Best Practices for handling NGS Data

Author: Petros Mina, PhD

Table of Contents

Introduction

Early sequencing techniques exploited the template-based replication of DNA, catalysed in part by the DNA polymerase enzyme. DNA polymerase builds a new strand of DNA by reading sequence information (deoxynucleotides) from an existing strand of DNA (the template strand), and incorporating into the new strand an opposing deoxynucleotide using the G-C, A-T bidirectional matching rule (Berg, 2002).

A specific form of nucleotides, known as dideoxynucleotides cannot be catalysed by the polymerase enzyme and this stops the formation of bonds between adjacent nucleotides. This procedure can be exploited for interrogating the sequence of DNA. Using four reaction mixtures containing:

(i) a radiolabelled dideoxynucleotide (different dideoxynucleotide per mixture),

(ii) regular deoxynucleotides,

(iii) DNA template and,

(iv) DNA polymerase,

four different sequencing processes can be initiated which proceed until a dideoxynucleotide is incorporated into the newly built strand. The random incorporation of the dideoxynucleotide produces variable length DNA sequences. Gel electrophoresis of the reaction mixtures can then be used to size-separate and visualise the fragments in the form of DNA bands on the gel, allowing the sequence to be read-off from bottom to top (Berg, 2002).

Various technologies have since been developed in order to automate variants of the sequencing process into a more time-efficient procedure, leading to the advent of what is known as NGS sequencing. Various NGS platforms have been developed by competing companies. NGS sequencing, allows the sequencing of many samples in parallel (what is known as a multiplex reaction) in a cost-effective and time efficient manner (Barba, 2014). A historical overview of DNA sequencing and a description of the NGS procedure is given in (Mardis, 2013) and (Kulski, 2016).

NGS technologies can be used to construct sequences *de novo* (gain information into a previously unknown sequence and construct a reference genome) or for interrogation purposes (to sequence and quantify a DNA sample by aligning it to an existing reference genome). This document attempts to summarise “good practice” procedures, and can be used as a go-to guide, for the latter. The guide also provides examples of the various steps of the described procedure. Please note that the examples are not intended to promote any particular software. In the examples a dollar sign ($) indicates filenames or variables that are used in the commands and a dash sign (-) or double dash sign (--) to invoke particular options of the program and are defined in text.

Overview

Various tools and processes can be used to obtain a reference-aligned file from which data can be extracted from. The development of various tools has been accommodated in part by standardisation of data into various community-accepted formats. Examples of widely used, community-accepted standardised formats that are encountered during NGS data handling include:

- the FASTa Quality (FASTQ) format (Cock, 2016); a text-based format for storing nucleotide sequences and their attributed sequencing quality score,
- the Sequence Alignment/Map (SAM) format (Li, 2009); a text-based format for storing nucleotide sequences that have been aligned to a reference genome,
- the BAM format (Li, 2009); a binary version for the SAM format,
- the mpileup format (SAMtools documentation); a text-based format that describes the sequence information at each chromosomal position and,
- the Variant Call Format (VCF) format (Danecek, 2011); a text-based format for storing variant sequence information.

An extensive list of accepted formats in the biology community is found in the Wikipedia file type format entry (Wikipedia Biology Formats). In general, NGS data processing can be conceptually divided into the following sequential stages:

1. conversion of raw platform-specific data to the accepted text-based FASTQ data format,
2. pre-processing of FASTQ data to remove non-reference genome sequences
3. alignment of FASTQ data to a reference genome to produce a SAM/BAM file,
4. post-alignment processing to refine the aligned file, and
5. data extraction from the BAM file.

We discuss these processes, and various pitfalls to be aware of, in the sections that follow. It is important to note that most of these software were developed and intended to work on UNIX-based operating systems and may not be available for Windows-based operating systems. Furthermore, many of these use the command-line interface of a UNIX-based operating system and there rarely exists a Graphical User Interface (GUI) to accommodate the end-user. As such, some familiarity in using command-line programs of UNIX systems is required.

Conversion of raw data to the FASTQ format

Conversion to FASTQ

As discussed above, it is important to work in a community-accepted format to accommodate data exchange and data processing between parties. NGS sequencing platforms may store their data into

a proprietary format. Thus, the first step is to convert the platform-specific data to the accepted FASTQ format, as it is required by most alignment software.

Most NGS experiments are multiplex reactions, reactions where more than one sample is part of the sequencing procedure. Biochemically, this is implemented by adding non-genomic sequences to the ends of DNA fragments, known as indexes. Index sequences are specific oligonucleuotide sequences that are used to tag and identify a sample (see also pre-processing of fastq data section). Bioinformatically, the indexes are used during the conversion process to sort the data into separate files per sample, in a process known as demultiplexing. That is, for every sample a separate file containing the information for that sample is produced.

The vendor of each platform may have their own software to perform the process of conversion. For example, Illumina ® sequencing platforms store the data into .bcl format and the company offers the bcl2fastq software which converts the .bcl data into the .fastq format. There are also non-vendor software that are available to perform the conversion process, such as Flexbar, Kraken, Bayexer, deML and several others (Omicstools website).

A common error that may come up during conversion is the existence of conflicting index sequences. That is, sequences which cannot be separated efficiently into separate samples. It is best practice to remove all samples that have conflicting sequences as the conversion software might mix up sequencing information between samples.

Quality Control of the FASTQ files

As a first step after conversion from proprietary sequencing format to the fastq format, one can perform quality control (QC) on the unaligned data to indicate of any possible sequencing artifacts that could affect downstream processes. Examples of tools that provide this QC step are FastQC, ShortRead, FastX and PRINSEC.

One of the most commonly used software is FastQC. It provides a QC report which can detect issues that can originate either in the sequencer or in the starting library material. It supports formats other than FastQ, such as Casava FastQ, Colorspace FastQ, fastq.gz files as well as the aligned data file formats SAM and BAM.

FastQC can be run using the following command:

```
fastqc $input.fastq
```

and produces quality score statistics for the input fastq file (input.fastq) that can be viewed in an HTML browser. FastQC may also run interactively. Examples of what is considered “good data” and “bad data” are given on the FastQC homepage (see appendix for URL address).

Pre-processing of FASTQ data

Part of the wet-lab process that prepares samples for NGS involves the addition of various non-genomic sequences to the DNA fragments. These sequences can be used to identify a sample during the demultiplexing process (index sequences), or recognise clonal reads (barcode sequences), or are required to bind DNA fragments to the surface of the sequencing platform in order to initiate the process of sequencing (adaptor sequences) (Head, 2014) .

In addition to this, during sequencing several segments might be very short and/or attributed a low quality score by the sequencer. Very short segments are hard to align to unique positions in the reference genome (just by chance the same sequence will occur somewhere else in the genome too) and poor quality score reads might not have the correct contiguous sequence detected. Thus, it is good practice to remove reads shorter than 25 bases and with a quality score of under Q25 (the scoring is attributed by the sequencer during sequencing). Furthermore, if the data are from paired-end sequening it is best practice to remove the corresponding read even if it is not affected.

Several “trimming” software exists to perform this process such as cutadapt, trim galore! and fastx_clipper. An example using cut-adapt is provided below:

```
cutadapt -q 25 --minimum-length 25 -a $ADAPTORSEQ1 -A
$ADAPTORSEQ2 -o $my_trimmedR1.fastq -p $my_trimmedR2.fastq
$myfastqR1.fastq $myfastqR2.fastq
```

The cutadapt command will remove reads with quality score under 25 (-q 25), reads that are smaller than 25 bp (--minimum-length 25), with forward and reverse adapters (-a/-A options specifying the adaptor sequences) and store the output into the my_trimmedR1.fastq and my_trimmedR2.fastq files respectively. The command will remove reads from both input fastq files. That is, if a read is affected in one file, the corresponding pair will also be removed from the other file. The input FASTQ files in this instance are myfastqR1.fastq myfastqR2.fastq files for the forward and reverse read respectively. In single end sequencing, only one fastq file is present.

Reference Genome Alignment

Sequence alignment

Several software exists that performs alignment to a reference genome. As a minimum these software require that you provide the (trimmed) FASTQ file and a reference genome file to align to. Some software provide options such as aligning only segments/fragments of a particular score, reporting of unique/multiple hits or specifying a maximum number of allowed mismatches between reference and detected sequence. Examples of software that can be used for alignment are bwa, bowtie, MAQ, SOAP and segemehl.

Important trade-offs to consider is the accuracy of alignment and time required to align. Generally the more accurate the alignment the longer it takes to finish. Most algorithms build an index of the reference genome and use this as a way of increasing speed during alignment. Index-building only needs to be performed once per reference genome and it is best to do it before any alignment processes are initiated. By default aligners will search for the index in the path where the reference genome exists so it is best to keep the index along with the reference genome in the same folder. It is good practice to choose an aligner that takes into account the quality of sequenced reads and to specify the value of the parameter in the command, in case the previously used trimming tool does not remove bad quality reads (see pre-processing of FASTQ data section).

If more than one file exists for a single sample then several strategies can be considered on how to proceed with alignment. For paired-end sequencing, separate forward and separate reverse reads (fastq files) for a sample need to be merged into one forward and one reverse read prior to alignment. This can be achieved using the "cat" command from the UNIX operating system terminal on the fastq files of interest.

In some cases a sample will be found in multiple compartments (sequencing lanes) of the sequencing apparatus (flowcell). In this instance separate sequence files will be produced for the file; files from each sequencing lane. One strategy would be to merge all forward reads from all lanes into one file and all reverse reads from all lanes into another file and then provide these files to the aligner for sequencing. Another strategy would be to align separately files from different lanes and then merge them together (see also section on merging separate BAM files). It is sometimes a better computational strategy to proceed with the latter as it is more time- and memory-efficient to align smaller sized files, and then merge them together at a later stage. As such, the choice of operations may be guided by the available computational resources.

Below we present an example of producing a reference genome index and performing sequence alignment using the bwa software:

```
bwa index $REF

bwa mem $REF -t $cpus -M $fastq1 $fastq2 > $samfile
```

The first command (bwa index) builds an index file for the reference genome $REF. The second command performs sequence aligment using the "mem" algorithm to align the forward read ($fastq1) and reverse read ($fastq2) to the reference genome ($REF). The process uses parallel processing (-t option) employing many cores ($cpus is the integer number of cores) to complete the operation in a shorter amount of time. The -M option is used to flag secondary hits (non-unique alignments) as this is necessary for some downstream analysis tools. The resulting SAM file ($samfile) will store the output data.

Conversion of SAM to BAM

Most downstream analysis tools work on the binary format alignment file, namely the BAM file. Thus, it is good practice to convert the aligned SAM file to its binary version, the BAM file. Examples of tools that can be used to convert the SAM file to the BAM file include the SAMtools view module and the Picard SortSam module.

When interrogating an aligned BAM file, sequences appear in the same order as they were in the FASTQ file. To be able to quickly manipulate the BAM file, it requires that the aligned sequences appear in the same order as that of the reference genome of interest. Examples of tools that can be used to produced an ordered SAM/BAM file include, the SAMtools sort module and the Picard SortSam module.

Finally, most programs that manipulate BAM files require an index file. This usually has the .bai extension. In broad terms, the index file acts as a map, pointing out key locations of data in the corresponding BAM file (SAMtools documentation). In this way the entire file does not need to be loaded or processed in order to locate/view/extract information from the file. Instead, the tool will first read the index file and minimise the search space accordingly in the approximate location of the information of interest.

Below is an example using the SortSam module, part of the Picard suite of tools, to convert the SAM file to a BAM file. In this example we return a sorted BAM file and the corresponding index file using the same command.

```
java -jar $SortSam.jar SO=coordinate INPUT=$samfile
OUTPUT=$bamfile VALIDATION_STRINGENCY=LENIENT
CREATE_INDEX=true
```

The input SAM file (INPUT=$samfile) is converted to a BAM file (OUTPUT=$bamfile). The index file will be created and named as $bamfile.bai when the CREATE_INDEX option is set to true. The (SO) option indicates the type of sorting to be performed. SO is coordinate sorting and is the usual preference.

Post-Alignment Processing of data

Merging of separate BAM files

In some instances it may be necessary to merge separate BAM files into one file. For example, the researcher may require a higher read-depth for a particular sample and decide that it will be sequenced many times over different runs so as to achieve the desired depth. In other instances, for

some NGS platforms the same sample may be found across different compartments of the sequencing apparatus and data generated from the different compartments need to be merged to a single file.

Examples of software that can be used to merge separate BAM files to a single file are the SAMtools merge module, the Picard MergeBamAlignment and Picard MergeSamFiles modules and the sambamba merge module. An example of using sambamba to perform the merging is given below:

```
sambamba merge --nthreads $cpu $bamfileOUT $bamFileIN1
$bamFileIn2
```

The sambamba tool has a parallel processing option (--nthreads) which can be used to specify the number of threads ($cpu) to commit to the process. The output file ($bamfileOUT) is the result of the merging process and holds data from the constituent files ($bamFileIN1/2). More than two BAM files can be processed at the same time.

Removal of Duplicates in targeted sequencing

In some embodiments, the DNA material to be sequenced may undergo a selection process where specific regions of the genome under investigation may be targeted and enriched for (also known as deep sequencing). A description of various enrichment methods may be found in (Mamanova, 2010). The enrichment process may introduce duplicate fragments whose presence may introduce additional biases during subsequent (statistical) analysis of data.

Depending on the purpose of downstream analysis, duplicate fragments may need to be removed. Examples of software that can be used are the SAMtools rmdups module and the Picard MarkDuplicates module. An example of using Picard to remove duplicates is given below:

```
java -jar MarkDuplicates.jar INPUT=$bamfile
OUTPUT=$bamfile_dupFree.bam  REMOVE_DUPLICATES=true
METRICS_FILE=metrics CREATE_INDEX=true
VALIDATION_STRINGENCY=LENIENT
```

The input is a sorted BAM file ($bamfile) (see also Conversion of SAM to BAM section) and the output is a duplicates-free BAM file ($bamfile_dupFree.bam). This is achieved when the REMOVE_DUPLICATES option is set to true . If not, the duplicate fragments are simply marked and not removed from the file. The CREATE_INDEX option will also create an index file that helps in quick and efficient data processing of the file using downstream analysis tools. The index file

will have the same name as the specified output file with an added .bai suffix.

Retrieval of data from the BAM file

Data analysis can be performed once a final BAM file is available. There are several tools that can be used to obtain information from, or process BAM files. Examples of desired information are the coverage and sequence information obtained at specific genomic loci. Examples of software that can be used to extract information from a BAM file are the bedtools suite, the SAMtools suite, the BAMtools suite, the Picard suite and the GATK suite. In this section we provide certain examples of data retrieval from the final BAM file.

Coverage and sequence information

Examples of tools that may be used for obtaining coverage and sequence information are the SAMtools mpileup module, the GATK suite, various bedtools modules and the BAMtools coverage or BAMtools count modules. An example, using SAMtools, is given below:

```
samtools mpileup -f $REF -l $target_bed $bamfile -A > $info.pileup
```

Where $REF is the reference genome used during alignment, and $target_bed is a file which provides the coordinates of interest known as a bed file. A bed file is a tab-separated file where each row points to a single genomic region. In it's simplest form, the first column points to the chromosome of interest, the second column to the start coordinate and the third column to the stop coordinate. Up to 12 columns of data can be included in the bed file. It is important that all rows have the same number of columns otherwise certain software will not perform as expected.

The output file ($info.pileup) holds the results of the interrogated regions. Each row of the output file has information on the genomic coordinate, the reference genome nucleotide at the particular position, the times the nucleotide was sequenced and ASCII encoded information on differences between reference and physical sequence (see also section on variant detection).

The user should take care as some tools use zero-indexing (0-index) when accessing the aligned file and others may use one-indexing (1-index). A tool that uses 0-indexing begins counting the genome at the zero-th position. A tool that uses 1-indexing begins counting the genome with position one. As such, care should be taken when using separate tools; coordinates may differ by one position and the information returned by the tool may not reflect the information requested. We illustrate via example.

Sequence	A	G	T	C	G	C	A	C	C	G	C	T	C	C	A
1-index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

0-index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Using the table above, when requesting sequence information between positions 3 and 10 the 1-index tool will return the sequence TCGCACCG whilst the 0-index tool will return CGCACCGC.

GC-bias

When requesting coverage information, the user should be aware that certain sequencing methods may be biased towards sequencing certain genomic sequences more than others. A well known bias, is that of GC-content (Benjamini 2012 and Chen 2013). A sequence of interest may be covered more times than other sequences because of differences in GC-content (the proportion of G and C nucleotides in a sequence). As such, the user is advised to investigate any coverage vs sequence associations before proceeding to subsequent analysis, as it may affect conclusions.

An example of GC-bias is illustrated in Figure 1 (Benjamini 2012). GC-content and read-depth coverage are estimated from sequence data using 10kb windows across the genome and then plotted. A line of best-fit through the data illustrates a non-linear relationship and provides the expected coverage given GC-content. This association may help remove GC-content bias. For example, the read-depth achieved in a 10kb segment may be divided by the expected read-depth achieved for that particular GC-content. After GC-bias removal, coverage information obtained from different GC-content regions can be compared against each other.

Some of the exemplary tools given as examples in this guide may also have GC-bias estimation methods, such as for example GATK.

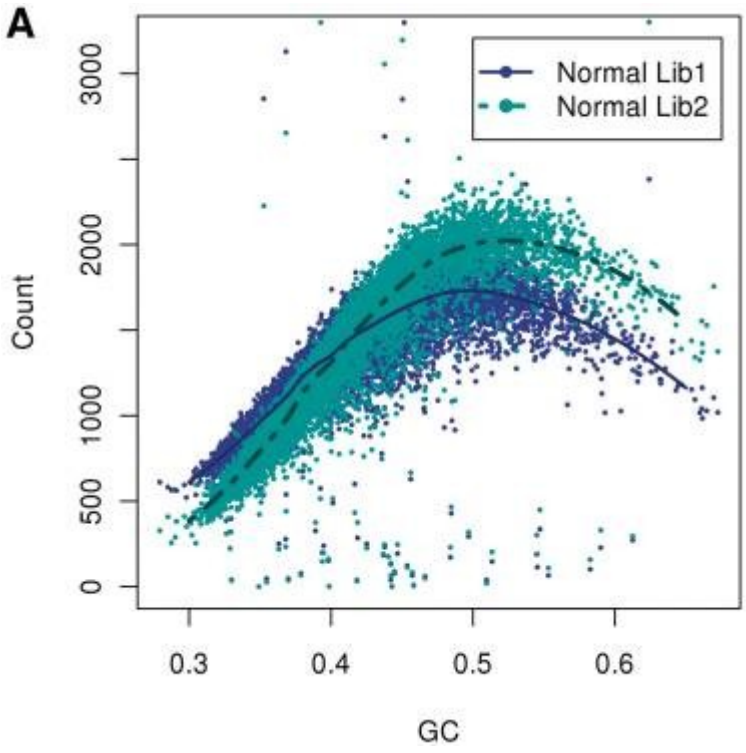

Figure 1: The figure illustrates the association between GC-content and sequence coverage. As illustrated by the figure, a region may be preferentially sequenced more times, dependent on its GC-content. The y-axis represents coverage and x-axis the GC-content. Each data point presented is the GC-content and read-depth of a 10kb bin. Two separate sequenced samples are presented. A line of best fit can be estimated (solid blue and green dotted lines) from the data using LOESS methods. The figure is adapted from (Benjamini 2012).

Variant detection

An open research question is the detection of variants. Variants are defined as sequence/structural differences that exist between the sequenced segment and the reference genome. Variant calling efficiency depends on the combination of the aligner used for sequence alignment and the tool used to perform the detection. A comparison/review of results obtained using various combinations of popular tools can be found in (Hwang, 2015). Another review focusing on variant callers is found in (Krøigård, 2016). As seen in the reviews (Hwang, 2015 and Krøigård, 2016) each combination of tools will produce different results, as each tool has its own algorithm/methodology for detection. Depending on the research question to be answered, it is best to use more than one tool and give more weight to the common results generated by the employed methods. The consensus human readable format used for variant call data is the VCF format (Variant Call Format). Binary versions may also exist such as the BCF format. Below we summarise some of the applicable tools.

GATK tools

There are various tools from the GATK suite which can be used to call variants. Several guidelines are published on the GATK website on what is considered “best practice” to perform variant calling

(https://software.broadinstitute.org/gatk/best-practices/). For purposes of completion, a short description of some of the tools is found below. However, the reader is urged to read the GATK tools website for additional information.

RealignerTargetCreator is a a local realignment tool, which can accept one or more BAM files. It uses full alignment context to determine whether an appropriate alternate reference sequence (eg indel) exists. Following local realignment, other GATK tools can be used to sensitively and specifically identify indels and/or SNPs, such as the IndelRealigner tool. The BaseRecalibrator tool assesses the probability of errors at genomic locations where a base is identified as a SNP but that particular site is not part of a user-provided SNP database, such as for example dbSNP. Other GATK tools include the PrintReads tool which can be used to reassess the quality scores in the QUAL field of each read in the output BAM and this may be useful towards SNP calling. The user is urged to read the documentation of these and other tools before usage (GATK website).

General usage of the GATK tools is as follows:

```
java -jar GenomeAnalysisTK.jar \
 -T GATKtoolToUse \
 -R example_reference.fasta \
 -I example_reads.bam
```

where `-jar` GenomeAnalysisTK.jar invokes the GATK engine itself, `-T` indicates the tool the user wants to initiate, `-R` points to the reference genome file and `-I` to the input file that will be processed with the GATK engine/tool.

Samtools mpileup and bcftools

The output pileup file produced when samtools mpileup is executed contains read-depth and sequence information. Each row provides details related to the reference genome coordinate that is interrogated (ie chromosome and base number) such as the reference genome nucleotide found at the particular position, how many times the nucleotide was sequenced and ASCII encoded information on any differences between the reference and the detected sequence. The encoded information can be piped into bcftools which parses the results to identify variants and store the output in a VCF file. An example of usage of the samtools-bcftools combination is given below:

```
samtools mpileup -ugf $REF -l $REGIONS.bed $BAMfile |
bcftools view -bvcg - > $mybcf_file
```

```
bcftools view $mybcf_file| vcfutils.pl varFilter -d $DEPTH >
$myvcf_file
```

The -ugf is a series of three options that instruct the tool to generate an uncompressed BCF file (-u), of genotype likelihoods (-g) using an indexed reference genome (-f), $REF. The operation can be performed over the entire file ($BAMfile) or few select regions listed in the $REGIONS.bed file and parsed to the program with the respective option (-l). The generated file is piped (|) to the bcftools view program to call SNPs . The -bvcg options store the output in a binary file (-b), including the identified variant sites (-v and -c) and genotypes (-g). This binary file is further processed using the vcfutils.pl perl script, which is part of th SAMtools suite, to output a human readable format of the file utilising a minimum read-depth (-d) option. The minimum read-depth option calls out variants that have been sequenced at least $DEPTH times. A minimum read-depth may be desired to ensure that any variant calls are not the result of sequencing error (a small probability that the sequenced base was not correctly identified/called).

VarDict

VarDict (Lai, 2016) can identify single and multiple nucleotide variants, indels and other complex and structural variants. It can perform variant calling on single and multiple samples, comparisons between normal and tumourigenic samples, variant filtering and other utility functions. Below we illustrate a SNP calling example:

```
java -jar VarScan.jar pileup2snp $pileupfile --min-coverage
$DEPTH --in-reads2 $DEPTH2 --in-avg-qual $QSCORE --output-file
$myresults.
```

where the pileup file ($pileupfile, see coverage and data extraction section) is used as input. There are also additional parameters where the user can specify to perform calling only under specific conditions. In the above example we set the calling to occur when a minimum read-depth condition (--min-coverage) is satisfied. That is, SNPs will be called only when at least $DEPTH coverage occurs. Furthermore, we set a condition that we need to have detected at least a certain number of reads for the variant allele (--in-reads2) in order to identify the site as a variant. This read-depth is $DEPTH2. The quality score of the base call can also be used as a criterion (--in-avg-qual) to call a SNP. In the example above, SNPs will be called when the base quality score is at least $QSCORE. Output (--output-file) is saved in $myresults file in tab-delimited format where SNP information is displayed.

There are many more tools in VarScan than can supply variant information on the sequenced segments (VarScan website).

Conclusive remarks

NGS methodologies have enabled the acquisition of genetic sequence information at previously unparalleled rates. However, in order to obtain meaningful and representative results of each sequenced sample the correct data analytic tools need to be employed.

In this guide, certain “good-practice” principles regarding data analysis of NGS experiments were presented and justified. Such principles were illustrated at all stages post-sequencing and examples were demonstrated using a variety of tools. Thus, the guide can be used as an introductory reading to NGS data analysis using already available tools and employing principles that will help in reaching meaningful results. However, readers should note that the exemplary tools used here are far from exhaustive and new tools that can perform analysis become available on a regular basis.

In this reading, we have focused on usage of individual tools to perform a specific task. It is good practice to become familiar with a tool's input/output relationships and identify the exact options and output that is suited for the application of interest. Once the desired options and output have been identified then one can connect all tools together in a single routine and form a bioinformatic pipeline. In such a pipeline, data output from one tool is given as input into the next tool, without the requirement of manually executing applications at each step. This process of automation frees up valuable time that can be dedicated to other experiments. It also allows large scale deployment of the analytic tools over a batch of samples without requiring human intervention or supervision. Furthermore, in this way one can deploy a pipeline at the end of one working day and have results ready by the time work resumes the next working day.

References

Barba, M., Czosnek, H. and Hadidi, A., 2014. Historical perspective, development and applications of next-generation sequencing in plant virology. Viruses, 6(1), pp.106-136.

Benjamini, Yuval, and Terence P. Speed. “Summarizing and Correcting the GC Content Bias in High-Throughput Sequencing.” Nucleic Acids Research 40.10 (2012): e72. PMC. Web. 31 Oct. 2017.

Berg, J.M. and Tymoczko, J.L., 2002. Stryer: Biochemistry. WH Freeman and Company, 5, pp.306-307.

Chen, Y.C., Liu, T., Yu, C.H., Chiang, T.Y. and Hwang, C.C., 2013. Effects of GC bias in next-generation-sequencing data on de novo genome assembly. PloS one, 8(4), p.e62856.

Cock, P.J., Fields, C.J., Goto, N., Heuer, M.L. and Rice, P.M., 2009. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic acids research, 38(6), pp.1767-1771.

Danecek, P., Auton, A., Abecasis, G., Albers, C.A., Banks, E., DePristo, M.A., Handsaker, R.E., Lunter, G., Marth, G.T., Sherry, S.T. and McVean, G., 2011. The variant call format and VCFtools.Bioinformatics, 27(15), pp.2156-2158.

GATK website, https://software.broadinstitute.org/gatk/, true on 05/10/2107.

Head, S.R., Komori, H.K., LaMere, S.A., Whisenant, T., Van Nieuwerburgh, F., Salomon, D.R. and Ordoukhanian, P., 2014. Library construction for next-generation sequencing: overviews and challenges. Biotechniques, 56(2), p.61.

Hwang, S., Kim, E., Lee, I. and Marcotte, E.M., 2015. Systematic comparison of variant calling pipelines using gold standard personal exome variants. Scientific reports, 5.

Krøigård, A.B., Thomassen, M., Lænkholm, A.V., Kruse, T.A. and Larsen, M.J., 2016. Evaluation of nine somatic variant callers for detection of somatic mutations in exome and targeted deep sequencing data. PLoS One, 11(3), p.e0151664.

Kulski, J.K., 2016. Next-Generation Sequencing—An Overview of the History, Tools, and “Omic” Applications. In Next Generation Sequencing-Advances, Applications and Challenges. Intech.

Lai, Z., Markovets, A., Ahdesmaki, M., Chapman, B., Hofmann, O., McEwen, R., Johnson, J., Dougherty, B., Barrett, J.C. and Dry, J.R., 2016. VarDict: a novel and versatile variant caller for next-generation sequencing in cancer research. *Nucleic acids research*, *44*(11), pp.e108-e108.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G. and Durbin, R., 2009. The sequence alignment/map format and SAMtools. Bioinformatics, 25(16), pp.2078-2079.

Mamanova, L., Coffey, A.J., Scott, C.E., Kozarewa, I., Turner, E.H., Kumar, A., Howard, E., Shendure, J. and Turner, D.J., 2010. Target-enrichment strategies for next-generation sequencing.

Nature methods, 7(2), pp.111-118.

Mardis, E.R., 2013. Next-generation sequencing platforms. Annual review of analytical chemistry, 6, pp.287-303.

OMIC Tools website, https://omictools.com/demultiplexing-category, true on 05/10/2017.

SAMtools documentation, http://samtools.sourceforge.net/, true on 05/10/2107.

VarScan documentation, http://varscan.sourceforge.net/using-varscan.html, true on 05/10/2017.

Wikipedia Biology Formats, https://en.wikipedia.org/wiki/List_of_file_formats#Biology, true on 05/10/2017.

Appendix

List of software used in examples of the guide:

Software Name	Description	Website
Bamtools	Process and retrieve data from aligned files	https://github.com/pezmaster31/bamtools
Bedtools	Retrieve data from aligned files	http://bedtools.readthedocs.io/en/latest/
bowtie	Perform sequence alignment to a reference genome	http://bowtie-bio.sourceforge.net/index.shtml
bwa	Perform sequence alignment to a reference genome	http://bio-bwa.sourceforge.net/
cutadapt	Perform trimming of FASTQ files	https://cutadapt.readthedocs.io/en/stable/
FastQC	Perform quality control on FASTQ and SAM/BAM files.	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
FastX	Perform quality control and other processing of FASTQ files.	http://hannonlab.cshl.edu/fastx_toolkit/
fastx_clipper	Perform trimming of FASTQ files	http://hannonlab.cshl.edu/fastx_toolkit/commandline.html#fastx_clipper_usage
GATK	Process and retrieve data from aligned files	https://software.broadinstitute.org/gatk/
MAQ	Perform sequence alignment to a reference genome	http://maq.sourceforge.net/
PICARD	Process and retrieve data from aligned files	http://broadinstitute.github.io/picard/
PRINSEC	Retrieve summary statistics from the FASTQ files.	http://prinseq.sourceforge.net/
Sambamba	Process and retrieve data from aligned files	http://lomereiter.github.io/sambamba/index.html
SAMtools	Process and retrieve data from aligned files	http://samtools.sourceforge.net/

segemehl	Perform sequence alignment to a reference genome	http://www.bioinf.uni-leipzig.de/Software/segemehl/
ShortRead	Perform quality control and trimming of FASTQ files.	http://www.bioconductor.org/packages/release/bioc/html/ShortRead.html
SOAP	Perform sequence alignment to a reference genome	http://soap.genomics.org.cn/
trim galore!	Perform trimming of FASTQ files	https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/
VarDict	Perform variant calling from aligned files	https://github.com/AstraZeneca-NGS/VarDict