ProSynAR - PRObabilistic SYNthesis of Aligned Read-pairs

1| Quick Start

Windows Users: Download the binaries from

https://github.com/Benjamin-Crysup/prosynar/releases/download/1.0/prosynar.zip

If truly dead set on building from source, the relevant batch file can be run: MinGW64 will need to be installed and its binaries will need to be on the path.

*nix Users (and MacOS): Download the source from

https://github.com/Benjamin-Crysup/prosynar

To build, run the relevant shell script (Buildit_***processor***_nix.sh) for your system. The binary will be in the builds folder (for most of you, builds/bin_x64_linux). The zlib development package is needed.

Once the software is built, the following items are needed: a genetic reference, a description of problematic regions, a cost specification in those regions, and a SAM file with paired-end reads to attempt to merge. If starting with a BAM, use samtools view: obviously, samtools will need to be installed for this.

A minimal example is provided (with both source and binary distributions) and can serve as a template. A GUI is also provided (requires Python). For the most common use of this program, open up the command line / terminal and run (as one line):

```
samtools view -h /path/to/start.bam | prosynar --cost
CostFile.pdc --prob ProbFile.bed --ref Reference.fa --faildump
NotMerged.sam --out Merged.fq -- Frover --over 10 -- Fprover --
over 10 --lr 0.0 -- Fpprobreg --lr 0.0 -- Malign --cost
MergeCost.agc --over 10
```

2| Basic Command Line Arguments

```
prosynar

   --ref Reference.fa
   --cost PDCost.pdc
   --prob Regions.bed
   --thread NT
   --faildump NotMerged.sam
   --out Merged.fq
   FILE
   [-- FILTER]*
   -- MERGER
```

ProSynAR takes a set of reference sequences (--ref Reference.fa), position dependent cost information (--cost PDCost.pdc) and a description of difficult regions (--prob Regions.bed): while base ProSynAR does not use these, the various filters and merge algorithms might. Additionally, ProSynAR is multithreaded (and can be made to spin up as many threads as you desire: --thread NT).

ProSynAR takes in a SAM file: it can either come in from standard in or be on the filesystem. Ideally, the SAM file should be sorted by query name (so that pairs show up next to each other). While ProSynAR will attempt to handle any SAM file, unsorted files (or those sorted on coordinate), it will require more memory. ProSynAR has two outputs: the main output is a fastq containing all the merged sequences. Optionally, the unmerged sequences can be passed through to another SAM file.

After the options, multiple filters can be specified (separated by "--"). At the end, a single merge algorithm must be specified. In order to get more detail on any filter or merge, --help can be used on that item (i.e. prosynar -- Malign --help). Also, it is worth noting that, when appropriate, the examples in the help infomation show the default value: prosynar --help gives an example for specifying the number of threads (--thread 1), and the default number of threads is 1.

3| Cost Functions

The position dependent cost function follows the same format as ProDerAl (and, by default, the same file should be used for both). The description of the format from the ProDerAl manual follows.

Position dependent cost functions for ProDerAl are specified in chunks; each reference gets its own chunk. Each chunk starts with the reference name and ends with a pipe character (|). After the reference name is a default alignment parameter set (detailed later), followed by the number of specific parameter sets, followed by the specific parameter sets. All entries are separated by any amount of whitespace (ASCII 9, 10, 13, 32).

A default set of alignment parameters starts with the cost of opening a gap (should be negative), the cost of extending a gap, and the cost of closing a gap. After that is the number of relevant types of characters that can show up in the file (if working with amino acid sequences, set this to 21, for nucleic acids it should be 4… ish). The ASCII code points of those characters follows: for the most common cases,

```
          ACGT -> 4 65 67 71 84

          ACDEFGHIKLMNPQRSTUVWY -> 21 65 67 68 69 70 71 72 73 75
               76 77 78 80 81 82 83 84 85 86 87 89
```

Following the list of bases is a 2d array of costs for character matching and mismatching: for nucleic acids, this will have 16 entries, and for amino acids, there will be 441. As for the order, an example is in order: for the nucleic acids, the first entry is for both the reference and the read being an A, the second entry is for the reference being an A and the read being a C, etc… (row major order).

The specific parameter sets have the same information as the default, but are preceded by the base indices that are covered. The first two are the start (zero based) and end (exclusive) in the reference, and the last two are the start and end in the read. Any position that is negative does not act as a constraint: -1 15 5 -1 would be any base in the reference up to (but not including) base 15, and any base in the read at or after base 5.

As an example:

```
chr1
```

```
-3 -1 -3
  4 65 67 71 84
  1 -4 -4 -4 -4 1 -4 -4 -4 -4 1 -4 -4 -4 -4 1
1
10 21 -1 -1
  -1 0 -1
  4 65 67 71 84
  1 -4 -4 -4 -4 1 -4 -4 -4 -4 1 -4 -4 -4 -4 1
|
```

This will use the default bwa mem parameters for chromosome 1, except between bases [10,20], where the gap costs are lower.

4| Mergers

ProSynAR supports multiple different merge algorithms.

The default is Malign: this performs a semi-global alignment between the two reads and uses that for the merge. By default, it respects (i.e. throws away) soft clipping: the soft clipped bases can be restored using --unclip. Additionally, an optional filter on the overlap can be specified (--over RequiredOverlap). Finally, the costs to use in the alignment are specified in a separate file: this file contains a single cost specification, in the same format as the default costs in the position dependent cost file.

ProSynAR also supplies an implementation of the algorithm of FLASH (namely, the algorithm in the original paper, Magoč and Salzberg, 2011). Soft clipped bases can be restored with --unclip, and an overlap requirement can be specified with --over ReqOver. The original FLASH paper used an alternate method to get posterior quality scores: ProSynAR uses the method of Edgar and Flyvbjerg (2015) by default, but if the original method is desired, specify --flashqual. FLASH also supports a limit on the ratio of mismatches (--misrat 0.75) and requires the maximum expected number of bases of overlap (--maxexp).

ProSynAR also supplies an implementation of the algorithm of PEAR (Zhang et. al., 2014). Soft clipped bases can be restored with --unclip, and an overlap requirement can be specified with --over. The significance level can be set with --alpha (by default, 0.01). The match/mismatch scores can be specified with --ms and --xs (respectively). Again, as with FLASH, the default method to produce posterior qualities is that of Edgar and Flyvbjerg (2015): to use the exact method in PEAR, use --pearqual.

5| Filters

Filters can be inserted into the pipeline for each pair: if a pair fails any filter, it will not be merged.

The first filter, Frover, is a filter on the number of bases of overlap apparent in the alignment. The only option is the number of bases to require (--over 10).

The second filter is a probabilistic version, Fprover. This will perform a probabilistic test of overlap in the reference (LR of overlap meeting threshold vs not). Two important options are the number of bases to require (--over) and the required (log) likelihood ratio (--lr 0.0). If desired, an alternate cost can be specified (--cost). This runs through some number of alternate alignments: change the number of non-optimal scores to entertain with --rank and the number of alternate alignments to examine with --count. The realignments are between the read and the immediate vicinity of the original mapping: what counts as immediate is expanded with --overrun. Soft clipped bases can be reclaimed (and the region of the reference under consideration can be expanded) with --reclaimsoft. Additionally, to limit the number of alternate paths this filter explores, the maximum number of times to re-examine a score can be given with --hfuzz.

The third filter is the problematic region filter, Fpprobreg. Specific costs and problem regions can be specified (--cost and --prob), and like Fprover, the number of alternate alignments and non-optimal scores can be limited using --count and --rank. Also, soft clipped bases can be reclaimed with --reclaimsoft. This performs a probabilistic test on region assignment: the threshold for this likelihood ratio can be given with --lr.

6| Other Files

The reference file is a fasta file: each entry represents a single chromosome with the name given on the name line. On the name line, everything after a whitespace does not matter, and newlines can show up in the sequence itself. As a simple example

```
>chr1
ACGCTGACTACCAGGAA
AATGC
>chr2 extra stuft that doesn’t matter
AGTTTACAAACAGT
```

The problematic region file is a bed file. A bed file is a tab separated value file. ProSynAR only uses the first three entries: the chromosome name, the start position (zero based, inclusive) and the end position (zero based, exclusive). As an example, the following

```
chr1 3    8
chr1 10   13
chr2 5    9
```

specifies the following bases

chr1
ACG**CTGAC**TA**CCA**GGAAAATGC
chr2
AGTTT**ACAA**ACAGT

The starting SAM file is a collection of alignments that need to be re-evaluated. The SAM format, while essentially a tab separated value file, is too complex to describe here: see https://samtools.github.io/hts-specs/SAMv1.pdf for a complete description.

7| Citing and License

For attribution, when using or referring to this software in your own publications, please cite

TBD