Guide to Bioinformatics tools and resources

Matthew Welland

January 6, 2015

File Types

BAM

- A Binary version of a SAM file, storing the same data in a more concise format
- BAM is used in most production pipelines, but SAM may be used where interconversion with other applications is crucial

Both of the file types contain the same actual data, but the overall space the data takes up and the format differs between the two.

Sam files contain an optional header section:

- HD - Header
- SQ - Seqence Dictionary
- RG - Read Group
- PG - Program
- CO - Comment

This is followed by an alignment section; multiple tab-delimited lines with each line describing an alignment.

CIGAR

CIGAR strings are used to show how well a read is paired to the appropriate reference sequence. This is a String composed of base lengths and the operation describing how the lengths indicate similarities or differences between the sequences. The operations are used to show:

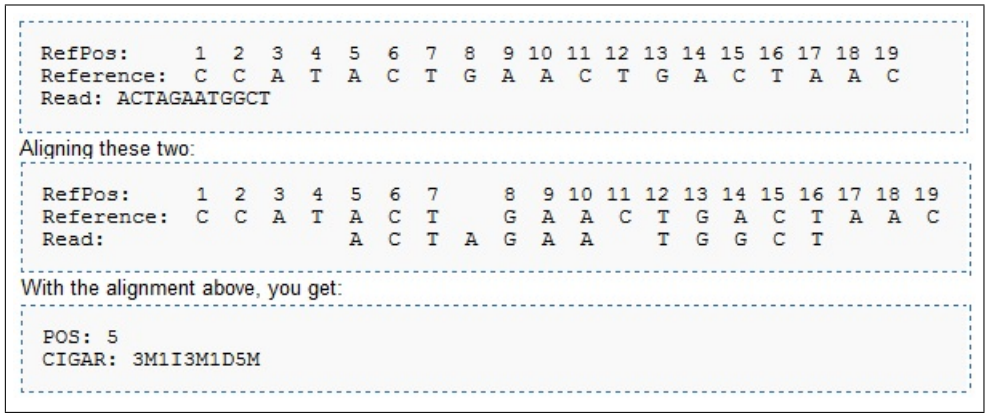

Figure 1: The CIGAR will indicate a start position (where the lengths and operations begin to correspond to the reference sequence) and a sequence of relations

- mismatches/matches
- lengths of bases relating to the operation
- deletions from the read
- insertions in the read

CRAM

CRAM is a potential successor to BAM - a compressed version of the allignment.

1. CRAM starts with a file definition followed by the BAM header and other containers
2. Each subsequent container starts with a container header followed by 1 or more block
3. Each block starts with a block header; all real data comes after this header
4. The first block in each container is the compression header block

5. Each subsequent block is organised into slices. A slice may contain a contiguous region of alignment data. Each slice has a slice header block followed by 1 or more data blocks

6. Each data block is divided into a mandatory core data block and optional external data blocks

FastA

A text-based format for representing either nucleotide or protein sequences, using a single-character encoding for each atomic component. This is a super simple format, making it simple to parse with any scripting language.

The first line of each FastA entry begins with a '>' character, followed by the description line. There shouldn't be a space between the '>' and the start of the identifier. The sequence ends if another line beginning with '>' appears, indicating the start of another sequence. Lines are typically limited to 80 characters in length. Several single-sequence Fasta files can be concatenated to display a multiple sequence file, with each new entry starting with a'>' and a description line.

FastQ

A text based format for storing a biological sequence and its corresponding quality scores. This format typically use four lines per sequence:

1. Begins with a '@' character and is followed by a sequence identifier and an optional description (similar to a title line)

2. Raw sequence letters

3. Begins with a + character and is **optionally** followed by the sequence identifier

4. Encodes the quality values for the sequence in line 2, and must contain the same number of characters as there are letters in the sequence

The character '!' in the quality String indicates the lowest quality, whilst '~' indicates the highest.

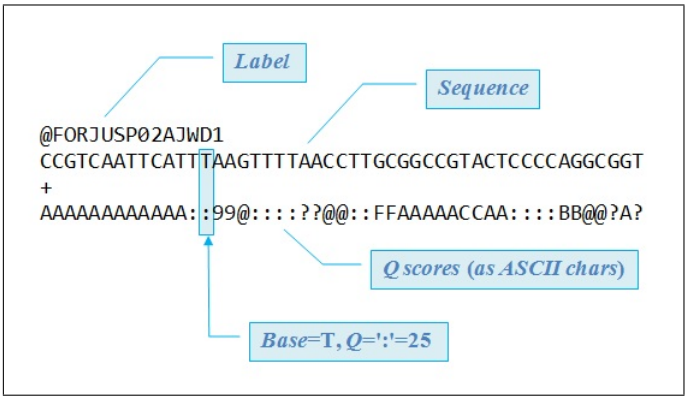

Figure 2: An annotated FastQ file example, showing the raw sequence and the corresponding quality values

SAM

- SAM stands for Sequence Alignment Map
- A text file format for storing sequence data in a series of tab-delimited ASCII files
- SAM files are generally created by aligners which read a FastQ file and assign sequences to a position within a reference genome

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

Figure 3: An example of a SAM format file, showing the tab-delimited columns containing data

PHRED

PHRED is a quality score, originally developed during the Human Genome Project. Phred quality scores Q are defined as a property which is logarithmically related to the base-calling error probabilities P:

$$Q = -10log_{10}P \tag{1}$$

This means that a Phred score of 3 indicates that the probability of the base having been called incorrectly is 1 in 1000. The most common method of using Phred is to disregard any bases which do not have a quality score of at least 20.

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

Figure 4: A table describing the relationship between Phred score and accuracy

BED

The Browser Extensible Data (BED) file format consists of one line per feature, each containing 3-12 columns of data. The first three fields are mandatory;

1. chrom: name of the chromosome or scaffold. Chromosome names can be provided with or without the 'chr' prefix

2. chromStart: start position of the feature in standard chromosomal coordinates

3. chromEnd: end position of the feature in standard chromosomal coordinates

There are nine additional fields which can be used to supply further information:

4. name: label to be displayed under the feature
5. score: a score between 0 and 1000
6. strand: '+' for forward, '-' for reverse
7. thickStart: field used by UCSC drawing code (not needed in Ensembl)
8. thickEnd: as thickEnd (?)
9. itemRgb: an RGB colour value for display
10. blockCount
11. blockSizes
12. blockStarts

Track definition lines can be used to configure the display further, e.g. by grouping features into separate tracks. The 'Track line' begins with the word "track", followed by a range of space-separated key-value pairs.

BED file start positions are zero-based, whilst end positions are one-based. This means that to identify the length of a specified segment, the values only have to be subtracted directly. This is different to some other formats (such as GFF or VCF), but the change in numbering is accounted for in all tools which manipulate this file type.

Variant Call Format

The Variant Call Format (VCF) specifies the format of a text file used in bioinformatics for storing gene sequence variations. This standardised format allows for improved interoperability between NGS tools, in the same way that the SAM/BAM specification revolutionised alignment tools. The VCF file format is text based, making it easy to generate and parse, and includes extensible meta-data passages alongside the mandatory header information.

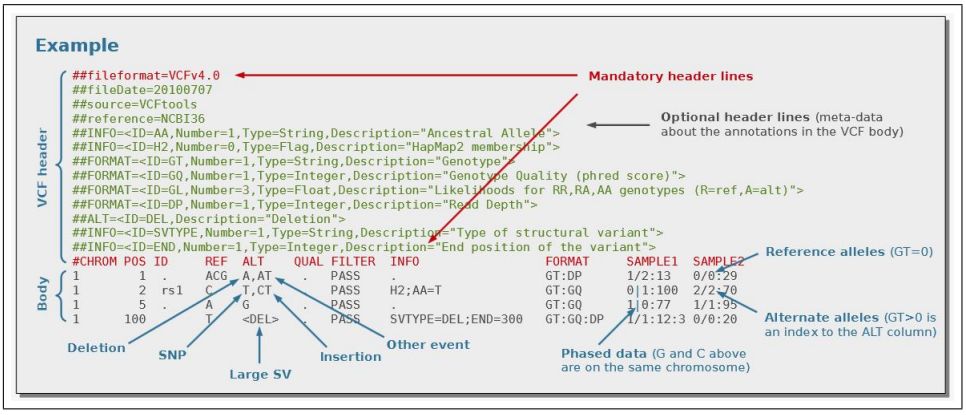

Figure 5: A demonstration of the VCF file format, labelling different structural features

Tools

FastQC

FastQC is a tool created by the Babraham institute which allows a quick analysis of the quality of sequence data encoded in input files. As huge amount of sequence data can be produced during a sequencing run, the overall quality of the data should be examined before conclusions are drawn. Most sequencers will generate a QC report as part of their analysis pipeline, but this is usually only focused on identifying problems which were generated by the sequencer itself. FastQC aims to provide a QC report which can spot problems which originate either in the sequencer or in the starting library material. This application allows users to:

- Import of data from BAM, SAM or FastQ files (any variant)
- Providing a quick overview to tell you in which areas there may be problems
- Summary graphs and tables to quickly assess your data
- Export of results to an HTML based permanent report
- Offline operation to allow automated generation of reports without running the interactive application

SAMTools

A set of utilities for interacting with and post-processing short DNA sequence read alignments in SAM, BAM, and CRAM formats, typically generated by short read aligners such as BWA. This toolkit provides utilities such as

sorting and indexing, as well as more complex variant calling and alignment. SAMtools allows a user to work directly with the SAM file format without having to decompress the contents.

A range of the SAMtools commands are listed on both the Wikipedia page and readme. The SAMtools tools are used via a command line, and UNIX style piping and stream processing are possible using standard syntax.

BEDTools

BEDtools is a toolset for genomic arithmetic; set theory on the genome. This enables users to intersect, merge, count, complement and shuffle genomic intervals from multiple files in formats such as BAM, BED & VCF. Combining multiple BEDtools operations can allow production of complex results. There are a wide range of tools included in the BEDtools package with such varied functions as changing file type, finding overlapping windows within a genomic interval, and creating an HTML page of links to UCSC locations.

The genome features which can be recognised, manipulated and interrogated using BEDtools can be genes, polymorphisms, SNPs, INDELs or structural variants. A laboratory may also incorporate novel findings into the data set as custom annotations. The basic attributes of genome features are:

- the scaffold or reference on which the feature is found
- the base pair where the feature starts
- the base pair on which the feature ends
- the strand (forward '+' or reverse '-')
- the name of the feature (where appropriate)

BEDtools also allows command line piping of commands, simplifying the process of creating a large chain of operations.

BEDOPS

An open-source command-line toolkit that performs efficient and scalable Boolean and set operations, statistical calculations, archiving, conversion, and other genomic management.

http://bedops.readthedocs.org/en/latest/content/overview.html

BEDops also includes a novel lossless format called 'Starch' which is capable of reducing file sizes of whole-genome BED files to ≈5% of original, and BAM files to around ≈35% of original, whilst adding useful metadata and random access.

IGV

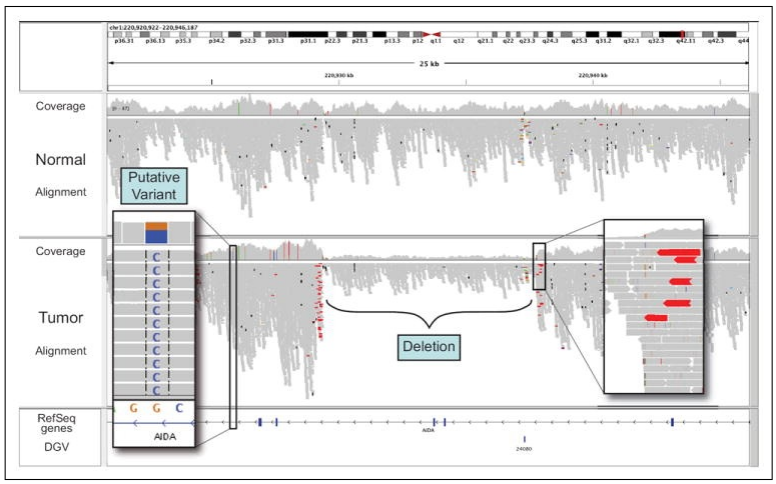

Figure 6: An image of IGV showing a sequence alignment at 20Kb resolution

The Integrative Genomics Viewer (IGV) is a response to the increasing presence of genome-wide data available by modern molecular genetic methods. This is a light-weight visualisation tool created at the Broad Institute which enable real-time exploration of diverse, large-scale genomic data sets including:

- Aligned Sequence Reads
- Mutations
- Copy number variations
- RNA interference screens
- Gene Expression

- Methylation
- Genomic annotations

This tool permits simultaneous visualisation of arbitrarily large data sets over all resolution scales, using efficient multi-resolution file formats. Whilst there are multiple genome browsers currently available, IGV focuses on the emerging integrative nature of genomic studies, placing equal emphasis on array and NGS platforms, and including clinical attributes and metadata.

Annovar

Variant Effect Predictor

The Variant Effect Predictor (VEP) is a suite of tools used to determine the likely impact of variants found within biological sequences (SNPs, insertions, deletions, CNVs or structural variants). By inputting the coordinates of variants and the nucleotide changes, this suite can be used to find:

- genes and transcripts affected by the variants
- location of the variants (e.g. upstream of a transcript, in coding sequence, in non-coding RNA, in regulatory regions)
- consequence of your variants on the protein sequence (e.g. stop gained, missense, stop lost, frameshift)
- known variants that match yours, and associated minor allele frequencies from the 1000 Genomes Project
- SIFT and PolyPhen scores for changes to protein sequence

All of the options available in the VEP suite can be selected using a graphical interface or a set of command line options. The VEP tool set can be downloaded, and a cache of local variant and reference sequence data can be stored to cut down on the network traffic created during variant identification. NOTE: The cache files required to run VEP locally are a few GB in size, download may take a long time (and if done via CMD may seem to hang for up to half an hour). Available identifiers include:

- Gene symbol - add the gene symbol for the gene to the output. This will typically be, for example, the HGNC identifier for genes in human.
- CCDS - add the Consensus CDS transcript identifier where available.
- Protein - add the Ensembl protein identifer (ENSP).
- Uniprot - add identifiers for translated protein products from three UniProt-related databases (SWISSPROT, TREMBL and UniParc).
- HGVS - generate HGVS identifiers for your input variants relative to the transcript coding sequence (HGVSc) and the protein sequence (HGVSp).
- Find co-located known variants - report known variants from the Ensembl Variation database that overlap with your input.
- The VEP can also report minor allele frequency (MAF) data for existing variants from two major genotyping projects, the 1000 Genomes Project and the NHLBI-ESP

The VEP makes use of a range of data formats, and can additionally use other formats such as VCF directly as input. The default input format is whitespace-separated (space or tab characters):

- Chromosome - just the name or number, with no 'chr' prefix
- Start coordinate
- End coordinate
- Allele - pair of alleles separated by a '/', with the reference allele first
- Strand - defined as + (forward) or - (reverse).
- Identifier (optional) - this identifier will be used in output. If not provided, the VEP will construct an identifier from coordinates and alleles.

Flags can be used to request specific output types using the VEP; JSON, VCF, or tab-delimited text output. VEP can also publish results in an HTML page format, incorporating descriptive charts and other details which may be of use.

```
1   881907    881906    -/C   +
5   140532    140532    T/C   +
12  1017956   1017956   T/A   +
2   946507    946507    G/C   +
14  19584687  19584687  C/T   -
19  66520     66520     G/A   +    var1
8   150029    150029    A/T   +    var2
```

Figure 7: An example of valid input lines for the VEP

VEP CMD Example

This section shows a few standard uses of the VEP on the command line, using common options.

- perl variant_effect_predictor.pl -i example_GRCh38.vcf –cache

This will run the VEP on the default example file, containing 1000 Genomes Project variants. The default output path is variant_effect_output.txt, if this file already exists, the –force_overwrite option can be used. the –quiet option reduces the progress output, only printing warnings and errors.

- head variant_effect_output.txt

This is just a quick head read command to view the output created through the previous set of processes. This will show the metadata lines included at the top of the file, indicated by "##"

- perl variant_effect_predictor.pl -i example_GRCh38.vcf –cache –force_overwrite –sift b

This sequence includes the –sift b command, which tells the VEP to conduct a SIFT analysis on the variants included in the VCF.

- perl filter_vep.pl -i variant_effect_output.txt -filter "SIFT is deleterious" — head -n15

Use of VEP's filtering script can narrow down this set of results to only show those in which the output as determined by SIFT as deleterious.

- perl variant_effect_predictor.pl -i example_GRCh38.vcf –cache –force_overwrite –sift b –canonical –symbol –fields Uploaded_variation,SYMBOL,CANONICAL,SIFT -o STDOUT — perl filter_vep.pl -format vep -filter "CANONICAL is YES and SIFT is deleterious"

This (huge) command is an example input which executes a standard VEP process with SIFT calculations, selects specific fields from that process to display, and then passes the output through a specific VEP filtering script to extract only the entries which satisfy the specific conditions listed. In this example, this shows only variants which have a deleterious effect on canonical gene transcripts, and shows the symbols for the appropriate genes.

These examples were taken from the Ensembl docs on VEP.

Bioconductor

GATK

Aligners and Algorithms

Burrows-Wheeler Aligner

Human Reference Genome Versions

Databases

Types of database

Each entry a separate section?

Flat file

A flat file database is a very token addition to this list; simply put it is a document containing data, or a location where unconnected documents can be stored. It will not usually contain any form of intelligent search, version control or other features common among specifically designed database implementations.

Object Database

An object database is a database management system in which information is represented in the form of objects (similar to the concept of object oriented programming (OOP)). By combining database and OOP-language capabilities a system can use a continuous model for data representation in the storage medium and the programming language used to control it. An example would be creating a Java object, setting its attribute values, and saving the object as an entry within the database. A join query would not be required to fully recreate the object, as a pointer could be used to identify the location of the complete set of values instead.

Class methods which are associated with the object within the programming language can be used to interact with objects and attributes. Some Object Database Management Systems (ODMSs) allow for version control

of the objects, and can show good performance when the objects stored are huge (compared with a field-by-field relational reconstruction of an entity).

Relational Database

A relational database is a data storage facility which makes use of Edgar Codd's relational model. This sorts data into one or more tables with a unique key labelling each row. The terminology within the relational model refers to each table as a relation, the fields or columns as attributes, and each row as a tuple.

- Each entity type to be modelled within the database is typically modelled by a table (relation), and each rown (tuple) represents an instance.
- Complex entities and relationships can be represented by joining entities represented by several tables - unique row identifiers can be used to select specific instances.
- Each attribute in each table will have a specific domain; a range of values which it may take. An attribute may be specified as an integer, a Boolean value, or a String for instance.
- Constraints may also be added to the field domain, such as ensuring certain fields are not left blank, or providing a range of values an attribute must fall between.
- Foreign keys are the method of linking between tuples in different tables, and indexes may be used to speed up queries executed on different tables. Indexes are often implemented as trees for their flexibility.
- Relational operations such a Joins, Intersections, Differences and Full joins can be used to create sets of records gathering all the details of an entity or a concept.
- Normalization is a logical process which can be applied to relational database tables. This involves a progressive decomposition of a table into several single tables to ensure that no conflicting relationships exist between the attributes, and to make sure that insertions or deletions to any given field will not cause errors elsewhere in the database.

Graph Database

This type of database implements graph structures with nodes, edges, and properties to represent and store data.

- Every element contains a direct pointer to its adjacent elements and no index lookups are necessary.
- Each entity is represented by a node (Person, place, concept...), and each node has a range of descriptive properties (age, value, location...).
- Edges connect nodes, and represent the relationship between objects. This relationship can have a descriptive name and attributes. Most of the relational information in the database is stored in/on the edges.
- Each node has an arbitrary number of edges. Compared with Object or Relational databases they may be capable of modelling real world situations more faithfully, and perform better for associative data sets.
- Operations such as computing the shortest path between objects via existing relationships is simple using the Graph schema.

NoSQL

Not only SQL (NoSQL) refers to a range of non-relational database methodologies. An example is a document based database (such as PLONE), where individual documents are uploaded and can be tagged with specific identifiers to enable searching across a range of items to select those relating to the chosen properties.

Normalization

For relational databases, normalisation is a process of reducing redundancy within a database and ensuring that data integrity is implicitly maintained

during updates and additions. An example of this is the repetition of a single field value when referring to multiple rows. e.g.:

RecordID	UserID	Name	Role
1	1	Matt	DB Analysis
2	1	Matt	Coffee making
3	2	Hagrid	Yodelling

In this database example, the UserID and the Name both refer to the same person, and the data is repeated across multiple rows. This means that there is a possibility of updating one of the 'Matt' records and altering the Name field, or inserting a new record with the wrong name/ID combination, corrupting the data the table represents. To ensure that this is not a risk, the single table can be split into two tables, one containing UserID : Name relationships, and the other containing UserID : Role relationships.

Although this is a simple example, the same logic applies when examining larger tables. If a single record consists of groups of data which each depend on the primary key, but are not linked together, separation of the single record into distinct tables can improve security against update and expansion problems.

Other

Reciever Operator Characteristic