# Handout for Lecture 1: Next-Generation Sequencing: Methods and Computational Analysis

Peter N Robinson

October 16, 2012

**Abstract**

For lecture #1, you should have an understanding of the differences between NGS and traditional Sanger sequencing, and you should understand the major steps of the Illumina technology and their relevance for computational analysis, without necessarily trying to memorize the primer sequences. You should understand the file formats discussed in the slides (especially FASTQ and SAM/BAM), and be able to explain the significance and meaning of the various parts of these files at the level we discussed in the lecture.

In this practicum, we will go over the major steps in processing Illumina data and will use some of the major applications. The goal is not yet to gain a detailed understanding, but rather to see the "big picture" now so that we can fill in some of the details in the course of this semester.

## 1 Introduction

The goal of this practicum, which you should try at home before coming to the practicum, is to go over all of the steps in calling variants in a re-sequenced *E. coli* genome and to visualize the result using the IGV viewer. We will explain all of the steps and will need to install several applications and download a relatively small NGS dataset from the Web. Note that I assume you are using linux or the shell on a Macintosh OSX system. If you are using Windows, you will probably need to install cygwin to complete this tutorial, but this might be a good opportunity to install a linux system (tip: Kubuntu is more comfortable than Windows for the most part and offers nearly all of the flexibility of debian linux).

This practicum was adapted from an online tutorial of the University of Texas Intro to NGS Bioinformatics Course[1].

## 2 The data

We will download Whole-genome shotgun re-sequencing of a clone from the long-term *E. coli* evolution experiment. The evolution will correspond to variants compared to the reference sequence for the *E. coli* genome, this will allow us to perform variant calling.

### 2.1 Sequence Read Archive of NCBI

We will download the data from the NCBI Sequence Read Archive[2] (SRA). Given the large size of NGS datasets, traditional file transfer methods (FTP or HTTP) do not perform well, and NCBI uses the Aspera protocol[3]. Aspera Connect is software that allows download and upload via a web plugin for popular browsers on machines running Linux, Windows, and Macintosh. The software also includes a command line tool (ascp) that allows scripted data transfer. Therefore, install the Aspera Connect plugin (on my system, this is Aspera connect 2.7.9, linux 64; you should find it easily on the NCBI SRA website or go to the URL in the footnote)[4].

This will download an installation script. Install it as appropriate for your system. On my system (kubuntu linux using firefox), the commands were

```
$ chmod +x aspera-connect-2.7.9.58060-linux-64.sh
$ ./aspera-connect-2.7.9.58060-linux-64.sh

Installing Aspera Connect

Deploying Aspera Connect (/home/peter/.aspera/connect) for the current user only.
Restart firefox manually to load the Aspera Connect plug-in

Install complete.
```

After restarting Firefox, go to the SRA website at NCBI as mentioned above and search for `SRR030257`. In case you do not find it, use the URL in this footnote.[5]. You should see a small table with a header and one row.

[1]https://wikis.utexas.edu/display/bioiteam/SSC+Intro+to+NGS+Bioinformatics+Course
[2]http://www.ncbi.nlm.nih.gov/sra
[3]http://www.ncbi.nlm.nih.gov/books/NBK47527/
[4]http://downloads.asperasoft.com/download_connect/
[5]http://www.ncbi.nlm.nih.gov/sra?term=SRR030257

In the column “Size”, you will see an entry “275.5Mb”. If you now click on this, you should be able to download the file. If this does not work, try to download it using normal ftp from this URL [6].

## 2.2 Converting from SRA format to FASTQ

Go to the main SRA website again[7] and download the SRA toolkit. You can find this by clicking on the download link and then clicking on the “Software” tab. Download the software that is appropriate for your system (I have used the precompiled binaries: `CentOS Linux 64 bit architecture`). Then move to the directory with the file downloaded as above, and enter

```
$ fastq-dump --split-files -A SRR030257.lite.sra
```

Note that you will either have to add fastq-dump to your PATH or indicate the complete path to it on the command line.

The split-files flag was used because the data were produce using paired-end sequencing an Illumina 1G Genome Analyzer, meaning that there are always two reads per sequence. The result should now be two files:

```
SRR030257.lite.sra_1.fastq
SRR030257.lite.sra_2.fastq
```

## 2.3 Assignment

Examine the fastq files. Try to understand the major components of the file. If necessary, read this article about the FASTQ format[1] or consult the lecture slides.

[6] ftp://ftp.ddbj.nig.ac.jp/ddbj_database/dra/sralite/ByExp/litesra/SRX/SRX012/SRX012992/SRR030257/
[7] http://www.ncbi.nlm.nih.gov/sra

## 3 Quality Control

An important topic in NGS analysis is quality control. We do not have time to go into this deeply, but if you have the time, I recommend the program fastqc[8]. The program is extremely easy to use and there is a good tutorial on youtube that is accessible via the program homepage (See Fig. 1 for an example).

In general, before downstream analysis, you should check your sequence data for the following things.

- per-base quality: Trim ends if quality too low
- per sequence quality: Avg quality score should be well above 20, otherwise experiment may need to be repeated
- sequence duplication levels: A large number of identical sequences may indicate duplication by PCR during library prep, this can bias estimates of mRNA expression in RNA-seq and is often filtered out
- Overrepresented sequences: Sometimes adapter sequences are represented in final sequences and these need to be filtered out before downstream analysis

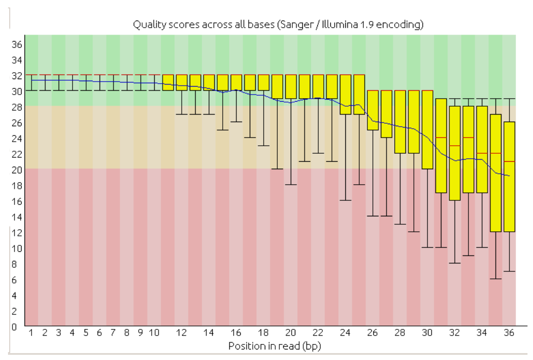


Figure 1: Quality control for the E coli dataset. Note the quality declines across the length of the read. Red: median value, blue: mean; yellow box: interquartile range (25%–75%); black whiskers: 10%–90%.

[8]http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

# 4 Mapping

The first step in most NGS analysis pipelines is to map the sequencing reads to a reference genome. There are over 50 mapping programs listed in Wikipedia, and you will be studying the algorithms implemented in some of these programs over the next several lectures.

## 4.1 The reference genome

Download the Escherichia coli B str. REL606 chromosome, complete genome, (NCBI Reference Sequence: `NC_012967.1`), from the NCBI Nucleotide database[9]. Choose FASTA format. In the following, it is assumed that the sequence you have downloaded was named `NC_012967.fa`.

## 4.2 Bowtie

We will use the bowtie read mapper[2], which can be downloaded from the project's sourceforge site[10]. I downloaded the source code version and made the binaries simply by entering `$ make` (there is no automake configuration file to be used).

The first step is now to make an index of the reference file. This file is used to speed up the alignment of short reads (the index allows quick access to subsequences of the main sequence; this will be a major topic of the following lectures). In the following, you will need to make sure the binary `bowtie-build` (which creates the index) is in the PATH or you will need to substitute the full path).

```
$ mkdir bowtie
$ bowtie-build NC_012967.fasta bowtie/NC_012967
```

With the above two commands, we first make a directory `bowtie` in which we will write the index files. We then run `bowtie-build`, which is the program from the bowtie package that creates index files. The first argument is the reference FASTA. The second argument is the "base" file name to use for the created index files. It will create a bunch of files beginning `bowtie/NC_012967*`.

To check that the index has been created properly, try the following command, which should print a single alignment and exit.

[9]http://www.ncbi.nlm.nih.gov/nuccore

[10]http://bowtie-bio.sourceforge.net/index.shtml

```
$ bowtie -c bowtie/NC_012967 GCGTGAGCT
0 + gi|254160123|ref|NC_012967.1| 2785732 GCGTGAGCT IIIIIIIII 18
# reads processed: 1
# reads with at least one reported alignment: 1 (100.00%)
# reads that failed to align: 0 (0.00%)
Reported 1 alignments to 1 output stream(s)
```

The following command executes the bowtie readmapper on the paired-end files of the *E coli* dataset (Note that the first line has been broken for better visibility)

```
$ bowtie -t --sam bowtie/NC_012967 \
      -1 SRR030257.lite.sra_1.fastq \
      -2 SRR030257.lite.sra_2.fastq \
       bowtie/SRR030257.sam
Time loading reference: 00:00:00
Time loading forward index: 00:00:00
Time loading mirror index: 00:00:00
Seeded quality full-index search: 00:02:45
# reads processed: 3800180
# reads with at least one reported alignment: 3500240 (92.11%)
# reads that failed to align: 299940 (7.89%)
Reported 3500240 paired-end alignments to 1 output stream(s)
Time searching: 00:02:45
Overall time: 00:02:45
```

## 5 SAM: Sequence Alignment/Map

SAM (Sequence Alignment/Map) format is a generic format for storing large nucleotide sequence alignments[11].

- Each line of header begins with '@' followed by a type code
- Except for `@CO` (comment) lines, each data field is `TAG:VALUE`
- The header we will produce in the practical is as follows:

```
@HD     VN:1.0  SO:unsorted
@SQ     SN:gi|254160123|ref|NC_012967.1|        LN:4629812
@PG     ID:Bowtie       VN:0.12.8       CL:"(...)"
```

- VN (version of SAM format): 1.0
- SO: sorting order of alignments
- SQ: reference sequence dictionary (here: the E coli genome)
- LN: reference sequence length (4.6 million nt)
- PG: program with version (VN) and command line (CL) arguments

All remaining lines are alignment lines, one read per line; Each alignment line has 11 mandatory fields. Let us examine the first two lines from the SAM file produced above (broken here for legibility).

```
SRR030257.lite.sra.1    99      gi|254160123|ref|NC_012967.1|   950180  255     36M     =       950295  151 \
  TTACACTCCTGTTAATCCATACAGCAACAGTATTGG    AAA;A;AA?A?AAAAA?;?A?1A;;????566)=*1    XA:i:0  MD:Z:32C3       NM:i:1
SRR030257.lite.sra.1    147     gi|254160123|ref|NC_012967.1|   950295  255     36M     =       950180  -151  \
  ACACTAAAGACAAAGAAATCACATTGACAAAGTTAA    </<+9*AA/A)AAAAAA+A9A>A9AAAAAAAAAAAA    XA:i:1  MD:Z:5G11G18    NM:i:2
```

| Col | Field | Description | Example |
|---|---|---|---|
| 1 | QNAME | query name | SRR030257.lite.sra.1 |
| 2 | FLAG | bitwise FLAG | 99 |
| 3 | RNAME | Ref. seq. name | `gi|254160123|ref|NC_012967.1|` |
| 4 | POS | 1-based leftmost mapping POSition | 950180 |
| 5 | MAPQ | mapping quality | 255 |
| 6 | CIGAR | CIGAR string | 36M |
| 7 | RNEXT | Ref. name of the mate/next segment | = |
| 8 | PNEXT | Position of the mate/next segment | 950295 |
| 9 | TLEN | observed Template LENgth | 151 |
| 10 | SEQ | segment SEQuence | TTACA... |
| 11 | QUAL | ASCII of Phred-scaled base QUALity | ¡/¡+9*AA/A)... |
| 12–? | optional | .. | |

[11] http://samtools.sourceforge.net/

### 5.0.1 QNAME

Query template NAME, i.e., the reads we are trying to align.

### 5.0.2 FLAG

Bitwise FLAG. Each bit is explained in the following table:

| Bit | Description |
|---|---|
| 0x1 | template having multiple segments in sequencing |
| 0x2 | each segment properly aligned according to the aligner |
| 0x4 | segment unmapped |
| 0x8 | next segment in the template unmapped |
| 0x10 | SEQ being reverse complemented |
| 0x20 | SEQ of the next segment in the template being reversed |
| 0x40 | the first segment in the template |
| 0x80 | the last segment in the template |
| 0x100 | secondary alignment |
| 0x200 | not passing quality controls |
| 0x400 | PCR or optical duplicate |

Thus, in our example, the first sequence has a flag of 99=0x40 + 0x20 + 0x2 + 0x1 = 64+32+2+1 in hexadecimal, meaning that we have a template with multiple segments in sequencing (paired ends), each of whose segments was properly aligned, and we have here the first sequence in the template with the sequence being reverse complemented.

### 5.0.3 RNAME

The reference sequence name in our example, `gi|254160123|ref|NC_012967.1|` comes from the NCBI file for the *E. coli* genome.

### 5.0.4 POS

The position of the leftmost base of the alignment is 950180 with respect to the reference sequence.

### 5.0.5 MAPQ

MAPping Quality. It equals $-10\log_{10}\Pr\{\text{mapping position is wrong}\}$, rounded to the nearest integer. A value of 255 indicates that the mapping quality is not available.

### 5.0.6 CIGAR

CIGAR string. The CIGAR operations are given in the following table (set '*' if un- available)

| Op | BAM | Description |
|---|---|---|
| M | 0 | alignment match (can be a sequence match or mismatch) |
| I | 1 | insertion to the reference |
| D | 2 | deletion from the reference |
| N | 3 | skipped region from the reference |
| S | 4 | soft clipping (clipped sequences present in SEQ) |
| H | 5 | hard clipping (clipped sequences NOT present in SEQ) |
| P | 6 | padding (silent deletion from padded reference) |
| = | 7 | sequence match |
| X | 8 | sequence mismatch |

Thus, in our example, we simply have 36M, i.e., all 36 nucleotides align to the reference genome at the indicated position.

Note that For mRNA-to-genome alignment, an N operation represents an intron. For other types of alignments, the interpretation of N is not defined.

CIGAR strings allow a compact representation of more complex alignments. Consider for instance, the following alignment, iin which the query (Q) sequence is aligned to the reference sequence begining at position 7 of the reference sequence. There are then 8 matches, then an insertion of 2 nucleotides in the query, then four additional matches, then a deletion of one base in the query, and then four more matches. We can represent this as `8M2I4M1D4M`.

```
AGCATGTTAGATAA--GATAGCTGG  R
      ||||||||  |||| ||||
------TTAGATAAAGGATA-CTGG  Q
```

### 5.0.7 RNEXT

Reference sequence name of the NEXT segment in the template.

This field is set as '*' when the information is unavailable, and set as '=' if RNEXT is identical to RNAME. In our example, all of these fields are '='.

### 5.0.8 PNEXT

Position of the NEXT segment in the template. Set as 0 when the information is unavailable. This field equals POS of the next segment. If PNEXT is 0, no assumptions can be made on RNEXT and bit 0x20.

In our example, PNEXT of the first sequence, 950295, is equal to POS of the next sequence.

#### 5.0.9 TLEN

signed observed Template LENgth. If all segments are mapped to the same reference, the unsigned observed template length equals the number of bases from the leftmost mapped base to the rightmost mapped base.

In our example,TLEN = 151 which corresponds to the total length spanned by both of the reads of the pair (leftmost positions of 950180 and 950295, together with length of each read being 36 nt)

#### 5.0.10 SEQ and QUAL

segment SEQuence and ASCII of base QUALity plus 33 (same as the quality string in the Sanger FASTQ format). In our example, we have for the first sequence

```
TTACACTCCTGTTAATCCATACAGCAACAGTATTGG
AAA;A;AA?A?AAAAA?;?A?1A;;????566)=*1
```

# 6 samtools: Variant calling

First download the samtools code from the sourceforge website[12]. Uncompress the archive file (`$ tar xjf samtools-0.1.18.tar.bz2`), enter the samtools directory and simply use `make` to build the program.

For simplicity, we will put all of the files needed for samtools into one directory

```
$ mkdir sam
$ cp  NC_012967.fasta sam/
$ mv bowtie/SRR030257.sam sam/
$ cd sam
```

First, we index the reference file. As above, you will need to put samtools into your PATH or adjust the command accordingly

```
$ samtools faidx NC_012967.fasta
```

`samtools faidx` indexes reference sequence in the FASTA format or extract subsequence from indexed reference sequence. If no region is specified, faidx will index the file and create `ref.fasta.fai` on the disk.

[12]http://samtools.sourceforge.net/

## 6.1 Convert to BAM file

SAM is a text file, so it is slow to access information about how any given read was mapped. BAM files are essentially gzipped versions of sam files that can be loaded more quickly. Typically, we will also want to sort the BAM files to accelerate finding reads etc., without having to search through the entire BAM file.

The relevant command is

```
$ samtools view -b -S -o SRR030257.bam SRR030257.sam
```

This command creates the file SRR030257.bam, which is 336M, as opposed to 1.4G for SRR030257.sam.

## 6.2 Sort and Index the BAM file

We can now sort and index the bam file to accelerate downstream analysis.

```
$ samtools sort SRR030257.bam SRR030257.sorted
```

This creates a new file, `SRR030257.sorted.bam`, that is 261M big.

We now index this file

```
$ samtools index SRR030257.sorted.bam
```

This command creates a 14K binary index file, `SRR030257.sorted.bam.bai`.

# 7 Variant calling

We now have aligned the reads to the E coli reference genome, and now ask if our reads contain variants with respect to this genome. These are then the positions that have underwent evolution in the experiment. Just as there are over 50 read mappers, there are quite a few different variant callers. We will use a simple caller from the samtools package for this exercise. We first create a BCF file, which is a binary version of the Variant Call Format (VCF) file that we will examine in greater detail below.

```
$ samtools mpileup -u -f NC_012967.fasta SRR030257.sorted.bam > SRR030257.bcf
```

The result is 413M file, `SRR030257.bcf`

To actually view it, we need to use the program `bcftools` (which is included in the directory of the same name from samtools)

```
 bcftools view -v -c -g SRR030257.bcf > SRR030257.vcf
```

The VCF file, `SRR030257.vcf`, is only 34K and can be examined in emacs or another editor.

# 8 Visualizing Genomic Analysis Results

It is important to inspect the results of genomic analysis by the old-fashioned eyeball-o-meter to avoid nasty surprises. There are a number of different tools to do this, but perhaps the most well known is IGV: the Integrative Genomics Viewer from the Broad Institute [3, 4]. This program has an enormous number of features, and we will barely scratch the surface here. In essence, we will use IGV to view the alignments and the variants against the backdrop of the E coli reference genome. There are several steps to set up the analysis.

Download the IGV from Broad[13], and start the program as appropriate for your system.

## 8.1 Load data into IGV

We can now load the genome sequence (fasta) and the sort BAM reads by using the `File|Load from file...` menu. It is extremely instructive to compare the VCF file with the visualization of the alignment in IGV. For instance, compare the following variant call and the ICF visualization in Fig. 2.

```
 gi|254160123|ref|NC_012967.1| 555154 . A C 151 .\
DP=27;VDB=0.1016;AF1=1;AC1=2;DP4=0,0,7,17;MQ=20;FQ=-99 GT:PL:GQ \
1/1:184,72,0:99
```

Thus, we can say that the variant at position 555154 is well supported by the evidence. On the other hand, consider the following call (Fig. 3) is poorly supported.

```
 gi|254160123|ref|NC_012967.1| 555660 . T C 25\
DP=3;VDB=0.0158;AF1=1;AC1=2;DP4=0,0,1,2;MQ=20;FQ=-36 GT:PL:GQ\
1/1:57,9,0:15
```

It pays to explore the data using different vislualizations.

## 8.2 Assignment

Examine 5 different variants in the IGV. Make sure you understand the relationship between the DP4 field and the visualization in IGV. Consider intuitively the relation between the QUAL score and the visualization of the mutaiton in IGV.

[13]http://www.broadinstitute.org/igv/

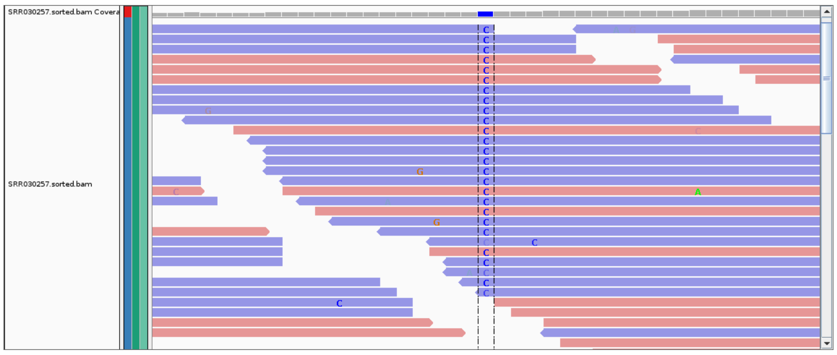


Figure 2: Visualization of mutation at position 555154, ref=A, alt=C, DP4=0,0,7,17. This means there were no reads supporting the reference sequence, but there were 7 forward reads (read) and 17 reverse reads (blue) supporting the ALT call.

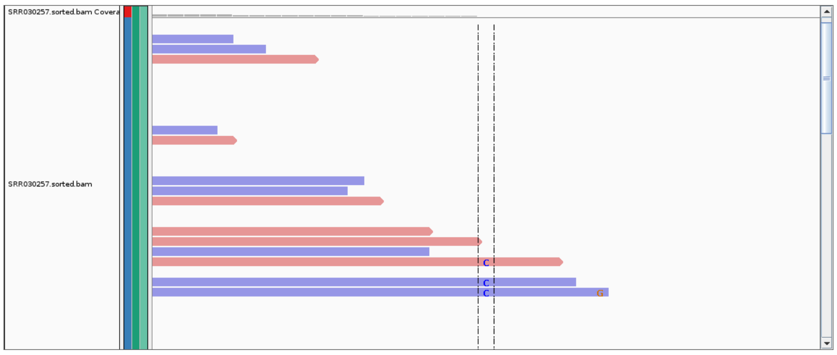


Figure 3: Visualization of mutation at position 555660, ref=T, alt=C, DP4=0,0,1,2.

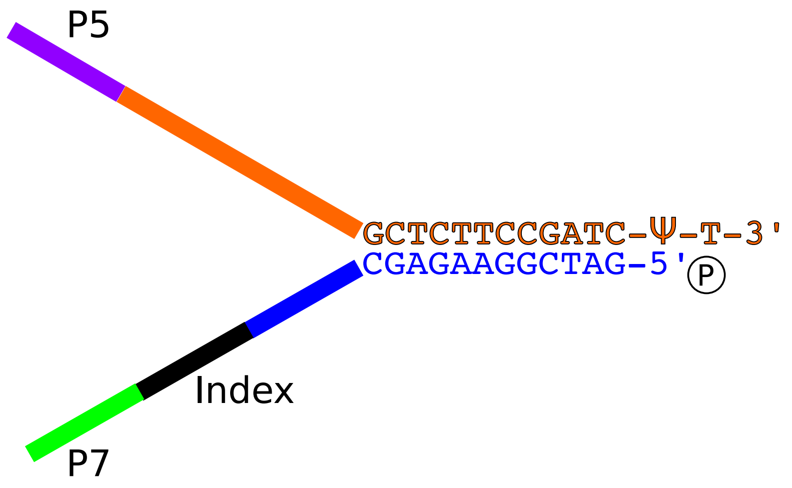
## 9 Illumina Primers and Adapters

It is not trivial to understand the various primers and adapters from all of the Illumina kits, which differ in some details depending on the type of experimentn being performed (single end, paired-end, etc). We provide a summary of the material discussed in the lecture here for reference.

**TruSeq Universal Adapter (P7)**
5 AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT 3

**TruSeq Indexed Adapter (P5)**
5 GATCGGAAGAGCACACGTCTGAACTCCAGTCAC–NNNNNN–ATCTCGTATGCCGTCTTCTGCTTG 3

The indexed adapter (IA) has 6 variable nucleotides ("N"), which are used for the **bar code**. The various segments of the primer sequences play several roles in the sequencing process. They are first combined to form the **forked adapter**



The twelve 3' nucleotides of the Universal Adapter (UA; P7) and the twelve 5' nucleotides of the indexed adapter (IA; P5) are reverse complementary to one another and can thus **anneal**. Note the C and T at the very 3' end of the UA are connected by a phosphothiorate (Ψ) bond, which is resistant to exonuclease activity. This T is of course necessary to bind to the 'A' overhang of the ligated fragments

The important features of the forked adapter are

- One adapter only (Two partially annealed oligos via 12 bp reverse complementary sequence)

- A single base 'T' overhang to encourage correct ligation
- A 5' phosphate on the bottom strand to ensure double-stranded ligation

This design enables the following for a single index adaptor run on a standard Illumina HiSeq or MiSeq. The procedure of library prep creates an amplicon with the following structure (shown as the top, 5'-3', strand:

```
<P5 primer/capture site><Read1 primer site> <INSERT><Read2 primer site> \
    <IndexRead1><P7 primer/capture site>
```

Here is the sequence of the TrueSeq Universal Adapter

**TruSeq Universal Adapter (P7)**
5'-AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT-3'

It contains a P5 PCR primer/flowcell capture site: 5'-AAT-GAT-ACG-GCG-ACC-ACC-GAG-A-3' (22nt) as well as the standard TruSeq Read 1 primer site: 5'-TCT-ACA-CTC-TTT-CCC-TAC-ACG-ACG-CTC-TTC-CGA-TCT-3' (36nt).

This is followed by the **insert** to be sequenced.

Consider now the sequence of the Indexed Adapter.

**TruSeq Indexed Adapter (P5)**
5'-GATCGGAAGAGCACACGTCTGAACTCCAGTCAC–NNNNNN–ATCTCGTATGCCGTCTTCTGCTTG-3'

Read2 primer site: Then the Index read primer site: 5'-A-GAT-CGG-AAG-AGC-ACA-CGT-CTG-AAC-TCC-AGT-CAC-3' (33+1nt; note the initial A is from the dA tailing of the insert and is not included in the index primer or adaptor sequences).

The reverse-complement of the read2 primer site is the Read 2 sequencing primer, but the Read 2 sequencing primer includes the T corresponding to the dA insert tail so sequencing starts with the insert.

IndexRead1 comprises a 6 bp "barcode" sequence that can be used to combine several samples in the same lane but still be able to identify the provenance of the samples following sequencing (several barcodes are shown in the table; with a few exceptions, the bar codes are separated from one another by a Hammod distance of at least 3).

| Sequence | TruSeq name |
|---|---|
| ATCACG | TSBC01 |
| CGATGT | TSBC02 |
| TTAGGC | TSBC03 |
| TGACCA | TSBC04 |
| ACAGTG | TSBC05 |
| ... | ... |

Finally, the sequence concludes with the P7 PCR primer/flowcell capture site: 5'-ATC-TCG-TAT-GCC-GTC-TTC-TGC-TTG-3' (24nt).

For the library prep PCR, the following primer ("P7")

5'-CAA-GCA-GAA-GAC-GGC-ATA-CGA-GAT-3' (24 nt) is complementary with the 24 nucleotides on the 3' end of the 5'-3' strands of the ligation products. In the first cycle of PCR, it produces a reverse complementary strand. Starting from the second PCR cycle, the other PCR primer ("P5"), can bind. Its sequence is 5'-AAT-GAT-ACG-GCG-ACC-ACC-GAG-ATC-TAC-ACT-CTT-TCC-CTA-CAC-GA-3'

Finally, the sequencing primer is equivalent to the 3' portion of the sequence of the UA (i.e., excluding the portion that binds to the flow cell

```
UA: 5'-AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT-3'
Seq-primer:              5'-TCTACACTCTTTCCCTACACGACGCTCTTCCGATCT-3'
```

## References

[1] Peter J A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Res*, 38(6):1767–1771, Apr 2010.

[2] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[3] James T. Robinson, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S. Lander, Gad Getz, and Jill P. Mesirov. Integrative genomics viewer. *Nat Biotechnol*, 29(1):24–26, Jan 2011.

[4] Helga Thorvaldsdóttir, James T. Robinson, and Jill P. Mesirov. Integrative genomics viewer (IGV): high-performance genomics data visualization and exploration. *Brief Bioinform*, Apr 2012.