User Manual

Date: 10 July 2012
Software Version: 1.10
Developed by: Marten Boetzer and Walter Pirovano

GapFiller v1.10 Marten Boetzer - Walter Pirovano, July 2012
email: walter.pirovano@baseclear.com

Citation

If you use GapFiller in a scientific publication, please cite:
Boetzer, M. and Pirovano, W., Towards (almost) closed genomes with GapFiller, Genome Biology, 13(6), 2012

License

What is GapFiller?

GapFiller is a program to close gaps within previously created scaffolds. Gaps within scaffolds (such as created with SSPACE) are defined as unknown nucleotides (N's). With GapFiller, the unknown nucleotides are filled with true nucleotides in order to (try) to close the gap.

Why GapFiller?

GapFiller can be used to close gaps and solve repeated elements or low-covered regions (that could previously not be assembled).

How to use GapFiller?

GapFiller comes with a number of files.

GapFiller_v1-10.pl
- Main program. Perl script for closing gaps.

README
- README file. Information about the process, input files/parameter options, and output files.

MANUAL
- This file.

TUTORIAL
- Small tutorial on an E.coli dataset.

Bowtie folder
- bowtie scripts for mapping the reads to the scaffolds

BWA folder
- BWA scripts for mapping the reads to the scaffolds

Example folder
- Example scaffolds, read TUTORIAL file.

To run the main script, type;

perl GapFiller.pl
Or
./GapFiller.pl

This will print the options and parameters to the screen. Below is each parameter explained in detail.

The '-l' library file:

The library file contains information about each library. The library file contains six columns, each separated by a space. An example of a library file is;

Lib1 bwa file1.1.fasta file1.2.fasta 400 0.25 FR
Lib1 bowtie file2.1.fasta file2.2.fasta 400 0.25 FR
Lib2 bowtie file3.1.fastq file3.2.fastq 4000 0.5 RF

Each column is explained in more detail below;

Column 1:

Name of the library. A short name to keep track of the names of the libraries. All temporary files and summary statistics are named after this

library name. Libraries that have the same name are considered to also have the same distance and deviation (column 4 and 5).

Column 2:

Name of the aligner, either bowtie, bwa or bwasw. Use bowtie for small reads (<50bp) and for fast analysis. Use bwa for longer reads (>50 and <150) and use bwa for very large reads (e.g. 454). BWA and BWA-sw are run in default mode.

Column 3 & 4:

Fasta or fastq files for both ends. For each paired read, one of the reads should be in the first file, and the other one in the second file. The paired reads are required to be on the same line. No naming convention of the reads is required, because names of the headers are not used in the protocol. Thus names of the headers shouldn't be the same and do not require any overlap of names like (…).x and (…).y, as is commonly used in assembly programs.
In conclusion, each read should be larger than 16 (or the '–m' parameter if -x 1). If they are shorter, the program will simply omit them from the whole process.

Column 5 & 6:

The fourth column represents the expected/observed inserted size between paired reads. The fifth column represents the minimum allowed error. A combination of both means e.g. that with an expected insert size of 4000 and 0.25 error, the distance can have an error of 4000 * 0.25 = 1000 in either direction. Thus pairs between 3000 and 5000 distance are valid pairs.

Column 7:

The final column indicates the orientation of the paired-reads. Orientations can be: FF, FR, RF or RR. Where the F stands for --> orientation, and R for <-- orientation. Orientation of FR thus means that the pairs are: --><--

MAIN PARAMETERS:

The ‘-s’ scaffolds fasta file

The ‘–s’ scaffolds file should be in a fastA format.

GapFiller PARAMETERS:

The ‘–m’ minimum overlap

Minimum number of overlapping bases of the reads with the gap in the scaffold. Higher ‘-m’ values lead to more accurate gapclosing at the cost of decreased coverage. We suggest to take a value close to the largest read length. For example, for a library with 36bp reads, we suggest to use a -m value between 30 and 35 for reliable gapclosing.

The -o number of reads

Minimum number of reads needed to call a base during gapclosing, also known as base coverage. The higher the ‘-o’, the more reads are considered for gapclosing, increasing the reliability of the extension.

The ‘-r’ minimal base ratio

Minimum base ratio used to accept a overhang consensus base. Higher '-r' value lead to more accurate gapclosing

The ‘-n’ sequence overlap

Minimum overlap required to merge two sequences surrounding the gap. Overlaps in the final output are shown in lower-case characters.

The ‘-t’ trim sequence

Number of nucleotides to be trimmed of the sequence edges of the gap. Example for -t 5;

Sequence:
AGATAGATAGTCGTAGATAGATAGATAGCANNNNNNNNNNNNNNGATATATATGGCTCATGCTGATCAA
Trimmed :
AGATAGATAGTCGTAGATAGATAGAnnnnnNNNNNNNNNNNNNNnnnnnATATGGCTCATGCTGATCAA

From the edges of the sequences surrounding the gap, 5bp are trimmed off, as can be seen by the lower-case 'n'. Usually the edges of the sequences are low-quality/misassembled sequences, which can cause the GapFiller to not proper extend the sequences or not close the gap because no overlap can be found.

The ‘-d’ gapclose difference

Window that specifies the difference between the gapclosed length and the original gapsize. If the length of the gapclosed sequence deviates too much from the original gapsize, gapclosing is either stopped (if > difference) or sequences are not merged (< difference).

BOWTIE MAPPING PARAMETERS:

The '-g' maximum gaps

Maximum allowed gaps for Bowtie. This option corresponds to the -v option in Bowtie. The more gaps allowed, the slower the alignment of the reads.

The '-T' number of threads

Number of search threads for reading in the files and mapping of the reads.

ADDITIONAL PARAMETERS:

The ‘-b’ prefix base name

All files start with the ‘-b’ prefix to allow for multiple runs on the same folder without overwriting the results.

The ‘-i’ number of iterations

Number of iterations to fill the gaps. It re-uses the initial reads and maps them against the remaining gaps. If no more reads are closed, compared with the previous number of gaps, the process is stopped. Otherwise, it will keep on closing until the specified number of iterations are finished.

Additional information about the input, output and general process of the script can be found in the README file.