bambus3 Documentation

Release 1.0

Jay Ghurye

Jul 07, 2021

Contents

MetaCarvel is an improved version of a metagenomic scaffolder Bambus2. MetaCarvel works with standard file formats and easier to use compared to Bambus2. We have made significant changes in the repeat detection module thus making it order of magnitude faster than Bambus2.

CHAPTER 1

Documentation

1.1 MetaCarvel Pipeline

MetaCarvel is a scaffolding program designed specifically to address the issues concerning the metagenomic data. The scaffolding algorithm consists of several steps. MetaCarvel requires contig assembly fasta file, mapping of reads to contigs in BAM file and output directory as an input. It first analyzes the read to contig mapping and estimates the library insert size. Using these mappings, it generates a scaffold graph where nodes are contigs and edges are mate pairs mapped to two different contigs. This is done in `libcorrect` program. Multiple links between two contigs are collapsed into a single link using `bundler` program. After this, repeats in the graph are removed. High betweenness centrality nodes are identified using `centrality` program and removed from the graph using `repeat_filter.py` program. Contigs remaining after filtering repetitive contigs are assigned a unique orientation using `orientcontigs` program. The bubbles caused due to polymorphic regions in the metagenome are identified using `spqr` program. Once these bubbles are identified, they are collapsed and contig layout is performed in `layout.py` program.

1.2 Installing the software

MetaCarvel is open source and available on github. You need to run following steps to get it installed

```
git@github.com:marbl/MetaCarvel.git
cd MetaCarvel
make
```

This would generate the executables required to run MetaCarvel.

1.3 Preparing the data

In order to run MetaCarvel, you will need two files. First one is the contig assembly of the raw reads using your favorite metagenome assembly program. The second one is the alignment of raw paired end reads to assembled contigs using your favorite alignment program. We strongly recommend to map paired end reads as single end reads since aligner enforces the reads to map in the distances given by the library size. If you are using Bowtie2 for read mapping, we recommend aligning reads this way:

```
bowtie2 -x idx -U forward_reads.fq | samtools view -bS - > alignment_1.bam
bowtie2 -x idx -U reverse_reads.fq | samtools view -bS - > alignment_2.bam
samtools alignment_total.bam alignment_1.bam alignment_2.bam
samtools sort -n alignment_total.bam -o alignment_total_sorted.bam
```

We need this file sorted by the read name and not the coordinates. You will also need to provide the output directory where you want your scaffolds to be written.

1.4 Running MetaCarvel

Before you run MetaCarvel, please make sure you have following dependencies: samtools, bedtools and NetworkX. After you have these dependencies, running MetaCarvel is pretty straightforward. You would need to execute `run_new.py` file.

```
python run_new.py -h
usage: run_new.py [-h] -a ASSEMBLY -m MAPPING -d DIR [-f FORCE] [-r REPEATS]
              [-k KEEP] [-l LENGTH] [-b BSIZE]

MetaCarvel: A scaffolding tool for metagenomic assemblies

optional arguments:
  -h, --help            show this help message and exit
  -a ASSEMBLY, --assembly ASSEMBLY
                        assembled contigs
  -m MAPPING, --mapping MAPPING
                        mapping of read to contigs in bam format
  -d DIR, --dir DIR     output directory for results
  -f FORCE, --force FORCE
                        force re-run of pipeline, will remove any existing
                        output
  -r REPEATS, --repeats REPEATS
                        To turn repeat detection on
  -k KEEP, --keep KEEP  Set this to kepp temporary files in output directory
  -l LENGTH, --length LENGTH
                        Minimum length of contigs to consider for scaffolding
  -b BSIZE, --bsize BSIZE
                        Minium mate pair support between contigs to consider
                        for scaffolding
```

The mandatory inputs for this are the FASTA file for contig assembly, the alignment file and output directory. There are other options as well. `-r` option lets you toggle the repeat detection process in the scaffolding pipeline. By default it is turned off, but you can provide `-r true` parameter to include repeat detection. :code: *-l* option lets you set the minimum length of the contigs to consider for scaffolding. By default this is set to :code: 500. `-b` option lets you set the minimum mate pair support between two contigs to consider for scaffolding. By default this is set to `3`. Each step in the pipeline generates temporary files as intermediate outputs. If you want to retain these files, you need to set `-k` option to `true`. By default, these files will be removed.

1.5 Interpreting the output

Since MetaCarvel consists of multiple steps, it generates some intermediate files. These files can be useful if you want to dig more into the analysis. The `libcorrect` generates two output files: `contig_links`, having a raw links between the contigs based on mate pairs and `contig_coverage`, having a coverage information for each contig. The `bundler` program uses `contig_links` file, bundles the links and outputs them in the `bundled_links` file in a tsv format and in the `bundled_graph.gml` in the gml format. If repeat detection is chosen then that would generate a file with name `bundled_links_filtered` which will have the bundled links corresponding to all the non-repetitive contigs. The `orientcontigs` program takes the bundled links as an input and outputs a graph with only one orientation corresponding to each contig. This graph is written in two formats: `oriented_links` in tsv format and `oriented.gml` in a gml format. The `spqr` program takes orientated graph as an input and produces a potential bubbles as an output in a file called `seppairs`. The `layout.py` produces three main files. One is `scaffolds.fa`, represnting the scaffold sequences, `scaffolds.agp`, representing scaffolds in an AGP format and `scaffolds.gfa` representing scaffold graph in GFA format. If you want to visualize the graphs used to generate scaffolds, you can either use either *oriented.gml* or *scaffold_graph.gfa* file and load it into any standard graph visualization software. We recommend using MetagenomeScope as you can visualize variants and scaffold paths simply by loading the agp output into the viewer.