BugBuilder - An Automated Microbial Genome Assembly and Analysis Pipeline

Abbott, J.C.

Bioinformatics Data Science Group, Department of Surgery and Cancer, Imperial College London, London UK, SW7 2AZ

Abstract

Summary: BugBuilder is a framework for hands-free assembly and annotation of microbial genomes. It produces outputs suitable either for database submission or downstream finishing processes. It is configurable to work with most command-line assembly and scaffolding tools which are selectable at run-time, and supports all common sequence types used in microbial genome assembly.

Availability and Implementation BugBuilder is implemented in Perl and is available under the Artistic License from [http://www.imperial.ac.uk/bioinformatics-data-science-group/resources/software/bugbuilder](http://www.imperial.ac.uk/bioinformatics-data-science-group/resources/software/bugbuilder). A virtual machine image is available pre-configured with the relevant freely-redistributable dependencies.

Contact: j.abbott@imperial.ac.uk

1 Introduction

The rapid increase in sequencing throughput in recent years has moved the bottleneck in the process of sequencing microbial genomes from sequence generation to the analysis. The cost of sequence analysis now typically outweigh those of sequencing, while the use of multiplexing combined with the scale of modern sequencing instruments typically results in large batches of sequences being produced simultaneously. Lab-based researchers often do not have the background knowledge necessary to carry out the required analysis themselves.

BugBuilder provides a portable framework for carrying out assembly and analysis of microbial genome sequences, taking sequence reads as inputs and producing submission-ready annotated genome assemblies, with ease-of-use prioritised to allow non-expert users to obtain acceptable results. The software selects appropriate tools and parameters based upon the type of sequence data provided. It is also suitable for more advanced applications, being designed to be readily deployed in a cluster environment while permitting manual run-time selection of tools and parameters, and can be readily customised to include any Linux command-line based assembly and scaffolding tools. The outputs of most major sequencing platforms are supported, including long-read and hybrid assemblies, and can either provide submission-ready outputs, or augment these with indications of potential sites of misassembly to assist with downstream finishing work.

2 Implementation

BugBuilder is implemented in Perl, utilising a YAML format configuration file. A BugBuilder job carries out its work in a temporary directory, within which a separate directory is created for each component of the pipeline. A series of symbolic links are maintained within the working directory which link to, for example, the most recent version of generated contig sequences, facilitating flexible workflows through consistent file naming.

In order to prevent the system being tied to particular software tools or sequencing platforms, multiple assemblers and scaffolders are defined with the central configuration. Any package which can be run on the command-line can be integrated into BugBuilder, either directly where the tools input and output requirements are appropriate for BugBuilder, or through the use of a script wrapper where more complex requirements exist. Default arguments to the tools are also defined in the configuration, which can be overridden at run time through command-line arguments.
3 Workflow

The BugBuilder workflow is shown in Figure 1. The required inputs are fastq files obtained from a sequencing experiment, where a single fastq file is required from a fragment library, or a pair of non-interleaved fastq files in the case of a mate-pair library. Sequence from long-read platforms (i.e. PacBio, MinION) can either be used for standalone assembly or combined with short-read sequences for a hybrid assembly. A fasta-formatted reference genome sequence can optionally be provided which may be used for scaffolding and ordering scaffolds if desired. Assemblers and scaffolders are categorised according to applicable sequencing platforms, so the user can either define the sequencing platform used and allow BugBuilder to select the most appropriate tools based on the sequence type, or manually select the desired programs from the applicable tools defined for the sequence category.

3.1 Sequence Assessment and Preprocessing

Prior to assembly, a number of preprocessing stages are carried out. The processes executed vary according to sequence type, and individual stages can be skipped if desired by the user. Sequence reads are inspected to determine the sequence characteristics, sequence quality is assessed using FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/), followed by sequence trimming using Sickle (https://github.com/najoshi/sickle) to remove low quality bases for assemblers which do not carry out trimming directly. High coverages in de Bruijn genome assemblers can lead to a decrease in assembly quality, consequently reads are downsampled to either 100x or a user-defined threshold for such assemblers.

3.2 Contig Assembly

The default BugBuilder configuration includes the ABySS (Simpson et al., 2009), SPAdes (Bankevich et al., 2012) and Celera WGS (CABOG and PBcR) (Miller et al., 2008, Berlin et al., 2015) assemblers, providing support for sequence obtained from Illumina short-read platforms (i.e. GAII, HiSeq, MiSeq), through to Roche 454, PacBio and MinION instruments. The assembly stage produces, at minimum, a fasta file of contig sequences, and optionally a fasta file of scaffolded sequences where scaffolding is supported directly by the assembler.

3.3 Scaffolding

BugBuilder considers two classes of scaffold; those which utilise read-pair associations between contigs, and those which carry out alignment to a reference sequence. Reference-free methods are less susceptible
to biasing the outputs to the organisation of the reference sequence, however typically produce considerably
less contiguous results, consequently the preferred algorithm is highly dependent upon similarity to the reference
genome and user preference. Predefined configurations for scaffolding with SSPACE [Boetzer et al.,
2011], SIS [Dias et al., 2012] and the Mauve Contig Mover [Rissman et al., 2009] are provided with the software.

Scaffolds can frequently be further improved by ordering and orientating them against the reference
sequence, especially if a reference-free scaffolding algorithm was used. The origin of replication
will typically occur within a contig sequence, although by convention this is used as the start of the genome
sequence. The location of the origin is determined by alignment against the reference (if provided), and
the assembly orientated around this locus. Remaining gaps between contigs can then be processed with
GapFiller [Boetzer and Piravano, 2012], which carries out incremental alignment of sequence reads around
contig ends to extend contigs and close gaps.

### 3.4 Annotation, Visualisation and Outputs

Annotation of assembled sequences is carried out using Prokka [Seemann, 2014], which combines various
feature predictions algorithms and similarity search methods to produce an annotation including coding
genes, rRNAs and tRNAs. BugBuilder also allows execution of a downstream validation process (amosvalidate
- Phillippy et al., 2008) to help identify potential misassemblies by evaluating read pair distribution to identify
compressed or extended read-pairs and inversions etc., and read depth to identify collapsed or expanded
repeats.

The main output of the pipeline is an EMBL-format record containing the annotated scaffold or contig se-
quences which is appropriate for submission to the ENA. Visualisation and interpretation of results are aided
by the production of a graphical genome map using Circleator (Crabtree et al., 2014), along with similarity
plots and percentage-identity plots generated with MUMmerplot [Kurtz et LA., 2004]. A full list of outputs
generated is provided in table 1.

### 3.5 Example Assemblies

Sequence data for E. coli K-12 strains generated from a range of platforms was obtained from the ENA
database and used to validate the pipeline outputs (see table 2). Each was assembled using the default as-
sembler, both with and without reference-based scaffolding using SIS, with just the platform, reference
sequence (U00096.3) and choice of scaffold (where required) being provided as command-line arguments.
Assemblies were run using 8 threads on AMD Opteron 6282 SE 2.6Ghz CPUs, with 8Gb RAM available per-
core.

BugBuilder has to date been successfully used on projects involving a range of species including Strep-
tococcus pyogenes (e.g. Turner et al., 2015), Escherichia coli, Gluconacetobacter hansenii (e.g. Florea et al.,
2016) and Klebsiella pneumoniae.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Type</th>
<th>When produced</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read[12]_fastqc.html</td>
<td>HTML</td>
<td>Always</td>
<td>Output of FastQC quality assessment</td>
</tr>
<tr>
<td>contigs.embl/gb</td>
<td>EMBL/GenBank</td>
<td>Unless scaffolded</td>
<td>Annotated contig sequence in submission-ready format.</td>
</tr>
<tr>
<td>contigs.fasta</td>
<td>Fasta</td>
<td>Always</td>
<td>Contig sequences in fasta format</td>
</tr>
<tr>
<td>unplaced_contigs.fasta</td>
<td>Fasta</td>
<td>If scaffolded</td>
<td>Contig sequences which could not be incorporated during reference-based scaffolding</td>
</tr>
<tr>
<td>scaffolds.embl/gb</td>
<td>EMBL/GenBank</td>
<td>If scaffolded</td>
<td>Annotated scaffold sequences in submission-ready format.</td>
</tr>
<tr>
<td>scaffolds.fasta</td>
<td>Fasta</td>
<td>If scaffolded</td>
<td>Scaffold sequences in fasta format</td>
</tr>
<tr>
<td>scaffolds.agp</td>
<td>AGP 2.0</td>
<td>Details of scaffold construction</td>
<td></td>
</tr>
<tr>
<td>comparison_vs_[accession].png</td>
<td>PNG image</td>
<td>With reference</td>
<td>MUMmerplot output of nucleotide alignment between scaffolds and reference, indicating structural differences between reference and sample</td>
</tr>
<tr>
<td>comparison_vs_[accession]_pip.png</td>
<td>PNG image</td>
<td>With reference</td>
<td>MUMmerplot percentage identity plot, indicating sequence similarity between reference and sample</td>
</tr>
<tr>
<td>circleator.png</td>
<td>PNG image</td>
<td>Always</td>
<td>Circleator genome map</td>
</tr>
<tr>
<td>circleator.svg</td>
<td>SVG image</td>
<td>Always</td>
<td>Circleator genome map</td>
</tr>
</tbody>
</table>

Table 1: Output files produced by BugBuilder. Outputs vary according to executed tools as indicated in ‘When produced’ column as follows: ‘Always’ - output is always produced; ‘If scaffolded’ - produced if the assembly is scaffolded, either by the assembler or a scaffold being run; ‘Unless scaffolded’ - produced if the assembly is not scaffolded; ‘With reference’ - output only produced if a reference sequence is provided.
4 ASSEMBLY RESULTS

<table>
<thead>
<tr>
<th>Platform</th>
<th>Organism</th>
<th>ENA Accession</th>
<th>Read Type</th>
<th>Insert Size</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illumina GA</td>
<td>E. coli K-12 substr. MG1655</td>
<td>PRJNA30551</td>
<td>37 bp paired</td>
<td>486 bp</td>
<td>112x</td>
</tr>
<tr>
<td>Illumina HiSeq 2000</td>
<td>E. coli K-12 substr. MG1655</td>
<td>PRJEB4687</td>
<td>102 bp paired</td>
<td>2408 bp</td>
<td>810x</td>
</tr>
<tr>
<td>Illumina MiSeq</td>
<td>E. coli K-12</td>
<td>PRJEB8559</td>
<td>128 bp paired</td>
<td>4914 bp</td>
<td>49x</td>
</tr>
<tr>
<td>Roche 454 FLX</td>
<td>E. coli K-12 substr. MG1655</td>
<td>PRJEB40875</td>
<td>Unpaired, mean 228 bp</td>
<td>-</td>
<td>29x</td>
</tr>
<tr>
<td>PacBio RSII</td>
<td>E. coli K-12 substr. MG1655</td>
<td>PRJEB237120</td>
<td>Unpaired long reads, mean 1954 +/- 2621 bp</td>
<td>-</td>
<td>-139x</td>
</tr>
</tbody>
</table>

Table 2: E. coli K-12 libraries used for assemblies.

Table 3: E. coli K-12 assembly results.

4 Assembly Results

The outputs of the test assemblies are shown in table 3 and figures 2-5. As would be expected the main factor determining the contiguity of assembly is the sequencing technology employed and the subsequent choice of assembly and scaffolding algorithm. The configurable nature of the pipeline readily permits the addition of new algorithms with improved capabilities or technology support.
Figure 2: Total assembly sizes for E. coli K12 MG1655 example assemblies.

Figure 3: Number of CDS predictions from E. coli K12 MG1655 example assemblies. CDS predictions were carried out on the scaffolding sequences resulting from each assembly using Prokka.

Figure 4: Numbers of contig and scaffold sequences resulting from E. coli K12 MG1655 example assemblies. Assembler and scaffolder selections used for each assembly are indicated in table 3. Statistics for scaffold sequences are either based upon scaffolds output by the assembler, or for the outputs of a downstream scaffolding process in assemblies labelled 'Scaffolded'.

Figure 5: Contig and scaffold N50 values resulting from E. coli K12 MG1655 example assemblies. Assembler and scaffolder selections used for each assembly are indicated in table 3. Statistics for scaffold sequences are either based upon scaffolds output by the assembler, or for the outputs of a downstream scaffolding process in assemblies labelled 'Scaffolded'.
REFERENCES

References


