Yeast cis-eQTL barcode mapping final report

Kimberly Insigne

Friday, April 24, 2015

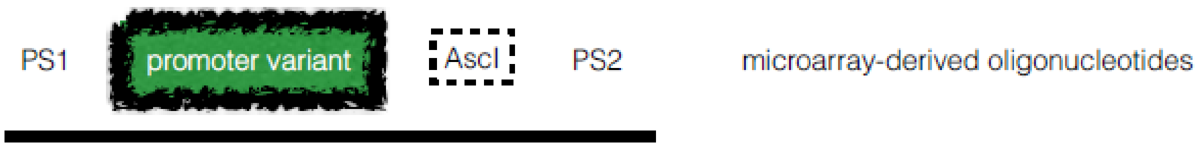
Overview

Goal: Determine which barcodes are suitable for downstream analysis of RNA-seq.

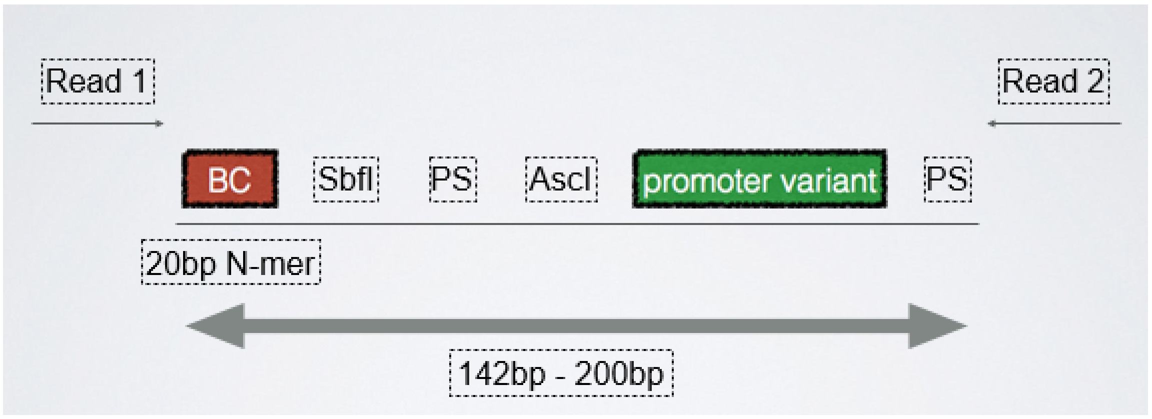
Ideally we want barcodes that:

- match to a perfect reference sequence (no synthesis errors)
- have enough coverage
- and only map to one variant (or a group of very close sequences).

Reference oligo structure:



Read structure:



Important: The reads are a reverse complement of the sequences in the reference file.

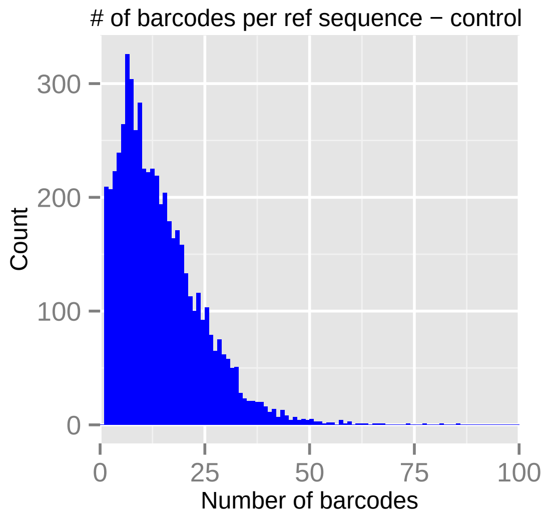
Barcode mapping general statistics

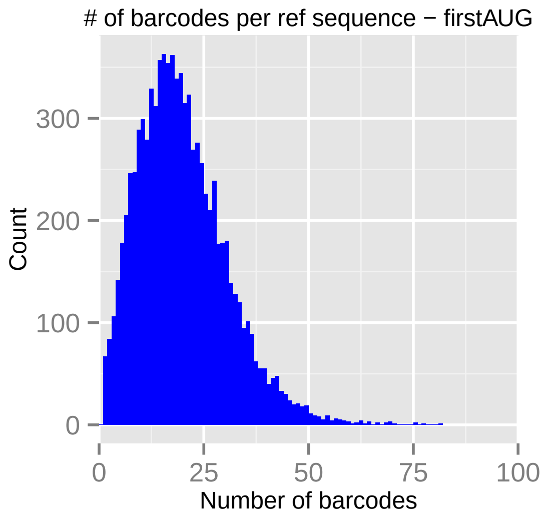
Overview of dataset:

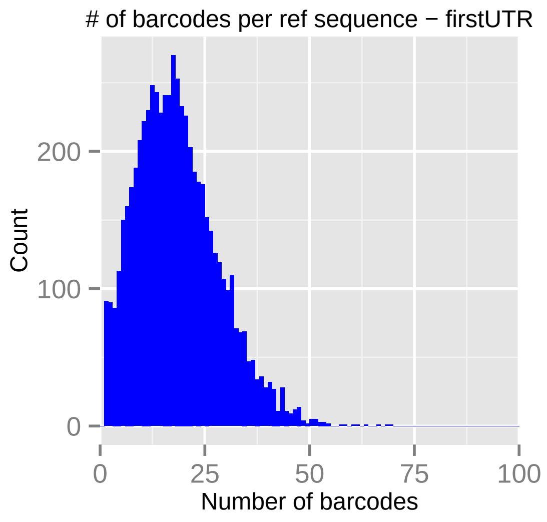
- Total number of oligos: 45,623
  - 6,000 control (5,923 unique)
  - 8,986 lib2-firstAUG (8,975 unique)
  - 6,271 lib3-firstUTR
  - lib4 not performed

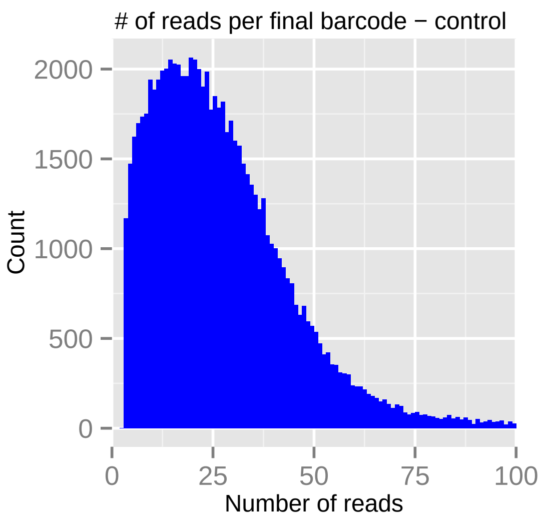
Number of reads: 25,748,575
Number of reads that perfectly matched reference: 39%

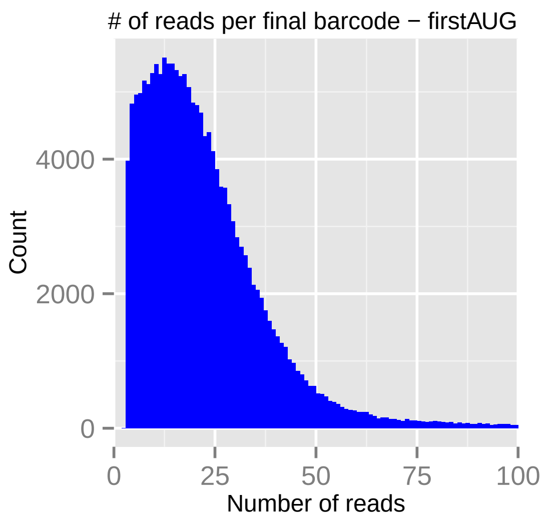
statistic	control	lib2-firstAUG	lib3-firstUTR
Number of barcodes	114,931	205,704	140,190
Number of barcodes >= 3	85,588	170,690	110,354
Number of final barcodes	79,412	168,811	108,186
% of potential barcodes kept	92.7%	98.9%	98.0%
% reference covered	5635/5923 = 95.1%	8781/8975 = 97.8%	6071/6271 = 96.8%
Median number of barcodes per variant	12	18	17
Median number of reads per final barcode	24	19	29

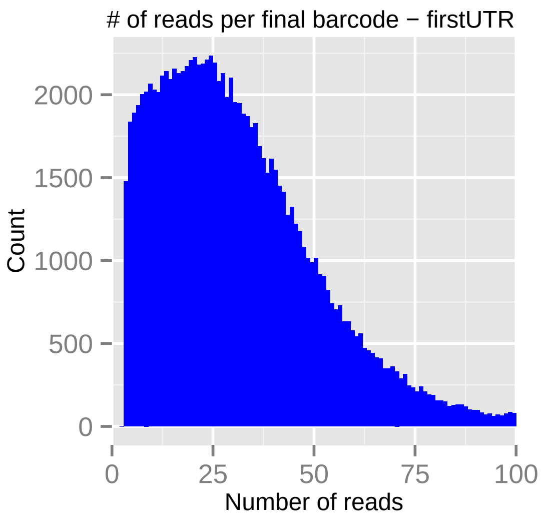












Running the analysis

All the analysis described below is automatically performed by `barcodes_for_RNA_seq.py`. This script takes in four arguments: `<reads_file> <reference_file> <library> <output>`.

- `<reads_file>`: FASTQ file containing all reads from all libraries that have already been processed with PEAR.
- `<reference_file>`: CSV that contains the reference sequences. The first field must be the name of the sequence and the second field must contain the sequence.
- `<library>`: The name of the library to be analyzed. These must match the names in the first field of the CSV file. Valid arguments in this study are: all, sharon, firstTile-AUG, firstTileUTR (sharon is the name of the control library)
- `<output>`: Output file name. The script outputs a list of the final barcodes.

Example usage:
```
python barcodes_for_RNA_seq.py master_reads.fastq tiledDesign4order_141101_noNBases.csv
sharon control_final_barcodes.txt
```

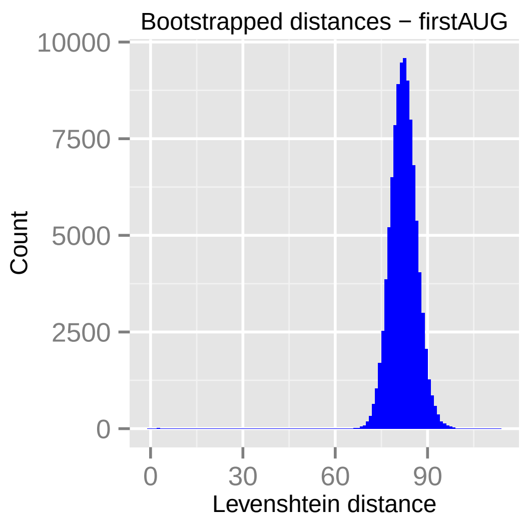
Workflow:

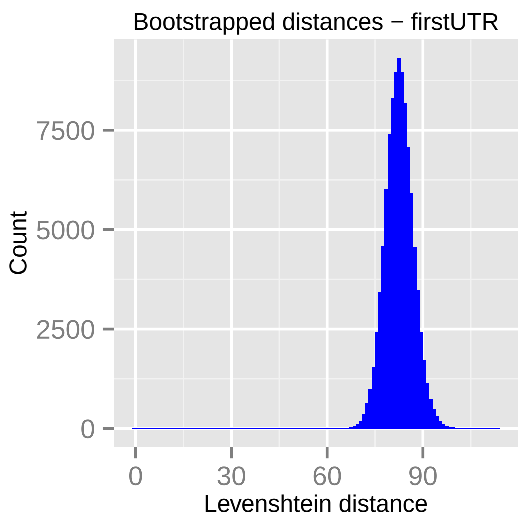
The paired-end reads must first be merged together with PEAR. Files denoted R1 are forward reads and R2 are reverse reads. Additional arguments: `-m 220 -n 100 -j 30` (max size = 220, min size = 100, 30 processors). Next, the assembled reads files from all libraries are concatenated into one master file which is used as input by the analysis script.

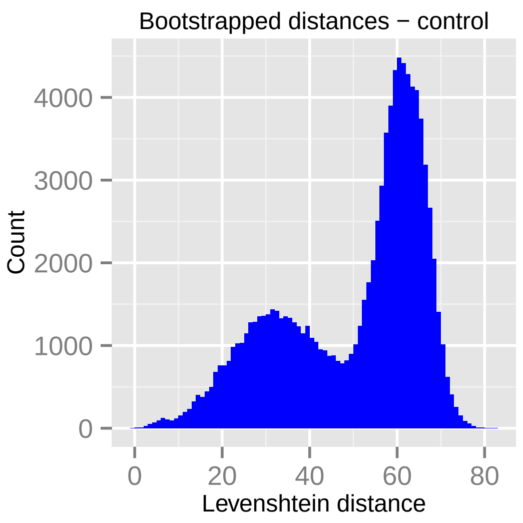
The script described above performs the following analysis:

- Only keep reads that perfectly match a reference sequence (won't catch synthesis errors).
- Of those barcodes that map to a reference sequence, only keep barcodes that appear at least 3 times. This gets rid of any sequencing errors that may be in the barcodes. Call these potential barcodes.
- From the set of **all** reads, map variants to potential barcodes.
- For each barcode:
  - Assign the most common read as the reference. Calculate the Levenshtein distance between the reference and all other variants that map to this barcode.
  - If the max Levenshtein distance >= 11 (this threshold determined by bootstrapping, explained below), throw out this barcode.

Determining cutoff for Levenshtein distance
Randomly select two reference sequences (with replacement) and calculate the distance. Do this 100,000 times to form a reference Levenshtein distance distribution. Take the 1% percentile as the cutoff threshold.







Conclusion: For the control, firstAUG and firstUTR libraries the 1% percentile is 11, 72, and 72. Based on these bootstrap distributions, take cutoff = 11.