtained using `c_curve`. Note that the dashed curves follow the solid curves very closely, only differing slightly towards the end, meaning `lc_extrap` gives a good predicted yield curve. Using this, it is clear that if the initial experiments were the only available data and `lc_extrap` was run, SRX205372 would likely be discarded, as it is a poor library, and SRX205369 and SRX205370 would probably be used for further sequencing, as their complexity curves indicate that sequencing more would yield enough information to justify the costs. If the researcher were to only want to sequence one library deep, then SRX205370 would be an obvious choice.

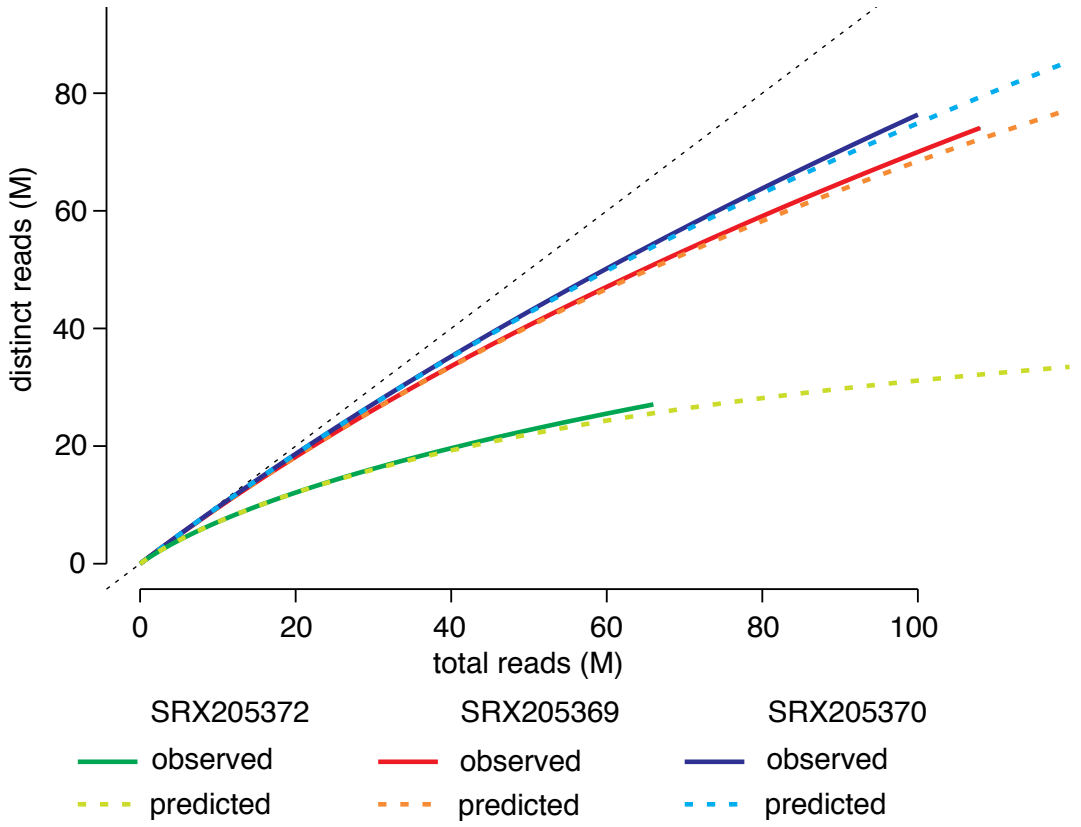

Figure 2: Estimated versus observed library complexities.

Saturation of reads and junctions for RNA sequencing experiments

A recent paper from the Rinn lab [**?**] developed a targeted capture RNA sequencing protocol to deeply investigate chosen portions of the transcriptome. A comparison of the results from a standard RNA sequencing experiment (RNA-seq; SRA accession SRX061769) and a targeted capture RNA sequencing experiment (Capture-seq; SRA accession SRX061768) reveals a startling amount of transcriptional complexity missed by standard RNA sequencing in the targeted regions. A large number of rare transcriptional events such as alternative splices, alternative isoforms, and long non-coding RNAs were newly identified with the targeted sequencing.

A current vigorous debate exists on whether these rare events are truly transcriptional events or are merely due to sequencing or transcriptional noise (see [**?**] and [**?**]). We do not seek to address these issues, but merely to comment on the estimated complexity of rare transcriptional events in sequencing libraries identified by current protocols.

We took the two Illumina sequencing libraries from [**?**] and mapped them according to the protocol given. We downsampled 10% of the library and compared the estimated library complexities (single end) with the observed library complexity for both libraries. We also took the junction information contained in the file junctions.bed in the Tophat output folder to estimate the junction complexity. Since the 5th column (excluding the first line) is the number of times each distinct junction is observed, we can simply cut out these values as input for `lc_extrap` or `c_curve` with the flag `-V`. A simple command line example follows.

```
sed '1d' tophat/junctions.bed | cut -f 5  > junction_vals.txt
./preseq lc_extrap -V junction_vals.txt -o junction_vals_extrap.txt
```

The output `TOTAL_READS` column will be in terms of the number of total junctions (not reads), so scaling by the average number of junctions per read will give the appropriate scale for plotting on the x-axis.

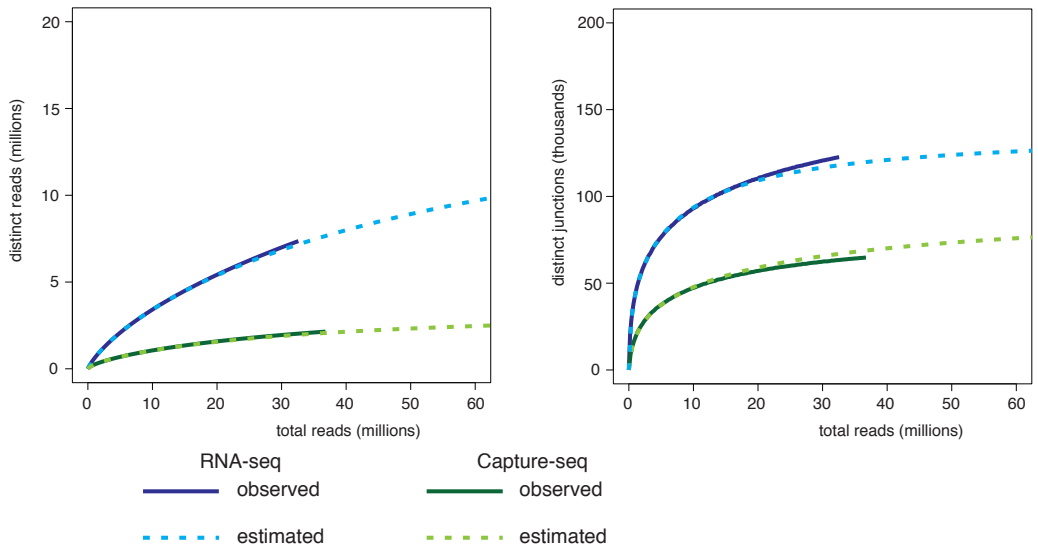

Figure 3: A comparison of complexities of standard RNA-seq and targeted capture RNA-seq. Estimated complexities for both cases were estimated using 10% of the data.

We see from the estimated library that the RNA-seq library is far from saturated, while it appears that the Capture-seq library may be close. On the other hand, the junction complexity of both libraries indicates that the full scope of juctions identified by Tophat is far from saturated in both libraries. This indicates that large number of rare junctions still remain to be identified in the libraries.

Comparing coverage for single cell whole genome sequencing library preparations

We will use `gc_extrap` to compare coverage profiles across library preparation protocols. We use the data from Fu *et al.* [**?**]. We downloaded SRA accessions , SRR1777243, SRR1777245, SRR1777251, SRR1777274, and SRR1777281. These correspond to bulk, DOP-PCR, eWGA, MALBAC, and MDA. We mapped the reads with bowtie2 to hg19 and ran `gc_extrap` on the libraries with default parameters. To compare the libraries we scaled the x-axis to the total number of sequenced reads by dividing by the number total bases (TOTAL BASES in verbose mode) and multiplying by the number of sequenced reads.

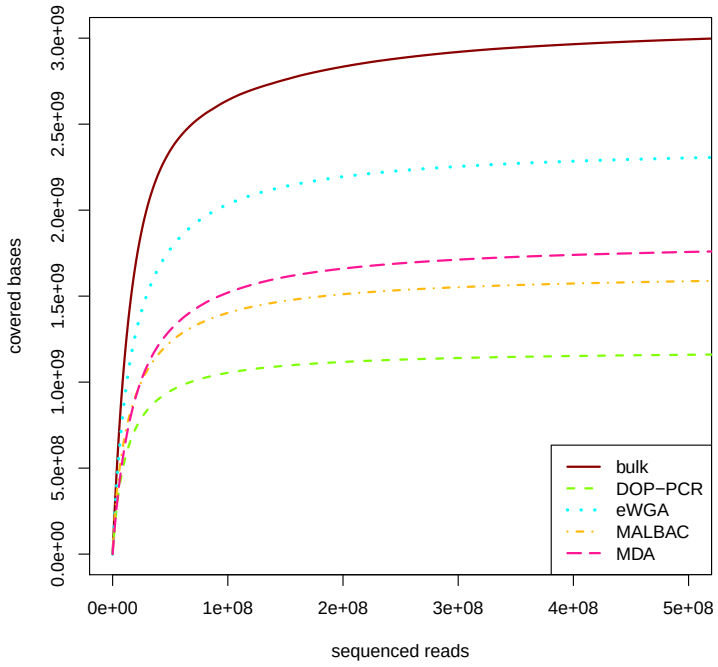

Figure 4: Coverage comparison of single cell whole genome sequencing experiments.

In our previous paper [**?**] we compared coverage for DOP-PCR, MALBAC, and MDA. We found that typically DOP-PCR results in the lowest coverage and MDA the highest. The results for the libraries investigated here agree with our previous results. Notably, eWGA had the highest estimated coverage. This indicates that more investigation of this new library prep is worthwhile to determine if the observed results hold in general.

Estimating and analyzing TCRβ richness

In this section we will use `bound_pop` to estimate TCRβ richness from TCRβ sequencing data taken from Britanova *et al.* [**?**]. We examined one of the datasets in section 8 but now we investigate all 39 TCRβ sequencing experiments.
The average number of observations, or sampled TCRβ sequences, was 992,359 with a range of (861,156, 999,024) indicating that all experiments were sampled to a similar depth. In contrast, the range of the number of distinct TCRβ sequences observed was quite large with a minimum of 133,464, a maximum of 772,223, and a mean of 461,797.
The observed TCRβ richness is obviously biased so we used `bound_pop` to estimate the total richness. The estimated TCRβ richness ranges from a minimum of 823,223 to a maximum of 8,945,797 with a mean of 2,969,305.
Note that any statistical test for relationship will be liberal since the estimation procedure adds extra variance that is not apparent from the estimated values [**?**]. A standard test will tend to reject more often when the null hypothesis is false and will tend to accept less often when the null hypothesis is true. Therefore we can use standard tests to determine where to investigate further.
First there appears to be no difference in estimated TCRβ richness between the sexes of the participants (mean of 3,296,501 from 19 females versus 2,658,468 from 20 males). We next investigate the relationship between the age of the participant and the estimated TCRβ richness, a topic investigated by Britanova *et al.*

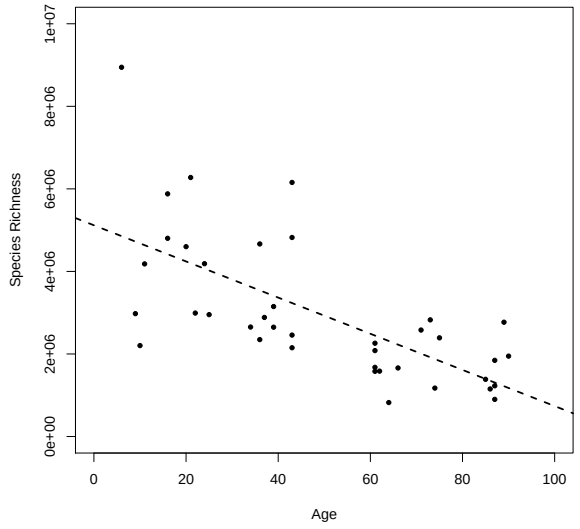

Figure 5: Estimated TCRβ richness versus age and the least square fit. $R^2 = 0.452$, intercept $= 5,117,862$, slope $= -43,802$.

We see that there is a strong negative relationship between the estimated species richness and age. Interestingly we see a lack of homoscedasticity in the data, as the data shows a noticable decrease in variation as a function of age. Clearly further investigation into the relationship between immune repertoire and age will be of interest.

10 FAQ

1. Q — When compiling the **preseq** binary, I receive the error
`fatal error: gsl/gsl_cdf.h: No such file or directory`

A. — The default location of the GSL library will be in '`/usr/local/include/gsl`'. Open the Makefile and append "`-I /usr/local/include`" after `CXX = g++`. You may be receiving this error because the GSL library is not installed on the default search paths of your compiler, and you will need to specify the location.

2. Q — When compiling the **preseq** binary, I receive the error
```
Undefined symbols for architecture x86_64:
   "_packInt16", referenced from:
      _deflate_block in bgzf.o
   "_packInt32", referenced from:
      _deflate_block in bgzf.o
   "_unpackInt16", referenced from:
      _bgzf_read_block in bgzf.o
      _check_header in bgzf.o
```

A. — Go to the SAMTools directory and open the file bgzf.c. Find the functions `packInt16`, `unpackInt16`, and `packInt32`. Comment out the "inline" before each function name.

3. Q — I compile the **preseq** binary but receive the error
`terminate called after throwing an instance of 'std::string'`

A. — This error is typically called because either the flag -B was not included to specify bam input or because the linking to SAMTools was not included when compiling **preseq**. To ensure that the linking was done properly, check for the flag `-DHAVE_SAMTOOLS`.

4. Q — When running `lc_extrap`, I receive the error
`ERROR: too many iterations, poor sample`

A. — Most commonly this is due to the presence of defects in the approximation which cause the estimates to be unstable. Setting the step size larger (with the flag `-s`) will help to avoid the defects. The default step size is 1M reads or 0.05% of the input sample size rounded up to the nearest million, whichever is larger. A consequence of this action will be a reduction in the observed smoothness of the curve.

5. Q — When running `lc_extrap`, I receive the error
`sample not sufficiently deep or duplicates removed`

A. — There may be two causes for this, either duplicates have been removed and every observed read is distinct or there is not sufficient variation in the library for `lc_extrap` to run.

The information required by `lc_extrap` is essentially the number of times each distinct read was observed, which we call the duplicate counts. Without sufficient variation in the duplicate counts we cannot extrapolate the complexity of the library. We have set the minimum required max duplicate count (the largest number of times any read has been observed) to 4. If the input library does not satisfy this, then either a parametric model such as a Poisson or Negative Binomial may be appropriate or deeper sequencing may be required.

6. Q — When running `lc_extrap`, I receive the error
`Library expected to saturate in doubling of size, unable to extrapolate`

A. — A simple two-fold extrapolation using the Good-Toulmin power series, which is within the radius of convergence and therefore rational function approximation is not needed, is performed to ensure that the sample is not overly saturated. If the Good-Toulmin formula is negative, this indicates that the library will likely completely saturate by doubling the experiment size and so extrapolation is not needed. Often this will occur if the number of reads observed twice (n_2) is greater than the number of reads observed once (n_1). In this case one can use simple estimators like Chao's [**?**] ($n_1^2/2n_2$) or Zelterman's [**?**] ($1/(exp(2n_2/n_1) - 1)$) can be used to estimate the number of remaining in the library.

If none of these solutions worked, please email us at `andrewds@usc.edu` and please include the standard output from running **preseq** in verbose mode (specifically the duplicate counts histogram) so that we can look into the problem and rectify problems in future versions. Also, feel free to email us with any other questions or concerns. The **preseq** software is still under development so we would appreciate any advice, comments, or notification of any possible bugs. Thanks!