Getting Started with Genomic Data

Getting started with genomic data

Authors: Ben Wright, Wojciech Lason

Session aims

- Introduce common file formats and their uses
- Introduce command line tools to work with those files
- Put bash and R skills to use on some of these files

The intention is not to teach the specifics of every format and every tool, but to make you familiar with the landscape of bioinformatics tools and data so if you need to solve a specific problem in the future you have an idea of where to start.

Text formats

Many data files used in genomics are in 'plain text' format. They are ordinary text with spaces or tabs used to separate columns. The latter is often referred to as `.tsv` format, although it's seldom encountered with that actual file suffix. The advantage to plain text is that you can read it on the command line or load it into a spreadsheet programme easily. The downside is that this is not generally an efficient storage format and leads to large file sizes. It's not at all uncommon to work with text files so large that common text editors struggle to open them, even on modern computers.

While it's entirely possible to manipulate these files using the standard command line tools like `cut` or `awk`, you probably don't actually want to do that if you can help it. There are better tools available, designed to work with specific types of file.

Binary formats

Many text formats have an equivalent 'binary' format. This holds the same data, but not in plain text you can read in a terminal or editor. Binary formats are more efficient storage, but require specific tools to read and work with. These tools always have a function to convert back into plain text format so you can inspect the data.

Compression

For large files, it's common to work with a compressed version of the file. This doesn't just help reduce the space needed to store the files. On modern computers, reading a compressed file from disk and decompressing it is faster than reading the same uncompressed data from disk (almost always). On the command line, you usually can't work with compressed files directly, even compressed text files. You need to decompress the file using a suitable tool then use pipes (`|`) to the command you wanted. If you want to save a compressed version of a file, you need to pipe the output to a command that compresses the data before sending it to a file.

You have probably encountered the common '.zip' compression format before. This format, and other compressed formats like it, have a drawback when it comes to working with genomic data: you have to decompress the entire file even when you only want a small part of it. When genomic data is compressed, it is often done using a 'block compression' algorithm. What block compression does is slice the data up into

blocks and compress each block separately, before saving them in the same file. Therefore, you only need to decompress the block that holds the data you are interested in. This makes accessing random parts of a large, compressed file much more efficient.

Index files

Command line tools still need information about where they need to look in block compressed formats. It's very common for compressed genomic data to be accompanied by a separate 'index' file. This index file is generated from the main compressed file, and just contains information on what is stored in each block. It's this index that lets a tool know where in the block compressed file it needs to look. In genomic data, which usually has some positional information in terms of chromosomal position, this is very convenient.

You commonly need to run a separate command to create an index after you have created a compressed file.

File headers

Depending on the format, these files may or may not have a header row that names the columns. Where they do not, this is because the columns, and their order, are defined precisely in the format specification.

It is also common for these files to have a header section before the table itself. This is used to store meta-data about the rest of the file. This *could* be kept in a separate file, but that makes it too easy for the two files to become separated. Many tools use information in this header to make sure they are interpreting the data correctly.

As these headers are at the start of the file, it is simple to view them from the command line, allowing you to check this metadata for yourself to make sure it is what's expected.

Versioning

Many data files will be created with reference to a specific build of the reference genome for an organism. It's vital to avoid mixing up these versions between different steps of data processing. Even if the organism is the same, and the sequence very similar, the positions of features such as genes can vary considerably and this can be a source of errors.

Common genomics file formats

It's not expected that you have all the details of file formats committed to memory. It *is* useful to be able to recognise most common ones and know where to look to find the detailed information you might need.

Fuller descriptions of these formats can be found here.

BED format

BED format definition

- Suffix: `.bed`
- Header: none
- Uses: defining regions of interest (such as targetted sequencing kits), output from feature detection algorithms
- Acquired from: downloads of sequencing kit targets, databases of genomic features

This format defines genomic regions as simply as possible. Chromosome identifer (or equivalent), start position and end position. It can optionally have extra columns with additional information about those regions, such as strand or defining sub-features.

BED example

```
chr7    127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7    127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7    127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7    127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7    127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7    127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7    127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7    127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7    127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
```

GFF3 format

GFF3 format definition

- Suffix: `.gff3`, `.gff` or `.gtf`
- Header: `##` comments providing context for the file such as version numbers and region identifiers
- Uses: defining genomic features such as genes and exons
- Acquired from: downloads of gene/exon/feature definitions

This set of related formats also defines genomic regions, like `.bed` format, but has additional required columns with information regarding feature type, strand and phase.

GFF3 example

```
##gff-version 3.1.26
##sequence-region ctg123 1 1497228
ctg123 . gene            1000  9000  .  +  .  ID=gene00001;Name=EDEN
ctg123 . TF_binding_site 1000  1012  .  +  .  ID=tfbs00001;Parent=gene00001
ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00001;Parent=gene00001;Name=EDEN.1
ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00002;Parent=gene00001;Name=EDEN.2
ctg123 . mRNA            1300  9000  .  +  .  ID=mRNA00003;Parent=gene00001;Name=EDEN.3
ctg123 . exon            1300  1500  .  +  .  ID=exon00001;Parent=mRNA00003
ctg123 . exon            1050  1500  .  +  .  ID=exon00002;Parent=mRNA00001,mRNA00002
ctg123 . exon            3000  3902  .  +  .  ID=exon00003;Parent=mRNA00001,mRNA00003
ctg123 . exon            5000  5500  .  +  .  ID=exon00004;Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . exon            7000  9000  .  +  .  ID=exon00005;Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . CDS             1201  1500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             3000  3902  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             5000  5500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             7000  7600  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             1201  1500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             5000  5500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             7000  7600  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             3301  3902  .  +  0  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             5000  5500  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             7000  7600  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             3391  3902  .  +  0  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             5000  5500  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             7000  7600  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
```

FASTA format

FASTA format definition

- Suffix: `.fasta` or `.fa`

- Header: none
- Uses: reference genomes for specific organisms, spike-in proteins and so on
- Acquired from: on-line repositories of reference genomes, Sanger sequencing

Rather than a table, a FASTA file is a series of lines of data. This data can be nucleic acid codes but sometimes amino acid codes. Each sequence has an identifier and can run to several lines.

FASTA example

```
>sequence A
ggtaagtcctctagtacaaacacccccaatattgtgatataattaaaattatattcatat
tctgttgccagaaaaaacacttttaggctatattagagccatcttctttgaagcgttgtc
>sequence B
ggtaagtgctctagtacaaacacccccaatattgtgatataattaaaattatattcatat
tctgttgccagattttacacttttaggctatattagagccatcttctttgaagcgttgtc
tatgcatcgatcgacgactg
```

FASTQ format

FASTQ format definition

- Suffix: `.fastq`
- Header: none
- Uses: inputs for mapping and other tools, deposition in on-line biodata repositories, for a single sample
- Acquired from: sequencing services or machines directly

A FASTQ file contains a series of lines of data, similar to the FASTA format. Each sequence fragment has nucleic acid codes, annotated with quality scores, and an identifier for that fragment. FASTQ files are usually stored compressed (`.fastq.gz`).

FASTQ example

```
@EAS54_6_R1_2_1_413_324
CCCTTCTTGTCTTCAGCGTTTCTCC
+
;;3;;;;;;;;;;;;7;;;;;;;88
@EAS54_6_R1_2_1_540_792
TTGGCAGGCCAAGGCCGATGGATCA
+
;;;;;;;;;;;7;;;;;-;;;3;83
@EAS54_6_R1_2_1_443_348
GTTGCTTCTGGCGTGGGTGGGGGGG
+EAS54_6_R1_2_1_443_348
;;;;;;;;;;;9;7;;.7;393333
```

SAM format

SAM format definition

- Suffix: `.sam`
- Header: `@` lines contain extensive information about the data fields in the main file, the tools and files used to generate them
- Uses: input to variant callers, output from alignment tools, archival for sequencing projects where space is not an issue
- Acquired from: alignment tools

This format contains most of the information in a FASTQ file, but also contains information as to where in the genome the sequence fragments map to, how well they map there, and how they differ from the reference for that region.

It is possible to reconstruct a FASTQ file from a SAM file.

SAM example

```
@HD VN:1.6 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
```

BAM and CRAM formats

A `.bam` file is the binary equivalent of a `.sam` file, for much more efficient storage. This is important because `.bam` files are generally the largest working files you will encounter (and their `.sam` equivalents would often be unworkably so).

A `.cram` file is similar, but uses a more finely-tuned compression algorithm to be even more efficient. It hasn't yet achieved the same level of adoption as the BAM format, and you should check that the tools you intend to use support it.

VCF format

VCF format definition

- Suffix: `.vcf`
- Header: `##` lines contain information about the sample, how the sequence was generated, and information about the content of columns in the file. The `#` line is the header for the table.
- Uses: showing variants for a sample or comparing variants between samples
- Acquired from: the output of a variant calling pipeline, downloads from variant databases

A format for storing variant information such as single nucleotide variants, short insertions and deletions, and other sequence variations, for one or more samples. It only stores differences in a particular sample from the supplied reference, and hence is smaller than full sequence data. It has 8 columns containing information about the variants, then 1 or more columns showing genotypes, 1 column per sample.

VCF example

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
```

```
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA00001 NA00002 NA00003
20 14370 rs6054257 G A 29 PASS NS=3;DP=14;AF=0.5;DB;H2 GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43
20 17330 . T A 3 q10 NS=3;DP=11;AF=0.017 GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3
20 1110696 rs6040355 A G,T 67 PASS NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0
20 1230237 . T . 47 PASS NS=3;DP=13;AA=T GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20 1234567 microsat1 GTC G,GTCT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP 0/1:35:4 0/2:17:2 1/1:40:3
```

BCF format

There is a binary equivalent of VCF, with the same data in a more compact format.

Common tools

These formats were usually developed by a team working on a bioinformatics tool with a need for a consistent input or output format. There a lot of these formats are closely associated with the original tool, although most have been adopted as common standards for other tools working with the same kind of data.

Unfortunately, although the formats are consistent, the tools themselves are not with respect to the syntax required to get them to do what you want. Although they are fully documented, ensuring you are suing them correctly is not a simple task. This is one reason why it's common to re-use a bioinformatics pipeline that's known to work correctly, rather than trying to create a new, bespoke solution for each project.

For many tasks there are more than one tool you can use; which you use should be guided by familiarity with the tool and how easy it is to set it up and run it. Computing time is cheap. Analyst time is expensive.

Bedtools

bedtools documentation

Bedtools is designed to summarise overlaps between genomic regions, or differences between genomic regions. It was designed around the `.bed` format, but can also work with `.gff`, `.bam` and `.vcf` formats.

Samtools

samtools documentation

Samtools is a Swiss army knife tool for working with `.sam`, `.bam`, and `.cram` files. It allows you to view regions of the file, convert it to other formats, manipulate and filter by metadata for reads, produce summary statistics about sequencing depth and more, and it even has a primitive genotype caller built-in.

Many of its functions require its input files to be indexed, and samtools itself has a mode to index those files.

Bamtools

bamtools documentation

Bamtools operates solely on `.bam` files. Whereas samtools includes features useful for an analyst, bamtools is more concerned with manipulating the files themselves, to merge, filter or convert.

Bamtools also has a function to index `.bam` files.

Vcftools

vcftools documentation

Vcftools, predictably, manipulates `.vcf` files. It can be used to filter variants, compare variants between files, summarise variants and handle file conversion, validation and merging.

Bcftools

bcftools documentation

Bcftools works on both `.bcf` and `.vcf` files, and does many of the same tasks as vcftools.

Bgzip and tabix

bgzip documentation tabix documentation

These tools are designed to be used together. bgzip is a blocked implementation of the widely-used gzip compression, and tabix is a tool for indexing those compressed files.

(IGV) Integrated Genome Viewer

IGV homepage

IGV is a useful tool for visualising data files. It can load a variety of formats including `.bed`, `.gff`, `.fasta`, `.sam`, `.bam` and `.vcf`. It's invaluable for looking at different files side-by-side, comparing samples or cross-referencing data and feature definitions.

IGV screenshot

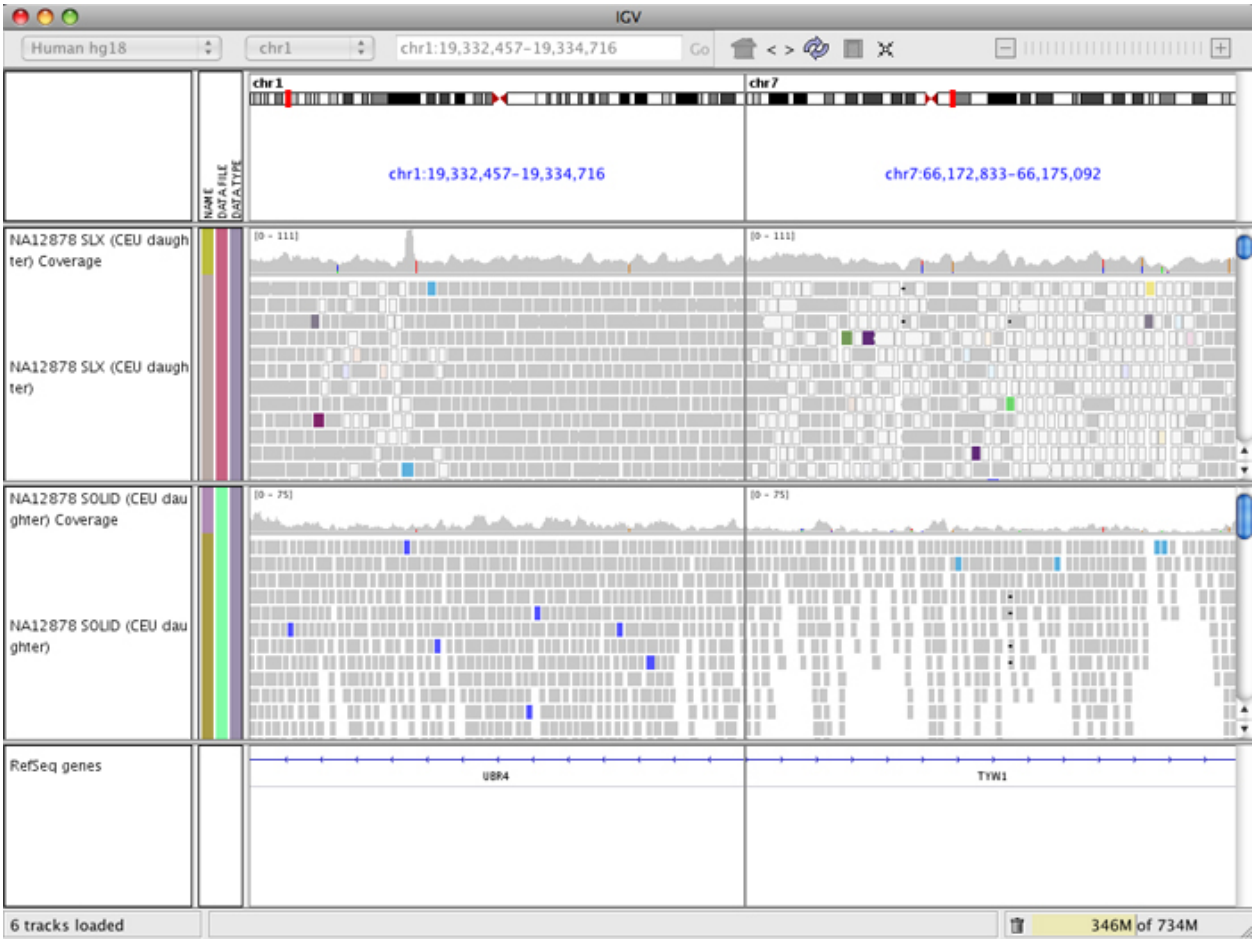

Figure 1: IGV screenshot