Next-Generation Sequence Alignment

A. Black P.

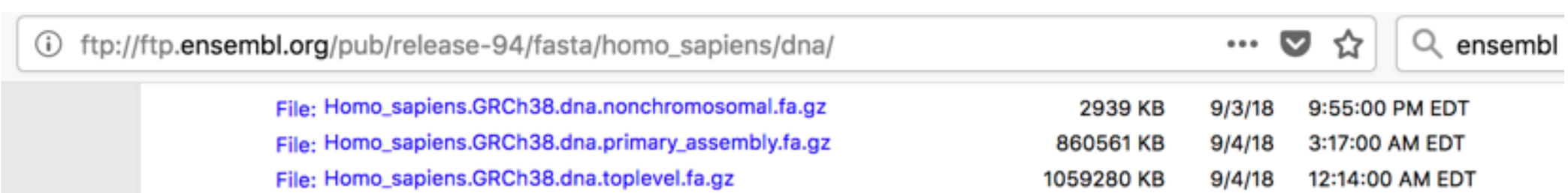
2018.10.18

Read Alignment Procedure

- Load reference genome on HPCC
- Index genome with alignment software (BWA)
- Aligned QC’ed reads to genome (BWA)
- Convert SAM to BAM (samtools)

Get a reference if possible

- Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz (all assembled chromosomes)
- Homo_sapiens.GRCh38.dna.toplevel.fa.gz (all assembled data)



BWA (original paper)

Bioinformatics. 2009 Jul 15;25(14):1754-60. doi: 10.1093/bioinformatics/btp324. Epub 2009 May 18.

Fast and accurate short read alignment with Burrows-Wheeler transform.

Li H[1], Durbin R.

Author information

Abstract

MOTIVATION: The enormous amount of short reads generated by the new DNA sequencing technologies call for the development of fast and accurate read alignment programs. A first generation of hash table-based methods has been developed, including MAQ, which is accurate, feature rich and fast enough to align short reads from a single individual. However, MAQ does not support gapped alignment for single-end reads, which makes it unsuitable for alignment of longer reads where indels may occur frequently. The speed of MAQ is also a concern when the alignment is scaled up to the resequencing of hundreds of individuals.

RESULTS: We implemented Burrows-Wheeler Alignment tool (BWA), a new read alignment package that is based on backward search with Burrows-Wheeler Transform (BWT), to efficiently align short sequencing reads against a large reference sequence such as the human genome, allowing mismatches and gaps. BWA supports both base space reads, e.g. from Illumina sequencing machines, and color space reads from AB SOLiD machines. Evaluations on both simulated and real data suggest that BWA is approximately 10-20x faster than MAQ, while achieving similar accuracy. In addition, BWA outputs alignment in the new standard SAM (Sequence Alignment/Map) format. Variant calling and other downstream analyses after the alignment can be achieved with the open source SAMtools software package.

AVAILABILITY: http://maq.sourceforge.net.

Where to get BWA

GitHub, Inc. (US) | https://github.com/lh3/bwa

bwa durbin

Features Business Explore Marketplace Pricing Search Sign in or Sign up

lh3 / bwa

Watch 116 Star 599 Fork 319

Code Issues 84 Pull requests 21 Projects 0 Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Burrow-Wheeler Aligner for short-read alignment (see minimap2 for long-read alignment)

bioinformatics sequence-alignment genomics fm-index

923 commits 11 branches 32 releases 22 contributors GPL-3.0

Branch: master New pull request Find file Clone or download

lh3 Merge remote-tracking branch 'origin/master' Latest commit f090f93 on Apr 10

bwakit	Exclude empty SAM lines in bounds check for ALT postprocessor	6 months ago
.gitignore	Added Makefile.bak and bwamem-lite to .gitignore	6 years ago
.travis.yml	removed coverity	4 years ago

BWA Manual

bio-bwa.sourceforge.net/bwa.shtml

bwa durbin

Manual Reference Pages - bwa (1)

NAME

bwa - Burrows-Wheeler Alignment Tool

CONTENTS

SYNOPSIS

```
bwa index ref.fa

bwa mem ref.fa reads.fq > aln-se.sam

bwa mem ref.fa read1.fq read2.fq > aln-pe.sam

bwa aln ref.fa short_read.fq > aln_sa.sai

bwa samse ref.fa aln_sa.sai short_read.fq > aln-se.sam

bwa sampe ref.fa aln_sa1.sai aln_sa2.sai read1.fq read2.fq > aln-pe.sam

bwa bwasw ref.fa long_read.fq > aln.sam
```

DESCRIPTION

BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the

BWA Submission Script

```
[ablackpz@dev-intel18 Session4]$ more runBwaIndex_slurm_giab_primary.sb
#!/bin/bash --login
########## Define Resources Needed with SBATCH Lines ##########

#SBATCH --time=03:00:00             # limit of wall clock time - how long the job will run (same as -t
)
#SBATCH --ntasks=1                  # number of tasks - how many tasks (nodes) that you require (same
as -n)
#SBATCH --cpus-per-task=1           # number of CPUs (or cores) per task (same as -c)
#SBATCH --mem=10G                    # memory required per node - amount of memory (in bytes)
#SBATCH --job-name Name_of_Job      # you can give your job a name for easier identification (same as
-J)


########## Command Lines to Run ##########


module load GCC/6.4.0-2.28
module load OpenMPI/2.1.1
module load BWA/0.7.17                    ### load necessary modules, e.g.

cd /mnt/scratch/ablackpz/CMSE890304/Session4/ ### change to the directory where your data is located

srun -n 1 bwa index -a bwtsw Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz            ### call your
executable

scontrol show job $SLURM_JOB_ID     ### write job information to output file
```

BWA Index Files

```
[ablackpz@dev-intel18 Session4]$ ls Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz*
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz.amb
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz.ann
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz.bwt
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz.pac
Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz.sa
```

.amb = text file with locations of ambiguous nucleotides (N's) in reference
.ann = text file with name and length of reference sequences
.bwt = BWT of reference sequence
.pac = packaged sequence (four nucleotides are encoded as one byte)
.sa = suffix array index

BWA Alignment Script

```
[ablackpz@dev-intel18 Session4]$ more runBwaAln_slurm_giab.sb
#!/bin/bash --login
########## Define Resources Needed with SBATCH Lines ##########

#SBATCH --time=00:10:00             # limit of wall clock time - how long the job will run (same as -t
)
#SBATCH --ntasks=8                  # number of tasks - how many tasks (nodes) that you require (same
as -n)
#SBATCH --cpus-per-task=1           # number of CPUs (or cores) per task (same as -c)
#SBATCH --mem=40G                    # memory required per node - amount of memory (in bytes)
#SBATCH --job-name Name_of_Job      # you can give your job a name for easier identification (same as
-J)


########## Command Lines to Run ##########

module load GCC/6.4.0-2.28
module load OpenMPI/2.1.2
module load BWA/0.7.17                    ### load necessary modules, e.g.
module load SAMtools/1.9

cd /mnt/scratch/ablackpz/CMSE890304/Session4/ ### change to the directory where your data is located

srun -n 8 bwa mem -t 8 -R @RG\tID:rg1\tSM:NA12878\tPL:illumina\tLB:lib1\tPU:H7AP8ADXX:1:TAAGGCGA Homo_
sapiens.GRCh38.dna.primary_assembly.fa.gz ../Session3/NIST7035_TAAGGCGA_L001_R1_paired.fastq.gz ../Ses
sion3/NIST7035_TAAGGCGA_L001_R2_paired.fastq.gz > NIST7035_TAAGGCGA_L001_R_paired_aln.sam
### call your executable

samtools view -Sb NIST7035_TAAGGCGA_L001_R_paired_aln.sam > NIST7035_TAAGGCGA_L001_R_paired_aln.bam
scontrol show job $SLURM_JOB_ID     ### write job information to output file
```

Really old SAM file

```
[ablackpz@dev-intel18 DataSet159]$ tail DataSet160E1R50G100_aln.sam
ENSGALT00000018587	0	ENSGALT00000039283	330	37	100M	*	0	0	CCCCGACGGCCGAGCGGGCCGCTCCGCGGGGCGGGCGGCCGTCGGGGAAAGCG
GAGGCGGTAGCGGCTATTCCTCCGCCTCGCCCTGTTCCTCCGAGGAG	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+330,100
M,0;
ENSGALT00000018587	0	ENSGALT00000039283	41	37	100M	*	0	0	CACGCTGCCGCCGCCTCCCCGCCTCTCCGGAGCTGCTGCGCTGCAAGCGCCGC
TTGGCCTTCGCTTCCCTCTCGGCGGCGGCACCGCGGCGGCGGCCGT	XT:A:U	NM:i:1	X0:i:1	X1:i:0	XN:i:1	XO:i:0	XG:i:0	MD:Z:64C35	XA:Z:ENSGALT00000039283,+41,100M
,1;
ENSGALT00000018587	0	ENSGALT00000039283	299	37	100M	*	0	0	TCCTCGACGAACACGACGCCGCCGCCGCGTTCCCCGACGGCCGAGCGGGCCGC
TCCGCGGGGCGGGCGGCCGTCGGGGAAAGCGGAGGCGGTAGCGGCTA	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+299,100
M,0;
ENSGALT00000018587	0	ENSGALT00000039283	295	37	100M	*	0	0	AGGCTCCTCGACGAACACGACGCCGCCGCCGCGTTCCCCGACGGCCGAGCGGG
CCGCTCCGCGGGGCGGGCGGCCGTCGGGGAAAGCGGAGGCGGTAGCG	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+295,100
M,0;
ENSGALT00000018587	0	ENSGALT00000039283	167	37	100M	*	0	0	ACCCCCTCCCCTTCCTCAACCTCCCCTTCCCCCCTCTACCCCACCACCTCCCC
CACGGGACCGCCAGCAAGAAGCTGAGCAAGGTGGAGACGCTGCGCTC	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+167,100
M,0;
ENSGALT00000018587	0	ENSGALT00000039283	108	37	100M	*	0	0	CCTCTCGGGCGGCGGCACCGCGGCGGCGGCCGTGGCCCGGCGCAACGAACGGG
AGCGCAACCGCGTGCGGTTGGTCAACCTGGGCTTCGCCGCTCTACGG	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+108,100
M,0;
ENSGALT00000018587	0	ENSGALT00000039283	288	37	100M	*	0	0	CCTCCAGAGGCTCCTCGACGAACACGACGCCGCCGCCGCGTTCCCCGTCGGCC
GAGCGGGCCGCTCCGCGGGCGGGCGGCCGTCGGGGAAAGCGGAGGC	XT:A:U	NM:i:1	X0:i:1	X1:i:0	XN:i:1	XO:i:0	XG:i:0	MD:Z:47A52	XA:Z:ENSGALT00000039283,+288,100
M,1;
ENSGALT00000018587	0	ENSGALT00000039283	258	37	100M	*	0	0	GCTGCGCTCCGCCGTCGAGTACATCCGAGCCCTCCAGAGGCTCCTCGACGAAC
ACGACGCCGCCGCCGCGTTCCCCGACGGCCGAGCGGGCCGCTTCGCG	XT:A:U	NM:i:1	X0:i:1	X1:i:0	XN:i:1	XO:i:0	XG:i:0	MD:Z:95C4	XA:Z:ENSGALT00000039283,+258,100
M,1;
ENSGALT00000018587	0	ENSGALT00000039283	108	37	100M	*	0	0	CCTCTCGGGCGGCGGCACCGCGGCGGCGGCCGTGGCCCGGCGCAACGAACGGG
AGCGCAACCGCGTGCGATTGGTCAACCTGGGCTTCGCCGCTCTACGG	XT:A:U	NM:i:1	X0:i:1	X1:i:0	XN:i:1	XO:i:0	XG:i:0	MD:Z:69G30	XA:Z:ENSGALT00000039283,+108,100
M,1;
ENSGALT00000018587	0	ENSGALT00000039283	83	37	100M	*	0	0	GCTGCAAGCGCCGCTTGGCCTTCGCCTCCCTCTCCGCCGGCGCGCACCGCGCGC
GCGGCCGTGGCCCGGCGCAACGAACGGGAGCGCAACCGCGTGCGGTT	XT:A:U	NM:i:0	X0:i:1	X1:i:0	XN:i:0	XO:i:0	XG:i:0	MD:Z:100	XA:Z:ENSGALT00000039283,+80,100M
,0;
```

CIGAR operations

https://genome.sph.umich.edu/wiki/SAM read alignment cigar

The sequence being aligned to a reference may have additional bases that are not in the reference or may be missing bases that are in the reference. The CIGAR string is a sequence of of base lengths and the associated operation. They are used to indicate things like which bases align (either a match/mismatch) with the reference, are deleted from the reference, and are insertions that are not in the reference.

For example:

```
RefPos:     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T  G  A  A  C  T  G  A  C  T  A  A  C
Read: ACTAGAATGGCT
```

Aligning these two:

```
RefPos:     1  2  3  4  5  6  7     8  9 10 11 12 13 14 15 16 17 18 19
Reference:  C  C  A  T  A  C  T     G  A  A  C  T  G  A  C  T  A  A  C
Read:                  A  C  T  A  G  A  A     T  G  G  C  T
```

With the alignment above, you get:

```
POS: 5
CIGAR: 3M1I3M1D5M
```

The POS indicates that the read aligns starting at position 5 on the reference. The CIGAR says that the first 3 bases in the read sequence align with the reference. The next base in the read does not exist in the reference. Then 3 bases align with the reference. The next reference base does not exist in the read sequence, then 5 more bases align with the reference. Note that at position 14, the base in the read is different than the reference, but it still counts as an M since it aligns to that position.

- M: match, I: insertion, D: deletion, N: skipped reference region, S: soft clipping, H: hard clipping, P: padding, =: sequence match, X: sequence mismatch

Size of SAM vs BAM files

```
[[ablackpz@dev-intel18 DataSet159]$ ls -lah DataSet160E1R50G100_aln.*
-rw-r----- 1 ablackpz cse 275M Oct 18 14:32 DataSet160E1R50G100_aln.bam
-rw-r--r-- 1 ablackpz cse 2.0G Jan 18  2012 DataSet160E1R50G100_aln.sam
```