Introduction to NGS analysis on a Raspberry Pi

Beta version 1.1 (04 June 2013)

Contents

Overview

This analysis workbook will assume that you already have an idea about Next Generation Sequencing and have a general overview of the key stages involved in a data analysis starting with the reads as they come off of the sequencer. We have devised a simple example workflow comprising the following stages:

$$\begin{tikzcd}
\text{Simulated Illumina Reads} \arrow[d] \\
\text{Simple quality control} \arrow[d] \\
\text{Map against reference genome} \arrow[d] \\
\text{Variant discovery}
\end{tikzcd}$$

You will have the opportunity to map your reads using two different mappers and then compare the outputs. As we are using simulated data, we have generally applied the default analysis settings for each stage and method. Working with real life datasets, you should give consideration to the data type you are working with and the experimental objective in order to tune your analysis for optimal results.

Download some simulated reads

Due to the limited memory on the Raspberry Pi, we are limited in the size of dataset we can undertake NGS analysis on. As an example we are going to use a set of simulated reads which have been designed to be representative of Illumina reads that should align to the BRCA1 gene.

If you are using our pre-compiled SD card image these files have already been downloaded into the /home/pi/ngs directory.

More details on the source of this dataset can be found in the associated paper:

```
http://f1000research.com/articles/1-2/v1
```

And details of the dataset found at:

```
http://figshare.com/articles/Simulated_Illumina_BRCA1_reads_in
_FASTQ_format/92338
```

You will note that these are actually paired end reads, but for the purposes of this example, we will treat them as the results of a fragment run. Once you've worked though the example you may wish to work out how to repeat your analysis treating the data as paired end reads.

Download simulated reads

*** Only required if not using the pre-compiled SD card image**

Commands:

```
wget http://files.figshare.com/92198/Brca1Reads_1.1.fastq
wget http://files.figshare.com/92203/Brca1Reads_1.2.fastq
```

Files created:

```
Brca1Reads_1.1.fastq  Brca1Reads_1.2.fastq
```

Merge into a single fastq file.

Commands:

```
cat Brca1Reads_1.1.fastq Brca1Reads_1.2.fastq > reads.fq
```

Files created:

```
reads.fq
```

Check the number of reads

Command:

```
grep @chr reads.fq | wc -l
```

Output:

```
200000
```

Alternatively, we can use the –c flag in grep to return a count.

Command:

```
grep –c @chr reads.fq
```

Quality Control

The first stage in our analysis is to undertake some basic quality control analysis on the reads. To do this we will use the fastqc program from the Babraham Insititute:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

To start we will create an output directory for fastqc in our home directory:

Command:

```
cd ~
mkdir fastqc
```

Before running fastqc in command line mode and outputing the results into our newly created directory:

Command:

```
/home/pi/FastQC/fastqc -o fastqc reads.fq
```

Output:

```
Started analysis of reads.fq
...
Analysis complete for reads.fq
```

We can review the text output using less:

Command:

```
less fastqc/reads.fq_fastqc/fastqc_data.txt
```

The directory also contains HTML output and graphs which can be viewed using the Midori in the Raspberry Pi GUI (startx to load the GUI).

Map reads using bowtie

Before we can map our reads, we need to download and prepare a reference genome. We are going to download the hg19 chromosome 17 sequence from UCSC and then index it for use by bowtie.

The indexing process takes the text formatted reference sequence and undertakes a number of processing and sorting steps which produces a number of binary output files which are used by the mapper to make the search process faster than working with the raw text sequence.

In some of these analyses, we have added the command time before the analysis command to give an idea of the expected run time for a step.

Download chr17 in fasta format

Command:

```
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/chr17.fa.gz
```

Command:

```
gunzip chr17.fa.gz
```

Index the genome for bowtie and name it hg19_chr17

Command:

```
time bowtie-build chr17.fa hg19_chr17
```

Selected Output:

```
real    17m37.033s
```

Files created:

```
hg19_chr17.1.ebwt   hg19_chr17.3.ebwt   hg19_chr17.rev.1.ebwt
hg19_chr17.2.ebwt   hg19_chr17.4.ebwt   hg19_chr17.rev.2.ebwt
```

Map reads against a reference

There are many, many options that can be set/tweaked to change the way reads are mapped and results handled, more information can be found at:

```
http://bowtie-bio.sourceforge.net/manual.shtml
```

In our example we are going to use mostly the default options save for returning our alignments in SAM format.

Command:

```
time bowtie -S hg19_chr17 reads.fq bowtie.sam
```

Explanation of parameters:

```
-S             Print the alignments in sam format
hg19_chr17     Bowtie index to align reads against
reads.fq       fastq file of reads
bowtie.sam     Output file
```

Output:

```
# reads processed: 200000
# reads with at least one reported alignment: 197212 (98.61%)
# reads that failed to align: 2788 (1.39%)
Reported 197212 alignments to 1 output stream(s)

real   2m6.157s
```

Files created:

```
bowtie.sam
```

Reassign MAPQ values

Bowtie assigned the MAPQ value as either 0 or 255 to indicate either no match or match for each read. This is different to many other alignment programs and causes problems in further downstream analyses, which incorporate the MAPQ into their analysis, and interprets the 255 values as untrusted. To overcome this, we must reassign the MAPQ value in our file to a more sensible value which is easily achieved using sed.

Command:

```
sed 's/\t255\t/\t60\t/g' bowtie.sam > bowtie_mapq60.sam
```

Explanation of parameters:

```
s              Subsitute
\t255\t        Old value (tab 255 tab)
\t60\t         New value (tab 60 tab)
g              Global replacement – every occurrence found.
```

Files created:

```
bowtie_mapq60.sam
```

For efficiency, we now convert our SAM file into BAM format using samtools, then sort and index it.

Convert SAM to BAM

Command:

```
samtools view -bS bowtie_mapq60.sam > bowtie.bam
```

Explanation of parameters:

```
view           Use the samtools viewer program
-b             Output in BAM format
-S             Input in SAM format
> bowtie.bam   Pipe the output to file
```

Output:

```
[samopen] SAM header is present: 1 sequences.
```

Files created:

```
bowtie.bam
```

Sort the BAM file

Command:

```
samtools sort bowtie.bam bowtie.sorted
```

Explanation of parameters:

```
sort           User the samtools sort program
bowtie.bam     Input file
bowtie.sorted  Outut file (.bam appended automatically).
```

Files created:

```
bowtie.sorted.bam
```

Index the BAM file

Command:

```
samtools index bowtie.sorted.bam
```

Explanation of parameters:

```
sort              User the samtools index program
bowtie.sorted.bam   Input file
```

Files created:

```
bowtie.sorted.bam.bai
```

Check / view the mapping results

To check that our mapping has worked, we can use the text view in samtools to have a quick look and check of our alignments.

Command:

```
samtools tview bowtie.sorted.bam
```

To move to the BRCA gene, press

```
g
```

And then enter

```
chr17:41,196,311
```

Selected output:

```
41196311  41196321  41196331  41196341  41196351  41196361  41196371
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
GTGGAAGTGTTTGCTACCAAGTTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTACA
GTGGAAGTGTTTGCTACCAAGTTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTACA
GTGGAAGTGTTTGCTACCAAGCTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTACA
  GGAAGTGTTTGCTACCAAGTTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTATT
     AGTGTTTGCTACCAAGTTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTACA
```

Using a slightly different command we can define the reference genone and simplify the view to only show bases that differ when compared to the reference.

Command:

```
samtools tview bowtie.sorted.bam chr17.fa
```

Selected output:

```
41196311  41196321  41196331  41196341  41196351  41196361  41196371
GTGGAAGTGTTTGCTACCAAGTTTATTTGCAGTGTTAACAGCACAACATTTACAAAACGTATTTTGTACA
......................................................................
......................................................................
.....................C................................................
  ..................................................................TT
     .................................................................
```

Variant discovery using GATK

If we try running this bam file through the GATK pipeline, we get an error.

```
##### ERROR MESSAGE: SAM/BAM file bowtie.sorted.bam is malformed: SAM file doesn't
have any read groups defined in the header.  The GATK no longer supports SAM files
without read groups
```

Unfortunately, the latest version of GATK has tightened the compliance of each input file to the defined file formats. In our case, the read group information is missing from the SAM file. GATK detects our file has an incorrect/incomplete header, so we need to fix this using picard tools.

Command:

```
time java -Xmx400M -jar /home/pi/picard-tools-
1.87/AddOrReplaceReadGroups.jar I=bowtie.sorted.bam
O=bowtie_final.bam SORT_ORDER=coordinate RGPU=na RGID=1
RGLB=input RGPL=Illumina RGSM=Company CREATE_INDEX=True
```

Selected output:

```
[Fri May 10 11:44:10 UTC 2013] net.sf.picard.sam.AddOrReplaceReadGroups
INPUT=bowtie.sorted.bam OUTPUT=bowtie_final.bam SORT_ORDER=coordinate RGID=1
RGLB=input RGPL=Illumina RGPU=na RGSM=Company CREATE_INDEX=true    VERBOSITY=INFO
QUIET=false VALIDATION_STRINGENCY=STRICT COMPRESSION_LEVEL=5
MAX_RECORDS_IN_RAM=500000 CREATE_MD5_FILE=false

[Fri May 10 11:44:10 UTC 2013] Executing as pi@raspberrypi on Linux 3.6.11+ arm;
OpenJDK Zero VM 1.6.0_27-b27; Picard version: 1.87(1380)

INFO   2013-05-10 11:44:10 AddOrReplaceReadGroups    Created read group ID=1
PL=Illumina LB=input SM=Company


[Fri May 10 11:51:27 UTC 2013] net.sf.picard.sam.AddOrReplaceReadGroups done.
Elapsed time: 7.30 minutes.
```

Files created:

```
bowtie_final.bam
```

Finally, before we can undertake a variant discovery analysis we need to create a dictionary file for the genome, again using picard tools.

Command:

```
java -jar /home/pi/picard-tools-
1.87/CreateSequenceDictionary.jar R= chr17.fa O= chr17.dict
```

Selected output:

```
[Wed Mar 20 17:10:03 UTC 2013] net.sf.picard.sam.CreateSequenceDictionary done.
Elapsed time: 3.87 minutes.
```

Files created:

```
chr17.dict
```

Run GATK Unified Genotyper

Command:

```
java -jar /home/pi/GenomeAnalysisTK.jar \
     -R /home/pi/ngs/chr17.fa \
     -T UnifiedGenotyper \
     -I bowtie_final.bam \
     -o bowtie_snps.vcf \
     -L chr17:41,196,311-41,277,499
```

Explanation of parameters:

-R	Reference genome
-T	Select which tools to run
-I	Input file
-O	Output file
-L	Limit analysis to defined region

Selected output:

```
INFO  12:18:09,479 ProgressMeter -        Location processed.sites  runtime
per.1M.sites completed total.runtime remaining

INFO  12:18:39,520 ProgressMeter -  chr17:41196911        0.00e+00   30.0 s
49.6 w      0.7%        67.7 m    67.2 m

INFO  12:19:09,539 ProgressMeter -  chr17:41197811        0.00e+00   60.0 s
99.3 w      1.8%        54.1 m    53.1 m

INFO  12:19:39,560 ProgressMeter -  chr17:41198811        0.00e+00   90.0 s
148.9 w      3.1%        48.7 m    47.2 m

INFO  12:20:09,586 ProgressMeter -  chr17:41199711        0.00e+00  120.0 s
198.6 w      4.2%        47.8 m    45.8 m

...

INFO  10:28:55,292 ProgressMeter - Total runtime 2478.09 secs, 41.30 min, 0.69
hours

INFO  10:28:55,297 MicroScheduler - 0 reads were filtered out during traversal out
of 127221 total (0.00%)

INFO  10:29:03,514 GATKRunReport - Uploaded run statistics report to AWS S3
```

Files created:

```
chr17.dict
```

Check the number of variants identified

Command:

```
grep -v "^#" bowtie_snps.vcf | wc –l
```

Explanation of parameters:

-v	Inverse results – return lines not matching
"^#"	Fine lines starting with a # character
| wc	Pipe into word count

Output:

```
43
```

Map reads using BWA

In the same way that bowtie requires an indexed genome to run, we must first prepare an index in the format bwa requires. We will reuse the chromosome 17 fasta file we downloaded previously.

Command:

```
bwa index -a bwtsw chr17.fa
```

Output:

```
bwa_index] Pack FASTA... 18.40 sec
[bwa_index] Construct BWT for the packed sequence...
[BWTIncCreate] textLength=162390420, availableWord=23426280
[BWTIncConstructFromPacked] 10 iterations done. 38642372 characters processed.
[BWTIncConstructFromPacked] 20 iterations done. 71387764 characters processed.
[BWTIncConstructFromPacked] 30 iterations done. 100487892 characters processed.
[BWTIncConstructFromPacked] 40 iterations done. 126348132 characters processed.
[BWTIncConstructFromPacked] 50 iterations done. 149328740 characters processed.
[bwt_gen] Finished constructing BWT in 57 iterations.
[bwa_index] 752.36 seconds elapse.
[bwa_index] Update BWT... 8.34 sec
[bwa_index] Pack forward-only FASTA... 15.06 sec
[bwa_index] Construct SA from BWT and Occ... 185.51 sec
[main] Version: 0.6.2-r126
[main] CMD: bwa index -a bwtsw chr17.fa
[main] Real time: 983.354 sec; CPU: 979.670 sec
```

Files created:

```
chr17.fa.amb  chr17.fa.ann  chr17.fa.bwt  chr17.fa.fai
chr17.fa.pac  chr17.fa.sa
```

Having done this we can then map our reads in fastq format and pipe the sai output into a file.

Command:

```
bwa aln chr17.fa reads.fq > bwa.sai
```

Selected output:

```
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 434.41 sec
[bwa_aln_core] write to the disk... 0.23 sec
[bwa_aln_core] 200000 sequences have been processed.
[main] Version: 0.6.2-r126
[main] CMD: bwa aln chr17.fa reads.fq
[main] Real time: 441.115 sec; CPU: 440.480 sec
```

Files created:

```
bwa.sai
```

Finally we can convert the sai file into SAM format for downstream processing. During this process we can add the read group information we needed to add to the bowtie data using picard tools.

Command:

```
bwa samse -f bwa.sam chr17.fa bwa.sai reads.fq -r
"@RG\tID:1\tSM:1"
```

Explanation of parameters:

```
-r               Add read group information
```

Output:

```
[bwa_aln_core] convert to sequence coordinate... 7.61 sec
[bwa_aln_core] refine gapped alignments... 2.66 sec
[bwa_aln_core] print alignments... 8.92 sec
[bwa_aln_core] 200000 sequences have been processed.
[main] Version: 0.6.2-r126
[main] CMD: bwa samse -f bwa.sam -r @RG\tID:1\tSM:1 chr17.fa bwa.sai reads.fq
[main] Real time: 27.764 sec; CPU: 25.080 sec
```

Files created:

```
bwa.sam
```

Convert SAM to BAM

Command:

```
samtools view -bS bwa.sam > bwa.bam
```

Explanation of parameters:

```
view          Use the samtools viewer program
-b            Output in BAM format
-S            Input in SAM format
> bowtie.bam  Pipe the output to file
```

Output:

```
[samopen] SAM header is present: 1 sequences.
```

Files created:

```
bwa.bam
```

Sort the BAM file

Command:

```
samtools sort bwa.bam bwa.sorted
```

Explanation of parameters:

```
sort           User the samtools sort program
bowtie.bam     Input file
bowtie.sorted  Outut file (.bam appended automatically).
```

Files created:

```
bwa.sorted.bam
```

Index the BAM file

Command:

```
samtools index bwa.sorted.bam
```

Explanation of parameters:

```
sort               User the samtools index program
bowtie.sorted.bam    Input file
```

Files created:

```
bwa.sorted.bam.bai
```

Run GATK on BWA mapping

Command:

```
java -jar /home/pi/GenomeAnalysisTK.jar \
     -R /home/pi/ngs/chr17.fa \
     -T UnifiedGenotyper \
     -I bwa.sorted.bam \
     -o bwa_snps.vcf \
     -L chr17:41,196,311-41,277,499
```

Explanation of parameters:

-R	Reference genome
-T	Select which tools to run
-I	Input file
-O	Output file
-L	Limit analysis to defined region

Selected output:

```
INFO  14:26:28,538 ProgressMeter -        Location processed.sites  runtime
per.1M.sites completed total.runtime remaining

INFO  14:26:58,580 ProgressMeter -  chr17:41197011        0.00e+00   30.0 s
49.7 w      0.9%        58.0 m    57.5 m

INFO  14:27:28,601 ProgressMeter -  chr17:41198011        0.00e+00   60.0 s
99.3 w      2.1%        47.8 m    46.8 m

INFO  14:27:58,621 ProgressMeter -  chr17:41199011        0.00e+00   90.0 s
149.0 w      3.3%        45.1 m    43.6 m

INFO  14:28:28,641 ProgressMeter -  chr17:41200111        0.00e+00  120.0 s
198.6 w      4.7%        42.7 m    40.7 m

INFO  14:28:58,661 ProgressMeter -  chr17:41201111        0.00e+00    2.5 m
248.3 w      5.9%        42.3 m    39.8 m

INFO  14:29:28,681 Pro

...

NFO  15:05:06,010 ProgressMeter - Total runtime 2317.50 secs, 38.62 min, 0.64 hours

INFO  15:05:06,015 MicroScheduler - 0 reads were filtered out during traversal out
of 114918 total (0.00%)

INFO  15:05:14,784 GATKRunReport - Uploaded run statistics report to AWS S3
```

Files created:

```
bwa_snps.vcf
bwa_snps.vcf.idx
```

Check the number of variants identified

Command:

```
grep –v “^#” bowtie_snps.vcf | wc -l
```

Explanation of parameters:

–v	Inverse results – return lines not matching
“^#”	Fine lines starting with a # character
| wc	Pipe into word count

Output:

```
     19
```

Compare SNPs between mappers

The easiest way to compare the differences in SNPs found by the two mappers is to do a comparison of the locations at which a SNP has been found between the files.

We will first extract the SNP locations from our bowtie vcf file:

Command:

```
grep -v "^#" bowtie_snps.vcf | cut -f 1,2 | tr "\t" ":" | sort
> bowtie_snps.txt
```

Explanation of parameters:

```
grep –v “^#”        Extract lines which don’t start with a #
cut –f 1,2          Extract the first two columns of data
tr “\t” “:”         Replace tabs with a colon
sort                Sort the SNP positions
> bowtie_snps.txt   Output to file
```

We then repeat this process on the bwa vcf file:

Command:

```
grep -v "^#" bwa_snps.vcf | cut -f 1,2 | tr "\t" ":" | sort >
bwa_snps.txt
```

You will now have two text files which looking similar to this:

Selected Output:

```
chr17:41201130
chr17:41201198
chr17:41209153
chr17:41215396
```

The final stage in the process is to use sdiff to undertake a side by side comparison of our two position files:

Command:

```
sdiff bowtie_snps.txt bwa_snps.txt
```

Selected Output:

```
chr17:41200031                                        <
chr17:41200036                                        <
chr17:41200040                                        <
chr17:41201130                                         chr17:41201130
chr17:41201198                                         chr17:41201198
chr17:41209153                                         chr17:41209153
chr17:41215396                                         chr17:41215396
chr17:41215756                                         chr17:41215756
chr17:41216205                                         chr17:41216205
chr17:41223265                                         chr17:41223265
chr17:41226398                                        <
```

You will see that variant discovery on data mapped using bwa has identified SNPs at all of the locations that analysis on data mapped by bowtie, but has identified a number of additional SNPs. You can investigate these positions and the reads mapped to them further using the samtools tview tool.

Summary of File Formats

fa / fasta

fai

.fa and .fasta files contain text in the classic format for representing nucleotide sequences:

```
>chr17

AAGCTTCTCACCCTGTTCCTGCATAGATAATTGCATGACAATTGCCTTGT

CCCTGCTGAATGTGCTCTGGGGTCTCTGGGGTCTCACCCACGACCAACTC

CCTGGGCCTGGCACCAGGGAGCTTAACAAACATCTGTCCAGCGAATACCT
```

The fai is an index file which accompanies the fa/fasta file.

fq / fastq

A fq/fastq file contained sequence information along with quality information about the sequence. It is the typical non-propietary output from a next generation sequencer.

A fastq file normally uses four lines per sequence.

```
@chr17:95885:F:237/1

GCGAACACATCCATGTGCCGGGAGGATGGTGCACCCCAACTCCACAAGGACCCTTCCAGACCTCACTCCCTGGGTGCCGTCAT
GAGAGCC

+

@<@?DDDD>FF:DAF9FFFCAGF<F3AAFD>2ACEF?CFC@?;FB:?@?;D@>86';EE;AE376?################
#######

@chr17:42746:F:247/1

TATCACCCAGTGTTGGCAAGGTACAGGAAAATGGGAACTATCATATACCACAGGGGCTGGAAGAGCATAAACTGGTTTAATCT
TTCTAAA

+
```

Line 1 begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line).

Line 2 is the raw sequence letters.

Line 3 begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again.

Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

sam

SAM stands for Sequence Alignment/Map format. The SAM Format is a text format for storing sequence data in a series of tab delimited ASCII columns.

```
@HD     VN:1.0 SO:unsorted

@SQ     SN:chr17        LN:81195210

@PG     ID:Bowtie       VN:0.12.7       CL:"bowtie -S hg19_chr17 reads.fq bowtie.sam"

chr17:95885:F:237/1 0       chr17  41292204       255     90M     *       0       0
        GCGAACACATCCATGTGCCGGGAGGATGGTGCACCCCAACTCCACAAGGACCCTTCCAGACCTCACTCCCTGGGTG
CCGTCATGAGAGCC
        @<@?DDDD>FF:DAF9FFFCAGF<F3AAFD>2ACEF?CFC@?;FB:?@?;D@>86';EE;AE376?##########
##############       XA:i:0 MD:Z:66C1T0G2T0A1C0T1T4C0T1G0C0T0G0      NM:i:14
```

A full explanation of the format and the data in each column can be found at:

```
http://samtools.sourceforge.net/SAM1.pdf
```

bam

bam.bai

The bam format is a BGZF compressed version of a sam file. The bam.bai file is an index file of the compressed index speed access to the compressed data. Again full details of the format can be found at:

```
http://samtools.sourceforge.net/SAM1.pdf
```

vcf

vcf.idx

VCF is a text file format containing information about variant calls. It contains meta-information lines, a header line, and then data lines, each containing information about a position in the genome. There is an option whether to contain genotype information on samples for each position or not.

The vcf.idx file is an index file to enable faster searching of the information by certain programs.

```
##fileformat=VCFv4.1
...
##contig=<ID=chr17,length=81195210>
##reference=file:///home/pi/ngs/chr17.fa
#CHROM POS     ID      REF     ALT     QUAL    FILTER INFO    FORMAT 1
chr17  41201130       .       A       G       4167.77        .
       AC=2;AF=1.00;AN=2;DP=110;Dels=0.00;FS=0.000;HaplotypeScore=2.8933;MLEAC=2;ML
EAF=1.00;MQ=36.72;MQ0=0;QD=37.89  GT:AD:DP:GQ:PL       1/1:0,109:109:99:4196,325,0
chr17  41201198       .       C       T       3911.77        .
       AC=2;AF=1.00;AN=2;DP=102;Dels=0.00;FS=0.000;HaplotypeScore=2.3113;MLEAC=2;ML
EAF=1.00;MQ=36.80;MQ0=0;QD=38.35  GT:AD:DP:GQ:PL       1/1:0,101:101:99:3940,304,0
```

More information available online at:

```
http://www.1000genomes.org/wiki/Analysis/
```

```
Variant%20Call%20Format/vcf-variant-call-format-version-41
```

ebwt

Genome index files for bowtie are labeled with a ebwt suffix. These files are formed from a reference sequence and enable faster searching of reads against a reference genome.

fa.amb / fa.ann / fa.bwt / fa.pac

There are index files created for bwa mapping. Their content is as follows:

amb is a text file, to record appearance of N (or other non-ATGC) in the ref fasta.

.ann is a text file, to record ref sequences, name, length, etc.

.bwt is a binary, the Burrows-Wheeler transformed sequence.

.pac is a binary, packaged sequence (four base pairs encode one byte).

.sa is a binary, suffix array index.

Fault finding / error checking

During the development and testing of these materials an error occurred on one occasion, which was related to the memory cache becoming full and corrupted. A reboot didn't fix this issue.

If you do come across errors which result in the memory intensive stages of your analysis failing, you may find the following commands will fix this issue.

First check that your cached memory is taking up more memory than your free memory:

Command:

```
free –m
```

If you do find that your memory cache is full and stopping things running, the following command will flush it and hopefully enable your analysis for run without issue:

Command:

```
sudo sh -c "echo 3 > /proc/sys/vm/drop_caches"
```

Required installs (if not using precompiled image)

samtools

```
sudo apt-get install samtools
```

bwa

```
sudo apt-get install bwa
```

bowtie

```
sudo apt-get install bowtie
```

picard-tools

```
sudo apt-get install picard-tools
```

This also installs java

GATK

```
cd /home/pi
wget http://xoanon.cf.ac.uk/rpi/GenomeAnalysisTK.jar
```

Latest picard-tools

```
cd /home/pi
wget http://xoanon.cf.ac.uk/rpi/picard-tools-1.87.tar.gz
gunzip picard-tools-1.87.tar.gz
tar -xvpf picard-tools-1.87.tar
```