misassembly_detection Documentation

Release 1.0

Rebecca R. Murphy

September 23, 2014

Contents

Contents:

CHAPTER 1

NxRepair: API

Contents:

class `nxrepair.`**`aligned_assembly`**(*bamfile*, *fastafile*, *min_size*, *threshold*, *step*, *window*, *minmapq*, *maxinsert*, *fraction*, *prior*)

Class to hold a set of mate pair or paired end reads aligned to the scaffolded genome assembly

`breakContigs_double`(*outfile*, *breakpoints*, *trim*)

Function to break a contigs at positions identified as assembly errors and write a new fasta file containing all contigs (both altered and unaltered).

Makes a two-point break at the identified misassembly position, splitting at 5 Kb upstream and downstream of the misassembly and (currently) excluding the misassembled region.

Arguments: outfile: name of the new fasta file (including filepath) breakpoints: dictionary of misassemblies. key = contig reference ID, value = list of misassembly positions within the contig trim: distance, in bases, to trim from each each edge of a breakpoint to remove misassembly (integer)

`get_anomalies`(*outfile*, *trim*, *img_name=None*)

Function to determine the frequency of anomalous mate pair behaviour across the entire genome assembly and return a dictionary where: key = contig reference IDs, value = list of postions within that contig where an assembly error is identified and the contig should be broken.

Calls get_size_anomalies and get_mapping_anomalies for each contig larger than the aligned_assembly.min_size; makes a .csv file listing for each contig the positions of identified misassemblies and their corresponding anomalous scores.

Arguments: outfile: name of file (including filepath) to store the list of contig misassemblies.

Keyword Arguments: img_name: name of file (including filepath, not including filetype) to store plots of alignment quality

`get_mapping_anomalies`()

Function to determine the frequency of strand mapping anomalies across the entire genome assembly.

Calls get_read_mappings for each contig larger than the aligned_assembly.min_size and returns: 1) a dictionary with keys = contig reference IDs; values = list of positions and strand alignment ratios as described in get_read_mappings 2) a dictionary of anomalies wiht keys = contig reference IDs, values = [list of positions for which the ratio of correctly aligned strands < 0.75 (currently hard-coded), corresponding ratio of correctly aligned strands]

`get_read_mappings`(*ref*)

Function to calculate the fraction of reads pairs within a contig that align correctly to opposite strands.

Return five arrays: the positions at which strand alignment was evaluated, the fraction correctly aligned, the fraction incorrectly aligned to the same strand, the unmapped fraction and the fraction that have some other alignment issue.

Arguments: ref: the reference id of the contig to be evaulated

`get_read_size_distribution`()

Function to calculate global insert size distribution across the whole assembly Return a frequency table of insert sizes as a dictionary with key = insert size, value = frequency

`get_reads`(*ref*, *start*, *end*)

Function to fetch reads aligned to a specific part of the assembled genome and return a list of aligned reads, where each list entry is a tuple: (read start position, read end position, read name, strand alignment) and strand alignment is a boolean indicating whether the two reads of a read pair align correctly to opposite strands. Reads are fetched that align to contig "ref" between positions "start" and "end".

Arguments: ref: the name of the contig from which aligned reads are to be fetched. start: the position on the contig from which to start fetching aligned reads end: the position on the contig from which to end fetching aligned reads

`get_size_anomalies`()

Function to determine the frequency of insert size anomalies across the entire genome assembly.

Calls probability_of_readlength for each contig larger than the aligned_assembly.min_size and returns: 1) a dictionary with keys = contig reference IDs; values = array of Zscores as described in probability_of_readlength 2) a dictionary of anomalies wiht keys = contig reference IDs, values = [list of positions for which abs(z-score) > 2 (currently hard-coded), corresponding z-score value]

`make_tree`(*ref*)

Function to construct an interval tree from reads aligning to a contig and return the interval tree.

The interval tree stores nodes with properties start (start postition of interval), end (end position of interval) and other, which is a tuple of the mate pair name (string) and the strand alignment of the two paired reads (boolean).

Arguments: ref: Reference ID of the contig for which the interval tree is to be constructed

CHAPTER 2

Tutorial: Misassembly Detection using NxRepair

NxRepair is a python module that automatically detects large structural errors in de novo assemblies using Nextera mate pair reads. The decector will break a contig at the site of an identified misassembly and will generate a new fasta file containing both the corrected contigs and the correct, unaffected contigs.

2.1 Installing NxRepair

NxRepair program can be cloned from github:

```
git clone https://github.com/rebeccaroisin/nxrepair
```

NxRepair uses several other python libraries, which you will need to install. Specifically, you will need:

- scipy
- numpy
- matplotlib
- pysam

Scipy, numpy and matplotlib can be installed using your favourite package manager or from the Python Package Index (PyPI) using:

```
pip install numpy
pip install scipy
pip install matplotlib
```

If you don't currently have any of these libraries, we recommend installing Anaconda, a python distribution that includes all of these libraries, along with all major scientific and analytical python packages.

Pysam can also be installed using "pip install", but errors are more common.

Assistance with installation errors for pysam can be found online. Note that the current version of pysam wraps samtools-0.1.19 and tabix-0.2.6.

2.2 Running NxRepair

NxRepair can be run from the command line.

```
python nxrepair.py aligned_matepairs.bam assemblyfasta.fasta error_locations.csv new_fasta.fasta
```

The required arguments are as follows:

- aligned_matepairs.bam: an indexed bam file of mate pair reads aligned to your assembly;
- assemblyfasta.fasta: the fasta file containing your contigs;
- outfile: the name of a csv file in which to store the Z scores generated by NxRepair;
- newfasta: filename of the fasta file that will hold the new contigs following analysis.

There are also several optional arguments, which can be used to tune the NxRepair analysis. These are described in the table below.

Parameter	Default Value	Meaning
imgname	None	Prefix under which to save plots.
maxinsert	30000	Maximum insert size, below which a read pair is included in calculating population statistics.
minmapq	40	Minimum MapQ value, above which a read pair is included in calculating population statistics.
minsize	10000	Minimum contig size to analyse.
prior	0.01	Prior probablility that the insert size is anomalous.
stepsize	1000	Step-size in bases to traverse contigs.
trim	5000	Number of bases to trim from each side of an identified misassembly.
T	-4.0	Threshold in Z score (standard deviations from the mean) below which a misassembly is called.
window	200	Window size across which bridging mate pairs are evaluated.

Optional arguments are called from the commmand line, as shown in the example below:

```
python nxrepair.py aligned_matepairs.bam assemblyfasta.fasta error_locations.csv new_fasta.fasta -mi
```

The program will parse a bam file of reads aligned to your de novo assembly. Each contig that is larger than the min_size parameter will be analysed for potential structural misassemblies. When the program completes, a minimum of two new files will be generated:

1. A new fastafile, specified by newfasta, that contains the improved contigs of the de novo assembly.
2. A csv file that identifies the exact position where altered contigs were broken.
3. If the optional argument -img_name was included, for each contig analysed, a plot will be generated showing the insert size distribution and directionality across the contig, with anomalous regions highlighted. These plots will be saved in the folder specified by img_name

Outputs 2 and 3 can allow identification of further, smaller structural misassemblies, as well as enabling verification of detected misassemblies using IGV.

2.3 How Does it Work?

NxRepair evaluates the insert sizes of mate pairs aligned across a contig. Regions of the contig that have unusual insert sizes, where few reads are aligned, or where a large fraction of the mate pairs have incorrect orientation are flagged as potentially anomalous based on a simple probabilistic model of the mate-pair size distribution. Where there is strong evidence that a region is misassembled, the contig will be broken into two pieces and 5 Kb of erroneous assemby will be trimmed from both sides of the break.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

Python Module Index