BarSeq : Counting molecular barcodes in HT-Seq Data.

Preamble:

Each mutant in the yeast-deletion collection has been constructed with distinct molecular barcodes that allow assessment of relative deletion strain abundance across the full collection in a single pooled sample. Based on the success of this approach, molecular barcodes have been incorporated into other collections, allowing high-throughput assessment of the effects of genome-wide deletions and modifications, over-expression and shRNA silencing in a variety of systems.

Resolution of individual barcode abundance in a mixed pool has traditionally been done with custom microarrays. The application of high-throughput sequencing to this platform has many advantages, that include i) direct identification of the individual barcode sequences found within the pool, versus inference based on signal generated after immobilization to a fixed position on an array and ii) the ability to run analysis across multiple samples on a single flow-cell via multiplexing strategies, possible because of the high cluster densities.

While the approach to counting barcodes might seem straightforward, there are a number of issues that have to be given consideration to ensure both in the counting as well as preventing loss of data. It is expected that sequences generated through HT-sequencing strategies will vary from the input templates due to sequencing errors (mismatches, indels). The expectation is that this will make up a small percentage of all reads generated from a given barcode. Of bigger concern though is the introduction of variation during preparation of the sample resulting in a variant of the expected sequence forming clusters on the flow-cell. These issues require a more robust approach to identification of sequencing reads than simply matching to an expected sequence.

Barcode collections vary in size, but there is a tendency to oversample so as to accurately capture barcode numbers that occur in lower numbers. Given the relative small degree of heterogeneity of the input material, there is a much larger degree of redundancy in the data that is generated than observed in other applications of HT-sequencing (rna-seq, chipseq). Analysis of this data benefits from first reducing the dataset to address this redundancy as well as the variation in sequences generated, prior to counting of the barcodes expected in a given sample. These benefits include reduction in analysis time and reducing the degree of misidentification of reads within the dataset.

Our barseq pipeline consists of the following steps:

1. <u>Sample preparation and sequencing</u>
    a. Barcodes in DNA from pooled strains within a single sample are amplified using common primers, incorporating an appropriate index barcode to allow multiplexing
    b. For the yeast deletion collection, which has two sets of barcodes for each strain (uptag and downtag), the barcodes are amplified in separate reactions using the same index barcode prior to recombining the.

c. Amplified Samples are pooled appropriately for multiplexing, then sequenced in a lane of a flowcell.
d. Sequencing protocols may vary depending on what other samples have been loaded on the flowcell. In general, 21 nt of barcode sequence is captured, but the read may be longer. There is an advantage to increasing the number of cycles further as this will generally traverse a common priming site, and this information can be used to filter out noise from the dataset.
e. The index barcode is generally captured in a second short 8 nt read, although this may be incorporated as part of the first read.

2. Demultiplexing of samples run within the same flowcell lane.

The Illumina software (Casava/OLB) has built in tools for demultiplexing. This allows categorization of reads into separate files, in fastq format, organized by sample and project, and is integrated with the basecall conversion. A single base mismatch in the 8 nt index read is permitted. This does require that the indices use differ sufficiently so that they can be resolved.

This process is similar in approach to a custom demultiplexing routine that has been used previously in our analysis, starting with converted basecalls in qseq format, but which is being discontinued.

3. Barcode Counting

The input for barseq counting is a dataset of reads and quality scores, either in qseq or fastq format, for each sample. Counting involves a number of steps, each generating various files, some of which are used for input to the next step in the analysis pipeline.

Sequence priming for our process is initiated directly of one of the flanking common primers so the first cycle enters directly into the barcode. To account for phase shifts that might result in sequence from either of the flanking common primer regions, reads are trimmed to 18nt, eliminating the first cycle and all sequence after cycle 19. These will ultimately be matched to 18mer sequences generated from each of the 20mer barcodes in the expected barcode pool (note, while most barcodes are 20mers, there is some variation between 18-21nt)

a. *Clustering of trimmed reads (18mers)*
The reads generated are expected to correspond to the barcodes in the pool that has been analyzed, allowing for sequencing errors that include both base mismatches and indels. The expectation is that the non-erroneous true-sequence will be generated with a higher frequency. The size of the set of variant sequences derived from the true-sequence will depend on the how abundant the true-sequence is in the pool. Effective clustering of reads around the true-sequences is dependent on there

being adequate distance between the input barcode sequences. Clustering is performed as follows:

i. Currently, quality scores are used only to remove reads from consideration, but not for cluster (see notes on this at the end)
ii. Distinct reads are tallied.
Typically, the resulting list will include two subsets. The first will be sequences seen with high to non-minimal (>1 or 2) counts, the number of which will approximate the size of the expected pool. The second subset will be a much larger set of reads, seen only a once or twice across the whole dataset.
iii. Each read is score based on the number of single base matches that are observed in the dataset.
iv. A core is defined as a read that scores higher than any of its single base mismatches observed in the dataset. All single-base mismatches are assigned as being derived from that core.
v. Reads with a score of 0 (ie. no single base mismatches observed in the dataset), are assigned to clusters as two-base mismatches where possible.
vi. Remaining reads, not assigned to a cluster, are designated as single read clusters and added to the dataset. This might include other derived sequences with more extensive sequencing errors. Despite not assigning these to the proper core, they are still available for counting in the cluster identification step.
vii. The number of times each cluster is seen in the dataset (core + mm1 + mm2 components) is summed to get a count for each cluster.
viii. A cluster file is generated from each qseq/fastq file, indicating statistics on the clustering process and a list of core sequences, their derived sequences and the associated counts.

Downstream assessment of the cluster members shows that identification of a cluster with multiple barcodes in the expected pool is a rare event, which may result when input barcodes are too close in sequence.

I am investigating other approaches to clustering the reads to identify the "true", error free sequences in the data, that will account for base quality and allow for more extensive sequencing errors such as indels. One publication takes an approach to clustering of short reads that accounts for quality scores and uses a statistical assessment to generate the clusters. (Qu et al, Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing. Genome Res. 2009. 19: 1309-1315). I am currently assessing use of their software in their process, and determining if I can utilize aspects of their approach in our process.

b. *Cluster Identification*
The clustering algorithm is run independent of any knowledge of the pool of barcodes that are expected to be found in the data. Identifying each cluster and single read generated from this algorithm is the next step. Input is the cluster file and pool of barcodes that are expected to be found. There may be advantages to using a larger set, if there is a possibility that other barcodes may be present resulting from the preparation process, but keeping this as trim as possible will decrease the possibility of misidentification. Generating an identify for each of the clusters is done in the following manner
    i. Each 18 mer core is matched to a pool of possible 18mers generated from the expected pool. It is best to have an understanding of expected orientation of the sequences and enforce this. For our mixed up and down pools, the orientation of the defined sequence are reverse complemented from each other, due to the method by which they are amplified.
    ii. If a match is found, then it is recorded. If multiple matches are found to a single cluster, they are all recorded.
    iii. Matches are then found for each of the mm1 and mm2 sequences associated with each cluster, if any.
    iv. If neither the core nor the children match any of the barcodes, than candidate identities are generated by identifying common 4mers shared between the core and all barcodes.
    v. Alignment to the candidate barcodes using a custom Needleman-wunsch dynamic programming approach. The highest scoring alignment is selected and assigned to the cluster.
    vi. Each cluster is designated to match in one of the following ways [core, mm1, mm2, align, none]. The classification is modified to indicate an ambiguous match if there are matches to multiple barcodes among the core, mm1 or mm2 matchings.
    vii. The output from this process is a single file, similar to the cluster file, but with the match designation and identities included.

c. *Count Extraction*
The extraction step is to filter the identified clusters to remove ambiguous identifications and weak alignments. In addition, the data is rearranged to account for all barcodes in the expected barcode pool, even those that do are not found in the dataset, as well as their strain assignments.
    i. Each barcode in the expected pool is compared to the identified clusters and any the counts from all clusters that have been associated with that barcode are summed exluding
        1. Matches marked as ambiguous.
        2. Alignments with scores that do not pass a pre- threshold (currently I use a score of 65, ..the value is dependent on the sequence length, and this threshold seems to eliminate

alignments which seem less "real"…by eye. A better assessment is likely needed.

d. *Data merging*
One final step is merging of data from multiple samples. We generally assess a variety of samples, all run with the same pool, across a multiplexed lane or even across multiple lanes of a flowcell. This step generates two files, one a matrix of "total counts", from the Count extraction output, with row= strain and column=sample, as well as a matrix of statistics, from the files generated by count extraction.

Final Notes
The current process is implemented completely in perl. These scripts are still in development and require a degree of editing prior to running without custom editing. Most of the functions are already in a separate library, but I'll move this all forward in the next few weeks so that they can be distributed.

I expect that there might be benefit to rewriting some of this code is either java or c++, mostly in terms of speed, although at this point the bottleneck appears to be the aligment process which relies on external software. Clustering may also be sped up considerably.

April 4, 2012
Author : Lawrence Heisler
Computational Scientist, HipHop Chemogenomics Laboratory
Bioinformatics and Computational Lead, Donnelly Sequencing Centre
Donnelly Centre for Cellular and Biomolecular Research
University of Toronto
160 College St, Rm 1212
Toronto ON, M5S 3E1
Canada
office: (416) 978-5408
fax: (416) 978-4842
http://chemogenomics.med.utoronto.ca**,**

Appendix 1 : Output file formats

Cluster File

1. Summary information on the process (total sequences, unique sequences, clusters formed, mm1 reads clustered, mm2 reads clustered, single read clusters)
2. A list of cluster core sequences with the following fields
    a. Core

b. Score (number of single mismatches)
c. Highest score for all children
d. Total counts for all members
e. Count for core sequence
f. Number of mm1 children
g. Total Counts for mm1 children
h. List of mm1 sequences (seen in data)
i. Number of mm2 children
j. Total Counts for mm2 children
k. List of mm2 sequences (seen in data)

Assignment File

1. the same as that generated by the clustering algorithm but with all the child sequences removed and the following additional fields
    a. Barcode identity (strain name, or other identifier)
    b. Type of match (from point v above)
    c. If match = align, a list of revisions and alignment scores

Barcode File

1. The output has a single line for each barcode indicating
    a. the strain (and tag for our yeast collection, which has two barcodes per strain, designated as “up” and “dn”)
    b. the expected barcode sequence
    c. the total counts from all assigned clusters
    d. the number of clusters (clusters or single reads)
    e. the counts from all clusters assigned by a match
    f. the number of clusters assigned by a match
    g. the counts from all clusters assigned by alignment
    h. the number of clusters assigned by alignment
    i. a list alignment revisions and scores that were accepted
2. In addition, two accessory files are generated
    a. A list of clusters and assignments and were rejected, with counts and cluster size indicated
    b. A file with statistics on the complete process, including the clustering information recorded previously and extended to include number for clusters and counts accepted and rejected.