ASCIIGenome Documentation

Release 0.1.0

Dario Beraldi

Jan 24, 2023

Contents

1 Description **3**
- 1.1 Getting help . . . 3
- 1.2 How to cite . . . 4
- 1.3 Credits . . . 4

2 Installation **7**
- 2.1 Quick start . . . 7
- 2.2 With conda . . . 7
- 2.3 With Homebrew . . . 7
- 2.4 A little more detail . . . 8
- 2.5 Compiling the source code . . . 8

3 Input and output **11**
- 3.1 Input file formats . . . 11
  - 3.1.1 Handling large files . . . 12
- 3.2 Setting a genome . . . 12
  - 3.2.1 Reference sequence . . . 12
- 3.3 Output . . . 12
  - 3.3.1 Formatting of reads and features . . . 12
  - 3.3.2 Title lines . . . 14
- 3.4 Saving screenshots . . . 15

4 Usage **17**
- 4.1 Quick start . . . 17
- 4.2 Interactive commands . . . 18

5 Examples **19**
- 5.1 Video clips . . . 19
- 5.2 Why the command line . . . 20
- 5.3 Open and browse . . . 21
- 5.4 Finding & filtering stuff . . . 22
  - 5.4.1 Advanced filtering with *awk* . . . 23
- 5.5 Batch and non-interactive mode . . . 24
- 5.6 Finding sequence motifs . . . 24

6 FAQ and miscellanea **27**
- 6.1 Can I change colour theme / Can I use my configuration? . . . 27

6.2 Can I turn off case sensitivity? . . . 27
6.3 Why reads and coverage in bam tracks sometimes disappear? . . . 27
6.4 Can I execute multiple commands inside `-x/--exec` or at the command prompt? . . . 28
6.5 How can I print the header of a VCF file? . . . 28

7 Command reference **29**
7.1 Navigation . . . 29
7.1.1 goto . . . 29
7.1.2 INT . . . 30
7.1.3 PERCENT . . . 30
7.1.4 plus + . . . 30
7.1.5 minus - . . . 30
7.1.6 f - forward . . . 31
7.1.7 b - backward . . . 31
7.1.8 ff . . . 31
7.1.9 bb . . . 31
7.1.10] . . . 31
7.1.11 [. . . 31
7.1.12 zi . . . 31
7.1.13 zo . . . 32
7.1.14 extend . . . 32
7.1.15 l - left . . . 32
7.1.16 r - right . . . 32
7.1.17 p . . . 32
7.1.18 n . . . 32
7.1.19 next . . . 32
7.2 Find . . . 33
7.2.1 find . . . 33
7.2.2 seqRegex . . . 33
7.2.3 bookmark . . . 34
7.3 Display . . . 34
7.3.1 grep . . . 34
7.3.2 awk . . . 35
7.3.3 featureColor . . . 37
7.3.4 hideTitle . . . 37
7.3.5 genotype . . . 38
7.3.6 editNames . . . 39
7.3.7 dataCol . . . 39
7.3.8 print . . . 39
7.4 Alignments . . . 40
7.4.1 readsAsPairs . . . 40
7.4.2 filterVariantReads . . . 40
7.4.3 rpm . . . 41
7.4.4 samtools . . . 41
7.4.5 BSseq . . . 42
7.5 General . . . 42
7.5.1 setGenome . . . 42
7.5.2 setConfig . . . 42
7.5.3 explainSamFlag . . . 43
7.5.4 show . . . 43
7.5.5 recentlyOpened . . . 44
7.5.6 open . . . 44
7.5.7 reload . . . 44
7.5.8 dropTracks . . . 44

7.5.9 orderTracks 45
7.5.10 posHistory 45
7.5.11 history 45
7.5.12 save 45
7.5.13 sys 45
7.5.14 q 46
7.5.15 h 46

8 Indices and tables **47**

ASCIIGenome is a genome browser based on command line interface and designed for console terminals.

Contents:

CHAPTER 1

Description

ASCIIGenome is a command-line genome browser running from terminal window and solely based on ASCII characters. Since *ASCIIGenome* does not require a graphical interface it is particularly useful for quickly visualizing genomic data on remote servers. The idea is to make *ASCIIGenome* the Vim of genome viewers.

The closest program to *ASCIIGenome* is probably samtools tview but *ASCIIGenome* offers much more flexibility, similar to popular GUI viewers like the IGV browser.

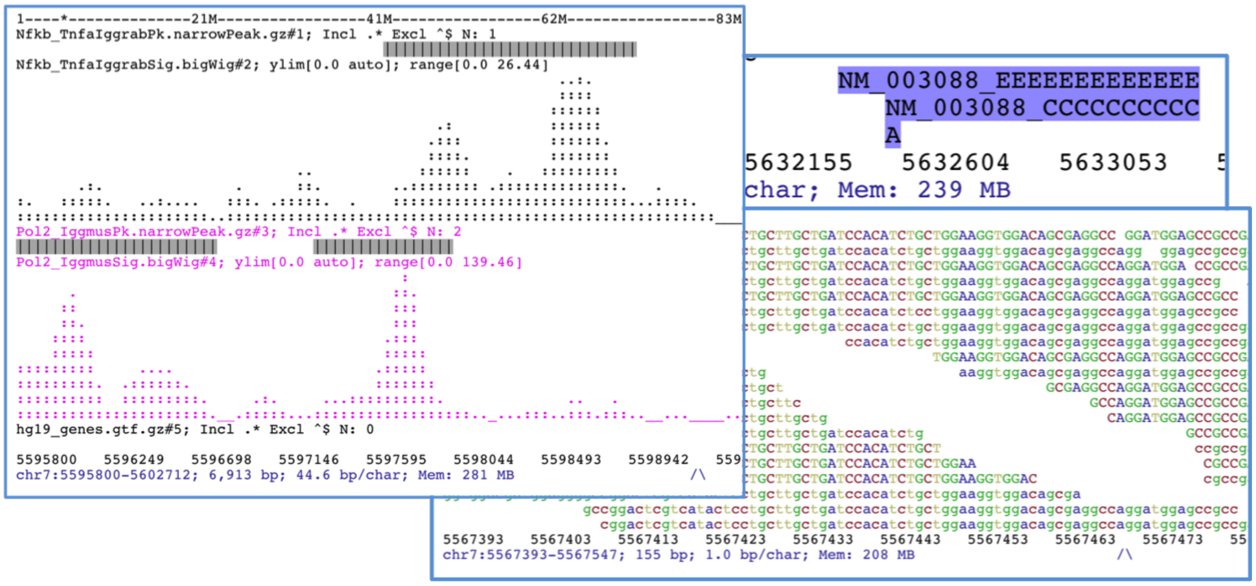
Some key features:

- Command line input and interaction, no graphical interface, minimal installation.
- Can load multiple files in various *formats*.
- Can access remote files via URL or ftp address.
- Easy navigation and searching of features and sequence motifs and filtering options.
- Support for BS-Seq alignment.

1.1 Getting help

Feel free to send any comment, bug, or issues to one or more of the following:

- Open an issue on GitHub
- Post a question on Biostars.org or bioinformatics.stackexchange.com (recommended: send a notification email to `dario<dot>beraldi<at>gmail<dot>com` so I read the question).
- Send email to `dario<dot>beraldi<at>gmail<dot>.com`



```
1----*--------------21M-----------------41M-----------------62M-----------------83M
Nfkb_TnfaIggrabPk.narrowPeak.gz#1; Incl .* Excl ^$ N: 1
Nfkb_TnfaIggrabSig.bigWig#2; ylim[0.0 auto]; range[0.0 26.44]
Pol2_IggmusPk.narrowPeak.gz#3; Incl .* Excl ^$ N: 2
Pol2_IggmusSig.bigWig#4; ylim[0.0 auto]; range[0.0 139.46]
hg19_genes.gtf.gz#5; Incl .* Excl ^$ N: 0

5595800   5596249   5596698   5597146   5597595   5598044   5598493   5598942   559
chr7:5595800-5602712; 6,913 bp; 44.6 bp/char; Mem: 281 MB                    /\
```

```
    NM_003088_EEEEEEEEEEEE
      NM_003088_CCCCCCCCCC
      A
5632155   5632604   5633053   5
char; Mem: 239 MB
```

```
TGCTTGCTGATCCACATCTGCTGGAAGGTGGACAGCGAGGCC GGATGGAGCCGCCG
tgcttgctgatccacatctgctggaaggtggacagcgaggccagg  ggagccgccg
TGCTTGCTGATCCACATCTGCTGGAAGGTGGACAGCGAGGCCAGGATGGA CCGCCG
tgcttgctgatccacatctgctggaaggtggacagcgaggccaggatggagccg
TGCTTGCTGATCCACATCTGCTGGAAGGTGGACAGCGAGGCCAGGATGGAGCCGCC
tgcttgctgatccacatctcctggaaggtggacagcgaggccaggatggagccgcc
tgcttgctgatccacatctgctggaaggtggacagcgaggccaggatggagccgccg
            ccacatctgctggaaggtggacagcgaggccaggatggagccgccg
                       TGGAAGGTGGACAGCGAGGCCAGGATGGAGCCGCCG
tg                        aaggtggacagcgaggccaggatggagccgccg
tgct                                GCGAGGCCAGGATGGAGCCGCCG
tgcttc                                   GCCAGGATGGAGCCGCCG
tgcttgctg                                  CAGGATGGAGCCGCCG
tgcttgctgatccacatctg                                GCCGCCG
TGCTTGCTGATCCACATCTGCT                               ccgccg
TGCTTGCTGATCCACATCTGCTGGAA                            CGCCG
TGCTTGCTGATCCACATCTGCTGGAAGGTGGAC                     cgccg
tgcttgctgatccacatctgctggaaggtggacagcga
      gccggactcgtcatactcctgcttgctgatccacatctgctggaaggtggacagcgaggccaggatggagccgcc
        cggactcgtcatactcctgcttgctgatccacatctgctggaaggtggacagcgaggccaggatggagccgccg
5567393   5567403   5567413   5567423   5567433   5567443   5567453   5567463   5567473   55
chr7:5567393-5567547; 155 bp; 1.0 bp/char; Mem: 208 MB                /\
```

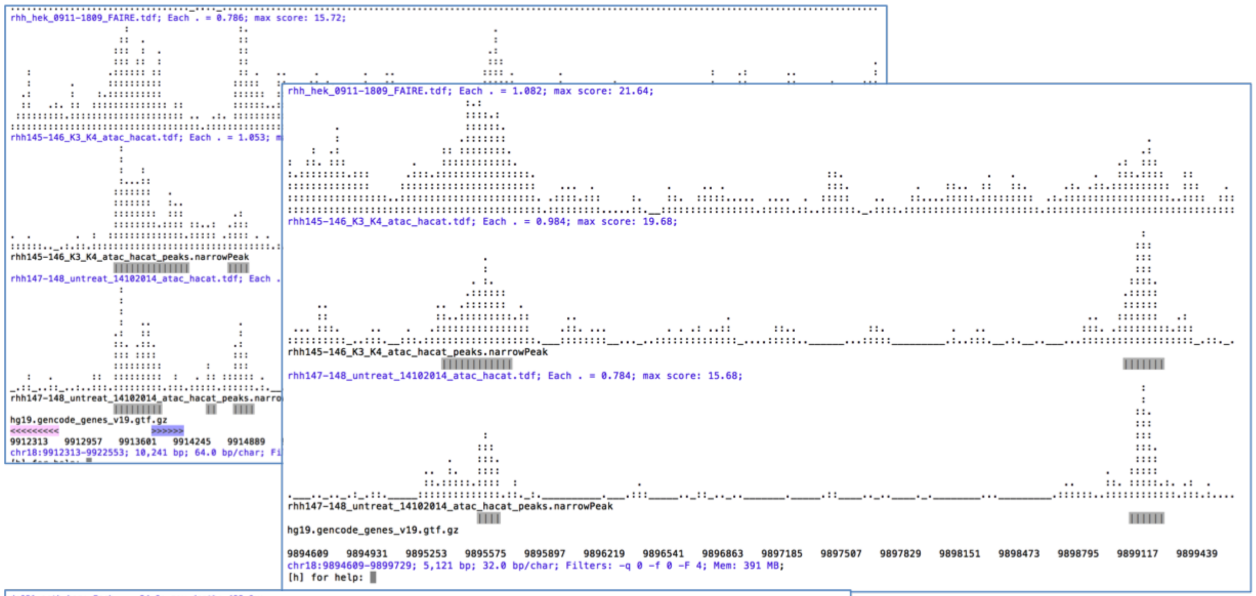
1.2 How to cite

If you happen to find *ASCIIGenome* useful and you would like to acknowledge it, please quote the GitHub repository (https://github.com/dariober/ASCIIGenome/).

An article describing *ASCIIGenome* has been published in Bioinformatics .

1.3 Credits

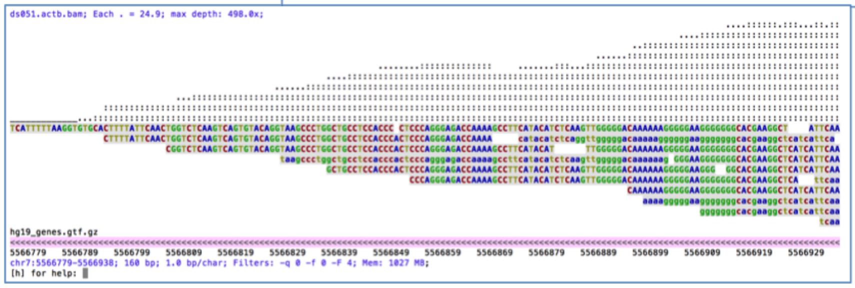
- Bam processing is mostly done with the samtools/htsjdk library.
- Bigwig and tdf are processed with classes from IGV source code.
- Block compression and indexing done using jvarkit.
- Brew installation thanks to dalloliogm.



```
rhh_hek_0911-1809_FAIRE.tdf; Each . = 0.786; max score: 15.72;
rhh145-146_K3_K4_atac_hacat.tdf; Each . = 1.053; m
rhh145-146_K3_K4_atac_hacat_peaks.narrowPeak
rhh147-148_untreat_14102014_atac_hacat.tdf; Each .
rhh147-148_untreat_14102014_atac_hacat_peaks.narro
hg19.gencode_genes_v19.gtf.gz
<<<<<<<<<               >>>>>>
9912313   9912957   9913601   9914245   9914889
chr18:9912313-9922553; 10,241 bp; 64.0 bp/char; Fi
```

```
rhh_hek_0911-1809_FAIRE.tdf; Each . = 1.082; max score: 21.64;
rhh145-146_K3_K4_atac_hacat.tdf; Each . = 0.984; max score: 19.68;
rhh145-146_K3_K4_atac_hacat_peaks.narrowPeak
rhh147-148_untreat_14102014_atac_hacat.tdf; Each . = 0.784; max score: 15.68;
rhh147-148_untreat_14102014_atac_hacat_peaks.narrowPeak
hg19.gencode_genes_v19.gtf.gz

9894609   9894931   9895253   9895575   9895897   9896219   9896541   9896863   9897185   9897507   9897829   9898151   9898473   9898795   9899117   9899439
chr18:9894609-9899729; 5,121 bp; 32.0 bp/char; Filters: -q 0 -f 0 -F 4; Mem: 391 MB;
[h] for help:
```



```
ds051.actb.bam; Each . = 24.9; max depth: 498.0x;
TCATTTTTAAGGTGTGCACTTTTATTCAACTGGTCTCAAGTCAGTGTACAGGTAAGCCCTGGCTGCCTCCACCC CTCCCAGGGAGACCAAAAGCCTTCATACATCTCAAGTTGGGGGACAAAAAAGGGGGAAGGGGGGGCACGAAGGCT    ATTCAA
                  CTTTTATTCAACTGGTCTCAAGTCAGTGTACAGGTAAGCCCTGGCTGCCTCCACCCACTCCCAGGGAGACCAAAA     catacatctcaggttgggggacaaaaagggggaagggggggcacgaaggctcatcattca
                              CGGTCTCAAGTCAGTGTACAGGTAAGCCCTGGCTGCCTCCACCCACTCCCAGGGAGACCAAAAGCCTTCATACAT      TTGGGGGACAAAAAAGGGGGAAGGGGGGGCACGAAGGCTCATCATTCAA
                                                taagccctggctgcctccacccactcccagggagaccaaaagccttcatacatctcaagttgggggacaaaaaag GGGAAGGGGGGGCACGAAGGCTCATCATTCAA
                                                        GCTGCCTCCACCCACTCCCAGGGAGACCAAAAGCCTTCATACATCTCAAGTTGGGGGACAAAAAAGGGGGAAGGG  GGCACGAAGGCTCATCATTCAA
                                                                    CCCAGGGAGACCAAAAGCCTTCATACATCTCAAGTTGGGGGACAAAAAAGGGGGAAGGGGGGGCACGAAGGCTCA   ttcaa
                                                                                                        CAAAAAAGGGGGAAGGGGGGGCACGAAGGCTCATCATTCAA
                                                                                                          aaaagggggaagggggggcacgaaggctcatcattcaa
                                                                                                                   gggggggcacgaaggctcatcattcaa
                                                                                                                                          tcaa
hg19_genes.gtf.gz
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
5566779   5566789   5566799   5566809   5566819   5566829   5566839   5566849   5566859   5566869   5566879   5566889   5566899   5566909   5566919   5566929
chr7:5566779-5566938; 160 bp; 1.0 bp/char; Filters: -q 0 -f 0 -F 4; Mem: 1027 MB;
[h] for help:
```

CHAPTER 2

Installation

2.1 Quick start

Basically, download `ASCIIGenome-X.Y.Z.zip` from releases, unzip, copy `ASCIIGenome` and `ASCIIGenome.jar` to a directory of your liking and that's it.

For example, in the commands below replace version number with the latest from releases:

```
wget https://github.com/dariober/ASCIIGenome/releases/download/vX.Y.Z/ASCIIGenome-x.y.
↪z.zip
unzip ASCIIGenome-x.y.z.zip

cd ASCIIGenome-x.y.z/
chmod a+x ASCIIGenome
cp ASCIIGenome.jar /usr/local/bin/ # Or else in your PATH e.g. ~/bin/
cp ASCIIGenome /usr/local/bin/     # Or else in your PATH e.g. ~/bin/
```

2.2 With conda

ASCIIGenome is available as a bioconda package and it can be installed with the conda package manager:

```
conda install -c bioconda asciigenome
```

2.3 With Homebrew

ASCIIGenome can also be installed through brew / Linux Brew, although it is still not an official package:

```
brew install https://raw.githubusercontent.com/dariober/ASCIIGenome/master/install/
↪brew/asciigenome.rb
```

2.4 A little more detail

`ASCIIGenome.jar` requires **Java 1.8+** and this should be the only requirement. There is virtually no installation needed as `ASCIIGenome` is pure Java and should work on most (all?) platforms. Download the zip file `ASCIIGenome-x.x.x.zip` from releases, unzip it and execute the jar file with:

```
java -jar /path/to/ASCIIGenome.jar --help
```

To avoid typing `java -jar ...` every time, you can put both the helper script `ASCIIGenome` and the jar file `ASCIIGenome.jar` in the same directory in your `PATH` and execute with:

```
ASCIIGenome [options]
```

Note: This part below has nothing to do with ASCIIGenome specifically. These are just general instructions to add executable files to your PATH.

For Unix users: If you have administrator rights and you want to make ASCIIGenome available to all users, a popular choice of installation directory is `/usr/local/bin/`, *e.g.*:

```
cp ASCIIGenome.jar /usr/local/bin/
cp ASCIIGenome /usr/local/bin/
```

If you don't have administrator rights (*i.e.* you get a `Permission denied` error) you can instead copy to a directory that you have on your PATH and where you have permission to write. A popular user directory for executable files is `$HOME/bin` (*e.g.* `/home/myName/bin` or `/Users/myName/bin` or short `~/bin`). *e.g.*:

```
cp ASCIIGenome.jar ~/bin/
cp ASCIIGenome ~/bin/
```

If `~/bin` does not exist or is not on your PATH create it with:

```
mkdir ~/bin/
```

And to add to it to your PATH edit your profile file to add the new directory. *E.g.* edit `~/.bash_profile` to:

```
PATH=/home/myName/bin:$PATH
```

Then reload the profile file or log off and log back in to make the changes effective.

Note the helper is a bash script. To set the amount of memory available to java use the `-Xmx` option as e.g. `java -Xmx1500m -jar ....`

If for some reason the text formatting misbehave, disable it with the `-nf` option.

2.5 Compiling the source code

From version 1.13 ASCIIGenome is built using the gradle build tool. If you want to edit the source code and re-compile it to an executable jar, all you need to do is:

```
# Get source
git clone https://github.com/dariober/ASCIIGenome.git
cd ASCIIGenome

```

(continues on next page)

(continued from previous page)

```
./gradlew clean
./gradlew build -x test
```

The executable jar file will be in `build/libs/ASCIIGenome.jar`. The `-x test` option builds the code without running the tests.

CHAPTER 3

Input and output

3.1 Input file formats

File name extensions matter as file types are usually recognized by their extension. Reading remote files might require starting java with the option `-Djava.net.useSystemProxies=true` (see issue#6).

Unless noted otherwise remote URL links are supported. However, there are issues reading tabix files on ftp servers (http are ok), see htsjdk/issue#797. Reading such files is possible but ASCIIGenome will first download them locally.

Format	Extension	Notes
	Annotation	
gtf, gff	`.gtf .gff .gff3`	Can be gzipped (`.gz`)
bigBed	`.bb .bigBed`	
bed	Any	Can be gzipped (`.gz`)
	Quantitative	
bigWig	`.bigWig .bw`	
bedGraph	`.bedGraph`	Can be gzipped (`.gz`)
tdf	`.tdf`	Useful for quantitative data on very large intervals.
	Other	
vcf	`.vcf`	Can be gzipped (`.gz`)
bam and sam	`.bam`	BAMs without index and SAM files are first sorted and indexed. Remote URLs are painfully slow (*same for IGV*).

Note that the recognition of the extension is *case insensitive*, so *.bigBed* is the same as *.bigbed*.

A notable format currently **not** supported is CRAM.

Tip: For input format specs see also UCSC format and Ensembl. For guidelines on the choice of format see the IGV recommendations.

Please see issue #2 and issue #41 about reading remote tabix files (*to be resolved*).

3.1.1 Handling large files

ASCIIGenome always makes use of indexing to access data so that the memory usage stays low even for large files. Tabular data files (bed, gtf/gff, bedgraph, etc) which are not block compressed and indexed are first sorted, compressed and indexed to temporary working files. This is usually quick for files of up to 1/2 million rows. For larger files consider compressing them with external utilities such as tabix. For example, to sort, compress and index a bed file:

```
sort -k1,1 -k2,2n my.bed \
| bgzip > my.bed.gz
tabix -p bed my.bed.gz
```

3.2 Setting a genome

An optional genome file can be passed to option `-g/--genome` or set with the `setGenome` command to give a set of allowed sequences and their sizes so that browsing is constrained to the real genomic space. The genome file is also used to represent the position of the current window on the chromosome, which is handy to navigate around.

There are different ways to set a genome:

- A tag identifying a built-in genome, e.g. hg19. See genomes for available genomes.
- A local file, tab separated with columns chromosome name and length. See genomes for examples.
- A bam file with suitable header.
- A fasta reference sequence (see *Reference sequence*).

3.2.1 Reference sequence

A reference sequence file is optional. If provided, it should be in fasta format, and uncompressed. If the fasta file does not have an index, *ASCIIGenome* will create a temporary index file that will be deleted on exit. A permanent index can be created with:

```
samtools faidx ref.fa
```

3.3 Output

3.3.1 Formatting of reads and features

When aligned reads are show at single base resolution, read bases follow the same convention as samtools: Upper case letters and `.` for read align to forward strand, lower case and `,` otherwise; second-in-pair reads are underlined; grey-shaded reads have mapping quality of <=5.

GTF/GFF features on are coded according to the feature column as below. For forward strand features the colour blue and upper case is used, for reverse strand the colour is pink and the case is lower. Features with no strand information are in grey.

Feature	Symbol
exon	E
cds	C
start_codon	A
stop_codon	Z
utr	U
3utr	U
5utr	W
gene	G
transcript	T
mrna	M
trna	X
rrna	R
mirna	I
ncrna	L
lncrna	L
sirna	S
pirna	P
snorna	O

This is an example of a BAM file and a GTF file. The top track shows the read coverage, the middle track the aligned reads and the bottom track the GTF features in this genomic window.

```
1--*----------------41M-----------------83M-----------------120M--------------
ds051.actb.bam#1; ylim[auto auto]; range[557.0 922.0]; -F4 -f0 -q0; Recs here/all: 1492/15098
..:.. ..       .::
::::::::::::::::::
:::::::::::::::::::
:::::::::::::::::::
:::::::::::::::::::                               .....
:::::::::::::::::::.                ....         :::::::...
:::::::::::::::::::::::::.:..:.:::..::::         ::::::::::::...  .
:::::::::::::::::::::::::::::::::::::::::...:...::::::::::::::::::::::..
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::.._
CAACTAAGTCATAGTCCGCCTAGAAGCATTTGCGGTGGACGATGGAGGGGCCGGACTCGTCATACTCCTGCTTGCTGA
ds051.actb.bam@2; -F4 -f0 -q0
c actaagtcatagtccgcctagaagcatttgcggtggacgatggaggggccggactcgtcatactcctgcttgctg
CAACTAAGTCATAGT CGCCTAGAAGCATTTGCGGTGGACGATGGAGGGGCCGGACTCGTCATACTCCTGCTTGCTGA
CAACTAAGTCATAGTCCGC              GGTGGACGATGGAGGGGCCGGACTCGTCATACTCCTGCTTGCTGA
caactaagtcatagtccgc                       TGGAGGGGCCGGACTCGTCATACTCCTGCTTGCTGA
CAACTAAGTCATAGTCCGCCTAGAAGCATT                   GCCGGACTCGTCATACTCCTGCTTGCTGA
CAACTAAGTCATAGTCCGCCTAGAAGCATTTGCGGTGGAC                   tcatcctcctgcttgctga
caactaagtcatagtccgcctagaagcatttgcggtggacgatgg
caactaagtcatagtccgcctagaagcatttgcggtggacgatggaggggccggactcg
CAACTAAGTCATAGTCCGCCTAGAAGCATTTGCGGTGGACGATGG
   CTAAGTCATAGTCCGCCTAGAAGCATTTGCGGTGGACGATGGAGGGGCCGGACTCGTCATACTCCTGCTTGCTGA
hg19_genes.gtf.gz#3; Incl .* Excl ^$ N: 3
NM_001101_eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
                   zzz
                      NM_001101_cccccccccccccccccccccccccccccccccccccccccccccc
chr7 unknown exon       5566779 5567522 . - . gene_id "ACTB"; transcript_id "N
chr7 unknown stop_codon 5567379 5567381 . - . gene_id "ACTB"; transcript_id "N
chr7 unknown CDS        5567382 5567522 . - 0 gene_id "ACTB"; transcript_id "N
5567360   5567370   5567380   5567390   5567400   5567410   5567420   5567430
chr7:5567360-5567437; 78 bp; 1.0 bp/char; Mem: 187 MB
```

If available, the feature name is shown on the feature itself. For BED features, name is taken from column 4, if available. Default for GTF/GFF is to take name from attribute `Name`, if absent try: `ID`, `transcript_name`, `transcript_id`, `gene_id`, `gene_name`. To choose an attribute see command `gffNameAttr`.

Read coverage tracks at single base resolution show the consensus sequence obtained from the underlying reads. If the reference fasta file is present the = symbol is used to denote a match. Heterozygote bases or variants are shown using the iupac ambiguity codes for up to two variants (N otherwise). Variants are called with a not-too-sophisticated heuristics: Only base qualities >= 20 are considered, an alternative allele is called if supported by at least 3 reads and makes up at least 1% of the total reads. The first and second allele must make at least 98% of the total reads otherwise the base is N (see `PileupLocus.getConsensus()` for exact implementation). Insertion/deletions are currently not considered.

3.3.2 Title lines

The title lines contains information about the track and their content depends on the track type.

For all tracks, the title line shows the file name (*e.g.* `hg19_genes_head.gtf.gz`) with appended an identifier (*e.g.* `#3`). The filename and the identifier together make the name of the track. All commands operating on tracks use this name to select tracks. The suffix identifier is handy to capture tracks without giving the full track name.

Annotation tracks (bed, gtf, gff, vcf)

Example:

```
hg19_genes_head.gtf.gz#1; N: 13; grep -i exon -e CDS
```

After the track name (`hg19_genes_head.gtf.gz#1`), the title shows the number of features in the current window (`N: 13`). Other information is shown depending on the track settings. In this example the title shows settings used to filter in and out features (`grep ...`).

Quantitative data

This title type applies to quantitative data such as bigwig and tdf and to the read coverage track.

Example:

```
ear045.oxBS.actb.bam#2; ylim[0.0 auto]; range[44.0 78.0]; Recs here/all:  255/
100265; samtools -q 10
```

Explanation:

`ear045.oxBS.actb.bam#2`: Track name as described above

`ylim[0.0 auto]` limits of the y-axis, here from 0 to the maximum of this window.

`range[44.0 78.0]` Range of the data on the y-axis.

`Recs here/all:  255/100265` number of alignments present in this window (255) versus the total number in the file (100265).

`samtools -q 10` information about mapping quality and bitwise filter set with the *samtools* command. omitted if not applicable and if no filter is set. See also explain sam flag.

Read track

This is the track showing individual reads. Example:

```
ear045.oxBS.actb.bam@3; samtools -q 10
```

`ear045.oxBS.actb.bam@3` As before, this is the track name composed of file name and track ID. In contrast to other tracks, the id starts with @ instead of #. This is handy to capture all the read tracks but not the coverage tracks, for example *trackHeight 10 bam@* applies to all the read tracks but not to the coverage tracks.

3.4 Saving screenshots

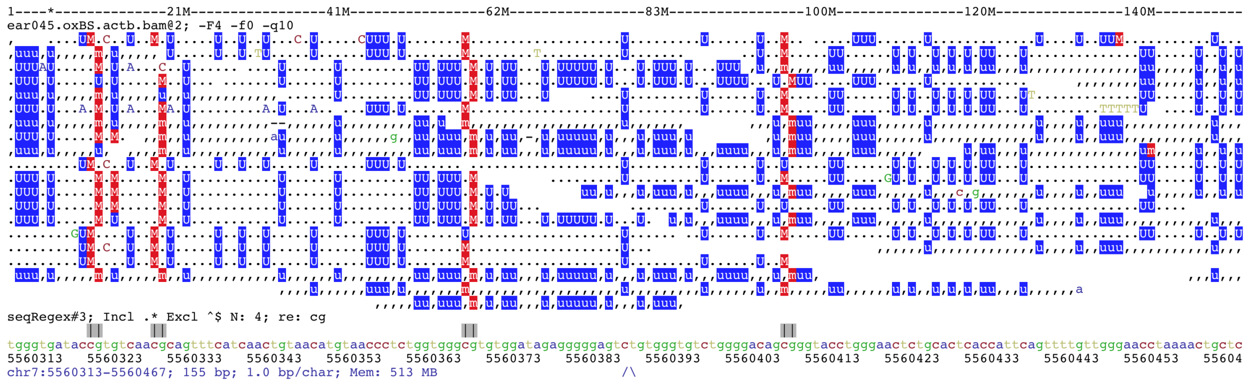
Screenshots can be saved to file with the commands `save`. Output format is either ASCII text or pdf, depending on file name extension. For example:

```
[h] for help: save mygene.txt ## Save to mygene.txt as text
[h] for help: save            ## Save to chrom_start-end.txt as text
[h] for help: save .pdf       ## Save to chrom_start-end.pdf as pdf
[h] for help: save mygene.pdf ## Save to mygene.pdf as pdf
```

Without arguments, `save` writes to file named after the current genomic position e.g. *chr1_1000-2000.txt*. The ANSI formatting (*i.e.* colours) is stripped before saving so that files can be viewed on any text editor (use a monospace font like `courier`). For convenience the variable `%r` in the file name is expanded to the current genomic coordinates, for example *save mygene.%r.pdf* is expanded to *e.g.* `mygene.chr1_1000_2000.pdf`.

See also *Advanced filtering with awk* for saving screenshots in batch by iterating through a list of positions.

This is a screenshots of bisulfite-seq data. The *BSseq* mode was set and methylated cytosines are shown in red while unmethylated cytosines in blue.



```
1----*--------------21M----------------41M----------------62M----------------83M----------------100M---------------120M---------------140M-----------
ear045.oxBS.actb.bam@2; -F4 -f0 -q10

seqRegex#3; Incl .* Excl ^$ N: 4; re: cg

tgggtgataccgtgtcaacgcagtttcatcaactgtaacatgtaaccctctggtgggcgtgtggatagaggggagtctgtgggtgtctggggacagcgggtacctgggaactctgcactcaccattcagttttgttgggaacctaaaactgctc
5560313   5560323   5560333   5560343   5560353   5560363   5560373   5560383   5560393   5560403   5560413   5560423   5560433   5560443   5560453   55604
chr7:5560313-5560467; 155 bp; 1.0 bp/char; Mem: 513 MB                  /\
```

CHAPTER 4

Usage

4.1 Quick start

The interface to *ASCIIGenome* should look familiar to those used to command line programs. To see the command line options use the usual syntax `-h/--help` as `ASCIIGenome -h`.

Open an indexed a bam file, as simple as:

```
ASCIIGenome aln.bam
```

Open with a reference genome:

```
ASCIIGenome -fa genome.fa aln.bam
```

In interactive mode use `-h` to browse available commands in interactive mode:

```
ASCIIGenome <options>
[h] for help: -h
```

And to see the help for a given command use `cmd_name -h`, for example:

```
[h] for help: next -h

next [-start] [track_id]
      Move to the next feature on track_id on *current* chromosome. `next` centers
      the window on the found feature and zooms out. This is useful for quickly␣
↪browsing
      through annotation files of genes or ChIP-Seq peaks in combination with read
      coverage tracks (bigwig, tdf, etc.). The `-start` flag sets the window right
      at the start of the feature, without centering and zooming out.
      ...
```

See also *Input and output*.

4.2 Interactive commands

As there is no GUI, everything is handled thorough command line. Once *ASCIIGenome* is started type a command and press ENTER to execute.

Some features of Unix console are enabled:

- Arrow keys UP and DOWN scroll previous commands.
- TAB auto-completes commands.
- ENTER without any argument repeats the previous command.
- Input after the `//` string is treated as comment and it is ignore (new in v1.13.0).

Examples:

```
[h] for help: ff <ENTER>                # Move forward
[h] for help: <ENTER>                   # Move forward again...
[h] for help: <ENTER>                   # ... and again
[h] for help: col <TAB>                 # Is expanded to colorTrack
[h] for help: <ARROW UP>                # Shows previous command
[h] for help: goto chr1 // Hi there     # Comments are allowed
[h] for help: h <ENTER>                 # Show help.
```

When track names are passed as arguments, it is not necessary to give the full name as partial matching is enabled. This is handy since track names have an ID appended as suffix which can be used in place of the full name, e.g. *next myLongfileName.bed#1* can be also typed as *next #1*.

These are just some functionalities to give an idea behind *ASCIIGenome*. See *Input and output* for the individual commands available.

CHAPTER 5

Examples

5.1 Video clips

These short clips should give just a feel for how *ASCIIGenome* works in interactive mode.

This example loads a bam file and shows the read track at the bottom and the coverage track on top. After loading the bam file, the follow operations are executed at the command prompt:

- Go to specified region.
- Zoom in.
- Repeatedly move forward by half a window size.
- Zoom out.
- Filter reads using samtools-like syntax.

Here there are two bigWig files of ChIP-Seq profiles. The corresponding peak regions are separately loaded. The following operations are executed at the command prompt:

- Change track height to 12 lines each.
- Add two *narrowPeak* files, captured with the glob `*.narrowPeak.gz`.
- Re-order tracks.
- Zoom out.
- Hide title lines.
- Change limits of the Y-axes to be from 0 to maximum of all tracks.
- Move to the next bed feature.
- Show help for the `ylim` command.

5.2 Why the command line

At first it may be nonsensical to use a viewer without graphical interface. In addition, typed commands involve a steeper learning curve and more frustration. However, the command line interface has some great benefits which, at least in part, explain why most bioinformaticians prefer R, python and Unix tools over Excel and Galaxy (which are great, by the way, just like IGV is great). Some advantages of the command line interface over the GUI:

- Streamline repetitive tasks.
- Finer control of the commands.
- Self-documented and therefore reproducible.

This example should illustrate these points. For a start, most of my data files, especially the big ones, live on the institutes's computer cluster or on our group server. Almost nothing is stored on my workstation. Consequently pretty much all the work I do is via Unix commands, bash and the familiar samtools, bedtools, etc. Popping up a GUI is often a disturbance of the workflow so having a visualisation tool with the same interface (*i.e.* command line) as these tools is more natural.

Now, say we want to visualise the following files:

```
ts058_TS10-PEO1-Pt-A2.tdf
ts059_TS10-PEO1-Pt-A4.tdf
ts060_TS11-PEO1-DMF-A13.tdf
ts061_TS11-PEO1-Pt-A12.tdf
ts062_TS11-PEO1-Pt-A7.tdf
ts063_TS11-PEO4-DMF-A5.tdf
ts064_TS11-PEO4-Pt-A2.tdf
ts065_TS11-PEO4-Pt-A4.tdf
ts069_PEO1-DMF-A5.tdf
ts070_PEO1-Pt-A18.tdf
ts071_PEO1-Pt-A2.tdf
```

Loading all these files by clinking one by one through a GUI can be annoying especially if they are in different directories, not counting the time spent to pop up the GUI and scroll through the relevant menus. With *ASCIIGenome* you can probably just do one of the following:

```
ASCIIGenome ts0{58..71}*.tdf
ASCIIGenome *PEO1*.tdf *PEO4*.tdf
ASCIIGenome `find . -name '*PEO*.tdf'` ## If files are in different subdirs
ASCIIGenome ts058_TS10-PEO1-Pt-A2.tdf ts059_TS10-PEO1-Pt-A4.tdf <etc>
```

The command line you have used can be copied in your documentation for reference and it can be used again by copying and pasting it to the terminal.

Once these files have been loaded you may want to order them to have the PEO1 tracks before the PEO4s. This is just:

```
orderTracks PEO1 PEO4
```

Similarly, settings can be changed without the need of scrolling through menu options, for example:

```
colorTrack blue PEO1    <- Turn blue the PEO1 tracks
trackHeight 10 DMF      <- Make 10 lines high the tracks matching DMF
```

Furthermore, the commands issued at the prompt can be scrolled with the UP and DOWN arrow keys. So if we want to change the colour of the PEO1 tracks again we just need to press UP two times, bring back the `colorTrack blue PEO1` command, and edit it as required.

We can put this together in a single command which, again, can go to the documentation:

```
ASCIIGenome -x 'orderTracks PEO1 PEO4 && colorTrack blue PEO1 && trackHeight 10 DMF'␣
↪ts0{58..71}*.tdf
```

5.3 Open and browse

Open some peak and bigWig files from ENCODE.

Note: Opening remote files is a little slow (IGV seems equally slow). You might also need to start Java with the option *-Djava.net.useSystemProxies=true* (see also issue#6)

```
encode=http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeSydhTfbs

ASCIIGenome \
    $encode/wgEncodeSydhTfbsGm10847NfkbTnfaIggrabPk.narrowPeak.gz \
    $encode/wgEncodeSydhTfbsGm10847NfkbTnfaIggrabSig.bigWig \
    $encode/wgEncodeSydhTfbsGm12892Pol2IggmusPk.narrowPeak.gz \
    $encode/wgEncodeSydhTfbsGm12892Pol2IggmusSig.bigWig
```

Find the first feature on the first file, then change colour of one of the tracks. Reset y axes to span 0 to 50, finally save as pdf to default file name:

```
    [h] for help: setGenome hg19
[h] for help: next #1
[h] for help: colorTrack pink wgEncodeSydhTfbsGm12892Pol2IggmusSig
[h] for help: ylim 0 50
[h] for help: save %r.pdf
```

Result on terminal screen should look like this:

```
*-------------------54M-----------------110M----------------160M----------------210M----------
wgEncodeSydhTfbsGm10847NfkbTnfaIggrabPk.narrowPeak.gz#1; Incl .* Excl ^$ N: 1
                                          ||||||||||
wgEncodeSydhTfbsGm10847NfkbTnfaIggrabSig.bigWig#2; ylim[0.0 50.0]; range[0.0 37.96]



                                               .
                                               :
                                               :
                                               ::
                                              :::
                                              :::
                                           . :::::.
..___ _.__...___._____...____.._.__..._...:::::::::..___  _________..__..._...._______________.
wgEncodeSydhTfbsGm12892Pol2IggmusPk.narrowPeak.gz#3; Incl .* Excl ^$ N: 1
                                          |||||||||||
wgEncodeSydhTfbsGm12892Pol2IggmusSig.bigWig#4; ylim[0.0 50.0]; range[0.0 134.91]
                                             :::
                                             :::
                                             ::: :.
                                             ::: ::
                                             ::: ::
                                            :::::::
                                            ::::::::
                                      .    .::::::::
                                     .:.:  :::::::::.           . .                  ..:   ..
_____ __....___::_____.._.__._.__::::.:::::::::::::::::.__  _..::::.:..__..........:..::::._:::.
996137    996890    997642    998395    999148    999900    1000653   1001406   1002159   1002
chr1:996137-1003137; 7,001 bp; 74.5 bp/char; Mem: 168 MB
```

The screenshot is saved to *chr1_996137-1003137.pdf*, note that the variable `%r` is expanded to the genomic coordinates.

5.4 Finding & filtering stuff

Once started, `ASCIIGenome` makes it easy to browse the genome. The picture below shows the distribution of transcripts on chromosome 36 of *Leishmania major*. It is clearly visible how transcripts in *Leishmania* tend to be grouped in blocks transcribed from the same direction (blue: forward strand, pink: reverse strand). Note how overlapping features are stacked on top of each other.

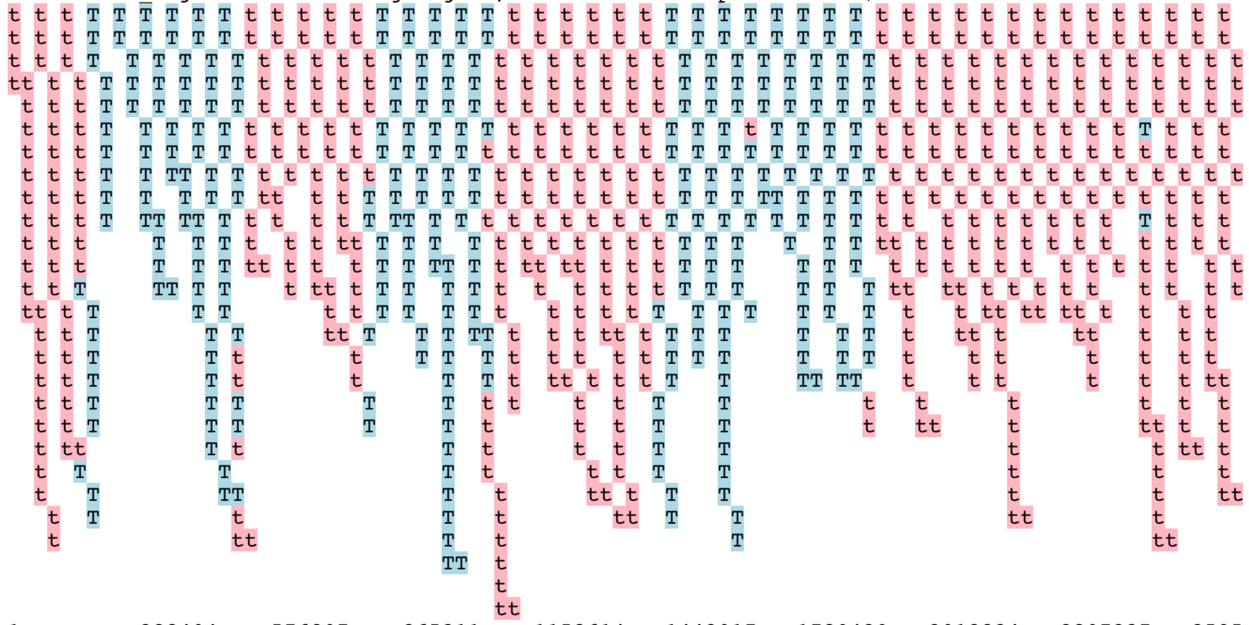
This screenshot has been produced by first loading the *L. major* GTF file:

```
ASCIIGenome ftp://ftp.ensemblgenomes.org/pub/release-31/protists/gtf/leishmania_major/
↪Leishmania_major.ASM272v2.84.gtf.gz
```

At the command prompt issue the following commands:

```
[h] for help: goto 36:1-2682151
[h] for help: grep -i transcript
[h] for help: trackHeight 100
```

```
Leishmania_major.ASM272v2.31.gtf.gz#1; Incl \ttranscript\t Excl ^$ N: 781
```



```
1          288404     576807     865211     1153614    1442017    1730420    2018824    2307227    2595
36:1-2682151; 2,682,151 bp; 28533.5 bp/char; Mem: 224 MB
```

Now return to the start of the chromosome and find the first feature containing *LmjF.36.TRNAGLN.01*, print it to screen:

```
[h] for help: 1
[h] for help: find LmjF.36.TRNAGLN.01
[h] for help: print
```

Now showing:

```
Leishmania_major.ASM272v2.31.gtf.gz#1; Incl \ttranscript\t Excl ^$ N: 1
LmjF.36.TRNAGLN.01:tRNA_tttttttttttttttttttttttttttttttttttttttttttttttt
1607643    1607653    1607663    1607673    1607683    1607693    1607703    1607713    1607723    1607733    1607
36:1607643-1607746; 104 bp; 1.0 bp/char; Mem: 246 MB
```

5.4.1 Advanced filtering with *awk*

From version 1.5.0, the awk program is available within ASCIIGenome to filter interval features and SAM records. See the command reference for *awk* for more detials. This an example of using awk to filter reads containing deletions, *i.e.* where the cigar string contains the character *D*:

```
awk '$6 ~ "D"' *.bam
```

The syntax `$6` selects the 6th column, the one containing the cigar string, the operator ~ returns true if the cigar string contains the string `D`. As usual, the last positional argument applies the command to the tracks matching the given regex.

The internal function `getSamTag()` extracts the value of a given SAM tag and this value can be used to as filter. For example, to select only reads containing mismatches, *i.e.* where the NM tag is greater than zero, use:

```
awk 'getSamTag("NM") > 0'
```

5.5 Batch and non-interactive mode

ASCIIGenome can be integrated in a script to be executed without direct human intervention. For example, a simple bash script may contain the following commands:

```
#!/bin/bash

## Find ChIP-Seq peaks
CHIP=ChIP.bam
macs2 callpeak -t $CHIP -c input.bam -n out

## Output pdf in a "control" region for later visual inspection
ASCIIGenome -ni -r chr1:1000000-1020000 -x "save ${CHIP}.ctrl.pdf" \
    $CHIP input.bam out_peaks.narrowPeak > /dev/null
```

In this script, a ChIP-Seq sample is first analysed to find peaks against an input control. ChIP, input and output from the peak caller are then loaded in *ASCIIGenome* to visualize a region of interest. *ASCIIGenome* will save the image in pdf file named after the ChIP sample and exit. An investigator can later inspect the pdf figure to assess the quality of the ChIP or to check whether a peak has been detected.

The example above can easily be extended to several regions to be visualised in batch for one or more tracks. For example, you have a list of ChIP-Seq peaks or RNA-Seq genes and you want to see the coverage profiles together with an annotation file. `ASCIIGenome` allows batch processing via the `--batchFile/-b` option.

This script iterates through the intervals in *peaks.bed*. For each interval, it displays two bigWig, a gtf file and the peak file itself. Each interval is zoomed out 3 times and the screenshot saved as pdf to `/tmp/peak.%r.pdf`, where *%r* is a special variable expanded to the current coordinates as *chrom_start-end*.:

```
ASCIIGenome -b peaks.bed \
    -x 'zo 3 && save /tmp/peak.%r.pdf' \
    chipseq.bigwig \
    input.bigwig \
    gencode_genes.gtf \
    peaks.bed > /dev/null
```

To save all the screenshots in a single pdf use the >> operator in the *save* command, *e.g.* `save >> myScreenshots.pdf`.

5.6 Finding sequence motifs

The reference fasta sequence can be searched for sequence motifs specified via regular expressions or via IUPAC notation.

This example is from Biostars. We want to find matches of the motif TATAWAA near gene ENSG00000168487.

First load the reference sequence and a (remote) annotation file:

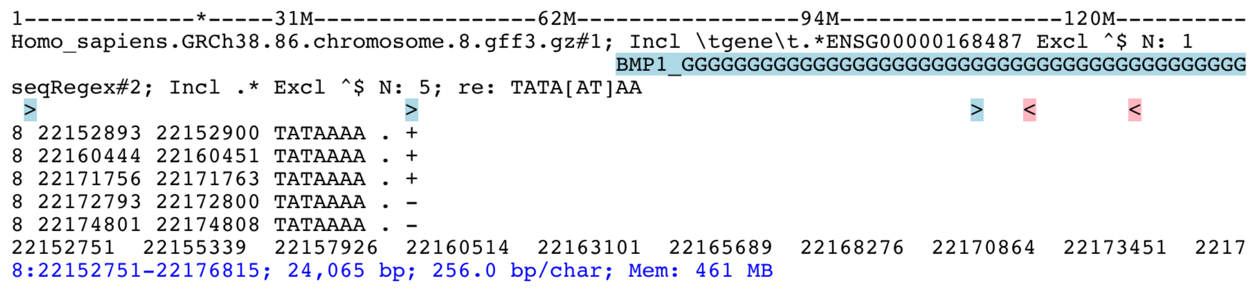
```
ASCIIGenome -fa Homo_sapiens.GRCh38.dna.chromosome.8.fa \
    ftp://ftp.ensembl.org/pub/release-86/gff3/homo_sapiens/Homo_sapiens.GRCh38.86.
↪chromosome.8.gff3.gz
```

Then at the command prompt issue these commands:

```
find ENSG00000168487
grep -i \tgene\t.*ENSG00000168487 gff3
seqRegex -iupac TATAWAA
zo 8
print seqRegex
print seqRegex > matches.bed
save matches.png
```

Explained: Find the gene ENSG00000168487, for clarity only show the "gene" feature (`grep...`). Then search the motif TATAWAA interpreted as iupac notation; zoom out *x* times (e.g. 8 times) to see some matches in the sequence.

The matches here are shown on screen with `print seqRegex` and then saved to file with `print seqRegex > matches.bed`. Finally save a picture as png, shown here:



```
1-------------*-----31M-----------------62M-----------------94M-----------------120M----------
Homo_sapiens.GRCh38.86.chromosome.8.gff3.gz#1; Incl \tgene\t.*ENSG00000168487 Excl ^$ N: 1
                                              BMP1_GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
seqRegex#2; Incl .* Excl ^$ N: 5; re: TATA[AT]AA
 >                            >                                          >   <       <
8 22152893 22152900 TATAAAA . +
8 22160444 22160451 TATAAAA . +
8 22171756 22171763 TATAAAA . +
8 22172793 22172800 TATAAAA . -
8 22174801 22174808 TATAAAA . -
22152751  22155339  22157926  22160514  22163101  22165689  22168276  22170864  22173451  2217
8:22152751-22176815; 24,065 bp; 256.0 bp/char; Mem: 461 MB
```

CHAPTER 6

FAQ and miscellanea

6.1 Can I change colour theme / Can I use my configuration?

To change colour theme use the `-c/--config` command line option or the interactive command `setConfig`. To make your own theme and set it as default, use as template one of the files in the repository directory config, edit it as desired and save it as `~/.asciigenome_config`.

Examples:

```
ASCIIGenome -c metal ...     <- Use the "metal" built-in them
ASCIIGenome -c mytheme.conf <- Read configuration from this file
ASCIIGenome ...              <- No args to -c: Read file ~/.asciigenome_config or use␣
↪default theme
```

For available colour names see the help in `colorTrack` or this cheat sheet.

6.2 Can I turn off case sensitivity?

For command that do not explicitly enable turning on or off case sensitivity, you can prepend `(?i)` to your regex to match in case insensitve mode, e.g. `(?i)bam` will capture `foo.bam` and `foo.BAM`. This is standard regular expression syntax unrelated to ASCIIGenome.

Note that the command `seqRegex` by default is case insensitive, unless the flag `-c` is set.

6.3 Why reads and coverage in bam tracks sometimes disappear?

When displaying bam files, *ASCIIGenome* is configured to disable the coverage and read tracks if the window size is >100,000 bp. This is to prevent the browsing to become too slow. To display such large windows consider bigWig or tdf file format.

6.4 Can I execute multiple commands inside `-x/--exec` or at the command prompt?

Use the `&&` to concatenate commands (similar to Unix syntax). E.g. `colorTrack red && goto chr1:150000 && zo`.

6.5 How can I print the header of a VCF file?

Assuming `bcftools` is available, use the `sys` command, for example:

```
sys bcftools view -H my_variants.vcf.gz | less
```

Of course you can further parse the header by piping to standard Unix tools. For example, to exclude the `contig` lines use:

```
sys bcftools view -h mutect/WW00282.vcf.gz | grep -v '##contig' | less
```

CHAPTER 7

Command reference

This is the documentation for the indvidual commands. The help documented here can be invoked also at the command prompt with *command -h*, for example to get the help for *ylim*:

```
ylim -h
```

Parameters in square brakets are optional and the default argument is indicated by the = sign. The syntax . . . indicate that the argument can be repeated multiple times. For example:

```
ylim min max [track_regex = .*]...
```

Means that *ylim* takes two mandatory arguments, *min* and *max*. The optional argument, *track_regex*, defaults to .* and can be repated multiple times.

7.1 Navigation

7.1.1 goto

```
goto chrom[:from[-to]] | chrom [from [to]]
```

Go to region *chrom:from-to* or to *chrom:from* or to the start of *chrom*. The region may be separated by *:* and - or by spaces. The character ':' is a shortcut for *goto*. Examples:

```
goto chr8:1-1000    # Go to region 1-1000 on chr8
goto chr8 1 1000    # Use spaces instead
goto chr8 1-1000    # Same as above
goto chr8 1 - 1000 # Same as above
goto chr8 1 1,000   # Comma in numbers is ok
goto chr8:10        # Go to position 10 on chr8
goto chr8           # Go to start of chr8
goto chr8 10 30 50 # Go to chr8:10-50
:chr8               # Colon ':' shortcut
```

7.1.2 INT

`INT from [c | to]`

Go to position *from* or to region *from to* on current chromosome. If a list of integers is given, the first and last are taken as *from* and *to*. This is handy to copy and paste intervals from the ruler above the prompt.

- `c` set the position of *from* at the center of the screen.
- `to` set the new window in the region delimited by *from* and *to*.

Examples:

```
10                     -> Will jump to position 10
10 1000                -> Go to region 10-1000
10 250 500 750 1000    -> Same as above again
750 c                  -> Put the position 750 right in the middle
750c                   -> Same as '750 c' space is optional
```

7.1.3 PERCENT

`PERCENT from [c | to]`

Zoom into the current window delimited by given PERCENT of screen. PERCENT is a number in the range 0-1 mapping to the given percent of the current genomic window. Similar to the *:code:INT* command, one number moves the genomic window to the position located at PERCENT and two numbers will zoom into the region PERCENT-PERCENT. This command is useful to quickly focus an a feature of interest, such as a ChIP-Seq peak or a variant.

- `c` set the position of *from* at the center of the screen.
- `to` set the new window in the region delimited by *from* and *to*.

Examples:

```
0.25       -> Jump to position at 25% of current screen.
.25        -> Same as above.
.25 .75    -> Zoom into the interval between 25-75% of current screen.
.25 c      -> Put the position at 25% of current screen right in the middle.
.25c       -> Same as '.25 c' (space is optional).
```

7.1.4 plus +

`+ INT [k|m]`

Move forward by *INT* bases. Suffix K/M recognized. Suffixes k (kilo) and M (mega) are expanded to x1000 and x1,000,000. Examples:

```
+2m
+10k
+10.5k
```

7.1.5 minus -

`- INT [k|m]`

Move backwards by INT bases. Suffix K/M recognized. Suffixes k (kilo) and M (mega) are expanded to x1000 and x1,000,000. Examples:

```
-100
-10k
-10.5m
```

7.1.6 f - forward

`f [NUM=0.1]`

Move forward NUM times the size of the current window, 1/10 by default.

7.1.7 b - backward

`b [NUM=0.1]`

Move backward NUM times the size of the current window, 1/10 by default

7.1.8 ff

`ff`

Move forward by 1/2 of a window. A shortcut for *f 0.5*

7.1.9 bb

`bb`

Move backward by 1/2 of a window. A shortcut for *b 0.5*

7.1.10]

`] INT=1`

Move forward by INT screen columns Same as **[** but moves forward. See **[** for details

7.1.11 [

`[ INT=1`

Move backwards by INT screen columns. The **[** character can be repeated and each **[** will move by one column. Examples:

```
[   -> Move one screen column
[[[ -> Move three columns
       [ 3 -> Same as above
       [3  -> Same as above (space is optional)
```

7.1.12 zi

`zi [INT = 1]`

Zoom in INT times. Each zoom halves the window size. To zoom quickly use INT= 5 or 10 e.g. *zi 10*

7.1.13 zo

```
zo [INT = 1]
```

Zoom out INT times. Each zoom doubles the window size. To zoom quickly use INT= 5 or 10 e.g. *zo 10*

7.1.14 extend

```
extend [mid|window] [INT left] [INT right]
```

Extend the current window by *INT* bases left and right.

- `window` (default): Extend the current window left and right by *INT* bases
- `mid` The new window is given by the midpoint of the current window plus and minus *INT* bases left and right.

If only one INT is given it is applied to both left and right. Negative INTs will shrink instead of extend the window.

7.1.15 l - left

```
l
```

Go to the Left half of the current window. Alternate the left and right command to quickly focus on a point of interest.

7.1.16 r - right

```
r
```

Go to the Right half of the current window. Alternate the left and right command to quickly focus on a point of interest.

7.1.17 p

```
p
```

Go to the previous visited position. Similar to the back and forward arrows of an Internet browser.

7.1.18 n

```
n
```

Go to the next visited position. Similar to the back and forward arrows of an Internet browser.

7.1.19 next

```
next [-back] [-start] [-c] [-zo INT=5] [track]
```

Move to the next feature not overlapping the current coordinates. By default *next* centers the window on the next feature and zooms out.

- `-back` Search backwards. I.e. move to next feature on the left of the current position.
- `-start` Set the window right at the start of the feature, without centering and zooming out.
- `-c` Set the window so that the start of the feature is right in the middle of the window. Useful to browse small features such as SNV and indels.

- `-zo INT` Zoom out INT times after having found the next feature. Ignored if the *-start* flag is set. If <= 0 the window spans exactly the feature coordinates. Default 5.
- `track` Track to search for next feature. Default to the first annotation track found.

next starts searching immediately after the current window and loops thourgh each chromosome until a feature is found.

7.2 Find

7.2.1 find

`find [-all] [-c] [-F] regex [track]`

Find the first record in *track* containing *regex*. The search for *regex* starts from the *end* of the current window (so the current window is not searched) and moves forward on the current chromosome. At the end of the current chromosome move to the next chromosomes and then restart at the start of the initial one. The search stops at the first match found. If *track* is omitted the first interval track found is searched.

- `-all`: Return the region containing **all** the regex matches.
- `-c` Match in CASE SENSITIVE mode. Default is case insensitive (changed in v1.12).
- `-F`: Interpret *regex* as a fixed, literal string instead of as a regex.

Examples:

```
find -all ACTB genes.gtf -> Find all the matches of ACTB. Case ignored
find -c 'ACTB gene'      -> Find the first match of 'ACTB gene'. Case sensitive
```

Use single quotes to define patterns containing spaces.

7.2.2 seqRegex

`seqRegex [-iupac] [-c] [regex]`

Find regex in reference sequence and show matches as an additional track. Options:

- `regex` Regex to search. If missing the seq regex track is removed.
- `-iupac` Enable the interpretation of the IUPAC ambiguity code. NB: This option simply converts IUPAC chracters to the corresponding regex.
- `-c` Enable case-sensitive matching. Default is to ignore case.

Examples:

```
seqRegex ACTG          -> Case insensitive, actg matched
seqRegex -c ACTG       -> Case sensitive, will not match actg
seqRegex -iupac ARYG -> Interpret (converts) R as [AG] and Y as [CT]
seqRegex               -> Disable regex matching track
```

To save matches to file, see the *print* command. This command is ignored if the reference fasta sequence is missing.

7.2.3 bookmark

`bookmark [-d] [-n name] [-print] [> file] [chrom:from-to]`

Creates a track to save positions of interest. Without arguments, add the current position to the bookmark track. Options:

- `chrom:from-to` Bookmark this region. If chrom is omitted, use the current chromosome.
- `-d` Remove the bookmark at coordinates [chrom:from-to].
- `-n name` Use name for this new bookmark.
- `-print` prints to screen the list of current bookmarks.
- `> file` saves the bookmark track to file.

Examples:

```
bookmark                -> Add the current window to bookmarks.
bookmark 100            -> Bookmark position 100 on current chrom
bookmark 100-110        -> Bookmark position 100-110 on current chrom
bookmark chr1:100       -> Bookmark position chr1:100
bookmark -d chr1:100    -> Delete bookmark at chr1:100
bookmark > books.txt    -> Save to file books.txt
bookmark -print         -> Show table of bookmarks
```

7.3 Display

7.3.1 grep

`grep [-i = .*] [-e = ''] [-c] [-F] [-v] [track_regex = .*]...`

Similar to grep command, filter for features including or excluding patterns. Options:

- `-i regex` Show features matching this regex.
- `-e regex` Exclude features matching this regex.
- `-c` Match in CASE SENSITIVE mode. Default is case insensitive (changed in v1.12).
- `-F` Interpret *regex* in *-i* and *-e* as a fixed, literal string instead of as a regex.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `track_regex` Apply to tracks matched by *track_regex*.

NOTES

- Use *single quotes* to delimit patterns containing spaces e.g. `-i 'ACTB gene'`

Regex *-i* and *-e* are applied to the raw lines as read from source file and it is applied only to annotation tracks (GFF, BED, VCF, etc). For example:

```
grep -i RNA -e mRNA gtf gff
```

Will show the rows containing 'RNA' but will hide those containing 'mRNA', applies to tracks whose name matches 'gtf' or 'gff'. With no arguments reset to default: `grep -i .* -e ^$ .*` which means show everything, hide nothing, apply to all tracks.

7.3.2 awk

`awk [-off ...] [-F sep_re] [-v VAR=var] [-V] '<script>' [track_regex = .*]...`

Advanced feature filtering using awk syntax. awk offers finer control then `grep` to filter records in tabular format.

Awk is column oriented. Awk splits each line into a list using a given regular expression as delimiter (default delimiter is the TAB character). To access an item, i.e. a column, use the syntax `$n` where *n* is the position of the item in the list, e.g. `$3` will access the third field (i.e. 3rd column). The variable `$0` holds the entire line as single string.

Awk understands numbers and mathematical operators. With awk you can filter records by numeric values in one or more fields since numbers are handled as such. You can also perform arithmetic operations and filter on the results.

OPTIONS

- `-off track_re ...` Turn off awk filtering for tracks captured by the list of regexes.
- `-F <sep_re>` Use regular expression <sep_re> as column separator. Default is 't' (tab). To separate on white space use e.g. 'b' (backspace) or 's' (any white space). Do not use ' '.
- `-v VAR=var` Pass to awk script the variable VAR with value var. Can be repeated.
- `script` The awk script to be executed. Must wrapped in single quotes.
- `-V` Invert selection: apply changes to the tracks not selected by list of track_regex

ADDITIONAL FEATURES

Function `get(...)` can indistinctly be applied to GTF, GFF, SAM records and to INFO and FORMAT fields in VCF files. Double quoting around <tag> is optional.

- `get(tag)` on **GTF**

Return the value of tag attribute.

- `get(tag, [value_idx])` on **GFF**

Return the value of tag attribute. If the attribute contains multiple values return the value at index value_idx (1-based). If value_idx is missing (as default), return the entire value as it is.

- `get(tag)` on **SAM**

Return the value of the given sam tag.

- `get(tag, [value_index])` on **VCF**

Return the value of the given **INFO** tag. If the tag contains multiple values, optionally return only the value at index *value_index*. If necessary, prepend 'INFO/' to tag to disambiguate it from FORMAT tags or if the header does not contain this tag. If the tag is of type 'Flag', return 1 if present, 0 otherwise.

- `get(tag, [sample_idx], [value_idx])` on **VCF**

Return the value of the **FORMAT** tag for sample index *sample_idx* (default to 1, first sample). If the tag contains multiple values, optionally return the value at index *value_idx*. If necessary, prepend 'FMT/' to tag to disambiguate it from INFO tags or if the header does not contain this tag. If the tag is of type 'Flag', return 1 if present, 0 otherwise.

- Column headers

The following variables are replaced by the appropriate column indexes, so they can be used to easily select columns. Make sure the track types are selected to be compatible with the headers.

- bam tracks:

```
$QNAME, $FLAG, $RNAME, $POS, $MAPQ, $CIGAR, $RNEXT, $PNEXT, $TLEN, $SEQ, $QUAL
```

- vcf tracks:

```
$CHROM, $POS, $ID, $REF, $ALT, $QUAL, $FILTER, $INFO, $FORMAT
```

- gtf and gff tracks:

```
$SEQNAME, $SOURCE, $FEATURE, $START, $END, $SCORE, $STRAND, $FRAME, $ATTRIBUTE
```

- bed tracks:

```
$CHROM, $START, $END, $NAME, $SCORE, $STRAND, $THICKSTART, $THICKEND, $RGB,
↪$BLOCKCOUNT, $BLOCKSIZES, $BLOCKSTARTS
```

EXAMPLES

Note the use of single quotes to wrap the actual script and the use of double quotes inside the script.

- Filter for lines where the 4th column is between 10 and 100. Apply only to tracks matching '.gtf' or '.gff':

```
awk '$4 > 10 && $4 <= 100' .gtf .gff
```

- Filter for either perfect a match or by matching a regex on 3rd column. Apply to all tracks. The second example matches regex on the entire line (similar to grep), The third example also requires features to be on + strand:

```
awk '$3 == "exon" || $3 \  ".*_codon"'

awk '$0 \  ".*_codon"'

awk '($3 == "exon" || $3 \  ".*_codon") && $7 == "+"'
```

- Filter for features size (assuming bed format) and for values after log10 transformation. For log10 we need to change base using ln(x)/ln(10):

```
awk '($3 - $2) > 1000 && (log($4)/log(10)) < 3.5'
```

- Remove awk filter for tracks captured by .gff and .gtf:

```
awk -off .gtf .gff
```

- Return bam records where NM tag (edit distance) is > 0. Double quotes around NM are optional:

```
awk 'get(NM) > 0' .bam
```

- Filter vcf records by FORMAT tag. Suppose tag AD in the *second* sample is `63,7`:

```
awk 'get(AD, 2) ...' my.vcf       # get() returns string '63,7'
awk 'get(AD, 2, 1) ...' my.vcf    # get() returns 63
awk 'get(AD, 2, 2) ...' my.vcf    # get() returns 7
awk 'get(FMT/AD, 2) ...' my.vcf   # If AD is also in INFO or missing in header
```

- Using header variables:

```
awk '$FEATURE \  "CDS" && $START > 1234' my.gff
```

With no args, turn off awk for all tracks.

NOTES & LIMITATIONS

- This is a java implementation of awk and it is independent on whether awk is on the local system. It should behave very similar to UNIX awk and therefore it has lots of functionalities. In fact, awk is a programming language in itself, search Google for more. The original code is from https://github.com/hoijui/Jawk

- Use awk only to filter features, do not use it to edit them. If features are changed by the awk script than nothing will be retained. This is because the awk command first collects the output from awk, then it matches the features in the current window with those collected from awk.
- Each line is processed independently of the others as a separate awk execution. This means that you cannot filter one line on the bases of previous or following lines.
- This awk is slow, about x5-10 times slower than UNIX awk. For few thousand records the slowdown should be acceptable. Other things being equal, use *grep* instead.
- The default delimiter is TAB not any white space as in UNIX awk.
- An invalid script throws an ugly stack trace to stderr. To be fixed.

7.3.3 featureColor

`featureColor [-r/-R expression color] [-v] [track_regex = .*]...`

Set colour for features captured by expression. This command affects interval feature tracks (bed, gff, vcf, etc) and overrides the default color for the lines captured by the expression. Expression is a regex or an awk script (autodetermined). It is useful to highlight features containg a string of interest, such as 'CDS' in gff files, or features where a numeric field satisfy a filter.

Options:

`-r <expression> <color>` Features matching `expression` will have color `color`. The expression is interpreted as regex or as an awk script and it is applied to the raw lines as read from file. This option takes exactly two arguments and can be given zero or more times.

`-R <expression> <color>` Same as `-r` but sets color for features NOT matched by regex.

`-v` Invert selection: apply changes to the tracks not selected by list of track_regex

`[track_regex]` Apply to tracks captured by this list of regexes.

Example:

```
featureColor -r CDS plum2 -r exon grey
featureColor bed         -> Reset to default the track matching 'bed'
       featureColor -R CDS grey -> Grey all features except those matching CDS

Color blue where 9th field is > 3; color red where 9th is > 6
featureColor -r '$9 > 3' blue -r '$9 > 6' red
```

Colors can be specified by name, name prefix, or integer in range 0-255. Available colours:here

Example:

```
colorTrack cyan1 ts.*gtf ts.*bam
colorTrack 40                  <- By INT
colorTrack darkv               <- Same as darkviolet
```

7.3.4 hideTitle

`hideTitle [-on | -off] [-v] [track_regex = .*]...`

Set the display of the title line matched by track_regex. Without argument -on or -off toggle between the two modes for all tracks matched by the list of regexes.

`-v` Invert selection: apply changes to the tracks not selected by list of track_regex

7.3.5 genotype

```
genotype [-n 10] [-s .*] [-r pattern rplc] [-f expr] [-v] [track_regex = .*]..
.
```

Customise the genotype rows printed under the VCF tracks.

`-n` Display up to this many samples (rows). -1 for no limit.

`-s` Select samples matching this regex.

`-r` Edit sample names to replace <pattern> with <replacement>. Names are edited only for display. To completely hide names replace with empty string `-r .* ''`. To restore original names use a regex matching nothing e.g. '^$'

`-f` Filter samples using an expression in javascript syntax. See below for details.

`-v` Invert selection: apply changes to the tracks not selected by list of track_regex

FILTER EXPRESSION

Samples can be filtered by applying arbitrary expressions to the VCF records. The VCF fields of a sample are accessed using the syntax `{TAG}`.

TAG is one of the fixed fields: CHROM, POS, ID, REF, ALT, QUAL, FILTER, or one of the INFO or FORMAT tags. In case of ambiguity, the prefix 'INFO/' or 'FMT/' should be used to identify the target tag (e.g. `{FMT/ID}` will access the ID field in FORMAT rather than the ID in the header).

The value(s) in a TAG are converted to the appropriate data type (Integer, String, etc). Tags holding more than one value are returned as arrays whose individual values should be accessed using the syntax `[index]`. E.g. `{ALT}[0]` will access the first alternate allele.

Note that the ALT and FILTER fields are always arrays, even if only one allele is present.

After substitution of the `{TAG}` placeholders with the actual values, the expression string is evaluated as a javascript script so any valid JS code is allowed including the common operators: `> < == != && ||`.

Importantly, the result of the expression must be a boolean, i.e. it must evaluate to true or false.

For each sample, the expression is evaluated for each VCF record in the current window and if ANY record returns *true*, the sample is filtered-in. To apply the filter to specific records either include only those records using e.g. commands `grep` or `awk` or make the expression more selective, e.g. by including the POS field.

As elsewhere in ASCIIGenome, if the argument (expression) contains spaces it must be enclosed in single quotes and single quotes inside the expression must be escaped. To remove the expression filter pass a blank string as argument `-f ' '` (note the white space between single quotes).

The following tags can be used to filter on the genotype. When substituted, they evaluate to true according to the sample genotype. Testing the `{GT}` tag, e.g. `{GT} == "0/1"`, achieves a similar result and gives more control but using these tags is less error prone:

- `{HOM}` genotype is homozygote.
- `{HET}` genotype is heterozygote.
- `{HOM_REF}` genotype is homozygote reference.
- `{HOM_VAR}` homozygote for an ALT allele.
- `{HET_NON_REF}` heterozygote and all alleles are non-reference.
- `{CALLED}` at least one allele is not a missing value ('.' in vcf).
- `{NO_CALL}` No allele is called (e.g. it appears as ./. in vcf).
- `{MIXED}` genotype is comprised of both calls and no-calls.

Examples of filters:

```
genotype -f '{DP} > 30' -> Display samples having DP > 30
genotype -f '{DP} > 30 && {ID} == "rs99"' -> Select also for ID
genotype -f '{FMT/XA} > 30 && {INFO/XA} == "foo"' -> Disambiguate tags
genotype -f '{ALT}[0] == "C"'  -> Access the first ALT allele
genotype -f '{HOM_REF} == false' -> Discard if homozygote ref.
```

7.3.6 editNames

`editNames [-t] [-v] <pattern> <replacement> [track_re=.*]...`

Edit track names by substituting regex pattern with replacement. Pattern and replacement are required arguments, the default regex for track is '.*' (i.e. all tracks).

- `-t` (test) flag shows what renaming would be done without actually editing the names.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex

Use '' (empty string in single quotes) to replace pattern with nothing. Examples: Given track names 'fk123_hela.bam#1' and 'fk123_hela.bed#2':

```
editNames fk123_ ''       -> hela.bam#1, hela.bed#2
editNames fk123_ '' bam   -> hela.bam#1, fk123_hela.bed#2
editNames _ ' '           -> fk123 hela.bam#1,  fk123 hela.bed#2
editNames ^.*# cells      -> cells#1, cells#2
editNames ^ xx_           -> xx_fk123_hela.bam#1, xx_fk123_hela.bed#2 (add prefix)
```

7.3.7 dataCol

`dataCol [-v] [index = 4] [track_regex = .*]...`

Select data column for bedgraph tracks containing regex. First column has index 1. This command applies only to tracks of type bedgraph.

`-v` Invert selection: apply changes to the tracks not selected by list of track_regex

For example, use column 5 on tracks containing #1 and #3:

```
dataCol 5 #1 #3
```

7.3.8 print

`print [-n INT] [-full] [-off] [-round INT] [-hl re] [-esf] [-v] [-sys CMD] [track_regex = .*]... [>|>> file]`

Print lines for the tracks matched by *track_regex*. Useful to show exactly what features are present in the current window. Features are filtered in/out according to the `grep` command. Options:

- `track_regex` Apply to tracks matched by one or more of these regexes.
- `-n INT=10` Print up to this many lines, default 10. No limit if < 0.
- `-clip` Clip lines longer than the screen width. This is the default.
- `-full` Wrap lines longer than the screen width.

- `-round INT` Round numbers to this many decimal places. What constitutes a number is inferred from context. Default 3, do not round if < 0.
- `-hl regex` Highlight substrings matching regex. If regex matches a FORMAT tag in a VCF record, highlight the tag itself and also the sample values corresponding to that tag. Alternatively, regex may be a comma separated list of column indexes to highlight. Indexes are recognized by the $ prefix. E.g. `-hl '$1, $3, $10'` will highlight columns 1, 3, 10.
- `-esf` Explain SAM Flag. Add to SAM flag an abbreviated description.
- `-off` Turn off printing.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `-sys` Parse the raw output with the given system command(s). Use `-sys null` to turn off the system commands. These commands are executed by `bash` so bash is expected to be available on the system. The commands should read from stdin and write to stdout, this is usually the case for Unix commands like `cut`, `sort`, etc. The command string must be enclosed in single quotes, single quotes inside the string can be escaped as ' (backslash-quote)
- > and >> Write output to *file*. > overwrites and >> appends to existing file. The %r variable in the filename is expanded to the current genomic coordinates. Writing to file overrides options -n and -off, lines are written in full without limit.

Without options toggle tracks between OFF and CLIP mode.

Examples:

```
print                        -> Print all tracks, same as `print .*`
print -off                   -> Turn off printing for all tracks
print genes.bed >> genes.txt -> Append features in track(s) 'genes.bed' to file
print -sys 'cut 1-5 | sort'  -> Select columns with `cut` and then sort
print -sys null              -> Turn off the execution of sysy commands
```

7.4 Alignments

7.4.1 readsAsPairs

`readsAsPairs [-on | -off] [-v] [track_regex = .*]...`

Show SAM records as pairs. If set, properly paired reads in the current window are showed joined up by tildes.

- `-on|-off` Turn on/off the pairing mode. Or toggle between the two modes if none of these flags is set.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `[track_regex = .*]...` Apply to read tracks captured by these regexes.

7.4.2 filterVariantReads

`filterVariantReads [-r from/to] [-all] [-v] [track_regex = .*]...`

Filter reads containing a variant in the given interval. `filterVariantReads` selects for reads where the read sequence mismatches with the reference sequence in the given interval on the current chromosome. This command is useful to inspect reads supporting a putative alternate allele at a variant site.

NOTES

- `filterVariantReads` requires a reference fasta sequence to be set, e.g. via the command line option `-fa <ref.fa>` or with command `setGenome`.
- The CIGAR string determines a mismatch between read and reference. Consequently, there may be an inconsistency between variant positions in reads and positions in a VCF file if some normalization or indel realignment has been performed by the variant caller that generated the VCF. In such cases consider enlarging the target interval.
- The position (POS) of deletions in VCF files refer to the first non-deleted base on the reference. Therefore, the interval to `-r` should be POS+1 to filter for reads supporting a deletion (but see also the previous point).

OPTIONS

- `-r region` Select reads mismatching in this interval. *region* can be given as: a single position, a position plus and/or minus an offset, an interval. See examples.
- `-all` Return *all* reads intersecting the `-r` interval, not just the variant ones.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `[track_regex = .*]...` Apply to read tracks captured by these regexes.

EXAMPLES:

```
filterVariantReads -r 1000+10   <- From 1000 to 1010
filterVariantReads -r 1000-10   <- From 990 to 1000
filterVariantReads -r 1000+/-10 <- From 990 to 1010
filterVariantReads -r 1000:1100 <- From 1000 to 1100
filterVariantReads -r 1000 vars.*vcf <- Apply to tracks captured by `vars.*vcf`
filterVariantReads              <- Remove filter for all tracks
```

7.4.3 rpm

`rpm [-on | -off] [-v] [track_regex = .*]`

Set display to reads per million for BAM and TDF files.

- `-on | -off` Set mode on/off. Without arguments toggle between on and off.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `track_regex` List of regexes to capture target tracks.

7.4.4 samtools

`samtools [-f INT=0] [-F INT=4] [-q INT=0] [-v] [track_re = .*] ...`

Apply samtools filters to alignment tracks captured by the list of track regexes. Useful for stranded RNA-Seq and BS-Seq: bit flag 4096 is selects reads mapping to TOP STRAND.

- `-F` Filter out flags with these bits set. NB: 4 is always set.
- `-f` Require alignment to have these bits sets.
- `-q` Require alignments to have MAPQ >= than this.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex

Examples:

```
samtools -q 10            -> Set mapq for all tracks. -f and -F reset to default
samtools -F 1024 foo bar -> Set -F for all track containing re foo or bar
samtools -f 4096          -> Select TOP STRAND reads
samtools -F 4096          -> Select BOTTOM STRAND reads
samtools                  -> Reset all to default.
```

7.4.5 BSseq

`BSseq [-on | -off] [-v] [track_regex = .*]...`

Set bisulfite mode for read tracks matched by regex. In bisulfite mode, the characters M and m mark methylated bases (i.e. unconverted C to T) and U and u are used for unmethylated bases (i.e. C converted to T). Upper case is used for reads on forward strand, small case for reverse.

- `-on | -off` Set mode. Without arguments toggle between on and off.
- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex
- `track_regex` List of regexes to capture target tracks.

Ignored without reference fasta sequence.

7.5 General

7.5.1 setGenome

`setGenome fasta|bam|genome`

Set genome and reference sequence. The genome, i.e. the list of contig names and sizes, can be extracted from the fasta reference, from a bam file or from a genome identifier (e.g. hg19). If a fasta file is used also the reference sequence becomes available.

Without arguments, set the genome using the last opened fasta file, if any and if compatible with the current tracks.

7.5.2 setConfig

`setConfig <file|tag> | <key> <value>`

Set configuration arguments.

If only one argument is given then the entire settings are replaced. Configuration can be set with one of the built-in themes: 'black_on_white', 'white_on_black', 'metal'. Alternatively, configuration can be read from file. For examples files see https://github.com/dariober/ASCIIGenome/blob/master/resources/config/

If two arguments are given, they are taken as a key/value pair to reset.

Examples:

```
setConfig metal
setConfig /path/to/mytheme.conf
       setConfig max_reads_in_stack 20000 <- Reset this param only
```

Parameters and current settings:

```
background                          231   # Background colour
foreground                          0     # Foreground colour
seq_a                               12    # Colour for nucleotide A
seq_c                               9     # Colour for nucleotide C
seq_g                               2     # Colour for nucleotide G
seq_t                               11    # Colour for nucleotide T
seq_other                           0     # Colour for any other nucleotide
shade_low_mapq                      249   # Colour for shading reads wit low MAPQ
methylated_foreground               231   # Foreground colour for methylated C
unmethylated_foreground             231   # Foreground colour for unmethylated C
methylated_background               9     # Background colour for methylated C
unmethylated_background             12    # Background colour for unmethylated C
title_colour                        0     # Default Colour for titles
feature_background_positive_strand 147   # Colour for features on forward strand
feature_background_negative_strand 224   # Colour for features on reverse strand
feature_background_no_strand        249   # Colour for features without strand␣
↪information
footer                              12    # Colour for footer line
chrom_ideogram                      0     # Colour for chromosome ideogram
ruler                               0     # Colour for ruler
max_reads_in_stack                  2000  # Max number of reads to accumulate when␣
↪showing read tracks
shade_baseq                         13    # Shade read base when quality is below this␣
↪threshold
shade_structural_variant            33    # Background colour for reads suggesting␣
↪structural variation
highlight_mid_char                  true  # Highlight mid-character in read tracks?
nucs_as_letters                     true  # Show read nucleotides as letters at single␣
↪base resolution?
show_soft_clip                      false # NOT IN USE YET - Show soft clipped bases␣
↪in read tracks?
```

7.5.3 explainSamFlag

`explainSamFlag INT [INT ...]`

Explain the list of bitwise SAM flags. Decode one or more sam flags to human readable form and print them as a table. Similar to https://broadinstitute.github.io/picard/explain-flags.html

7.5.4 show

`show <arg>`

Show or set features to display. The argument `arg` takes the following choices:

- `genome`: Show chromosomes and their sizes as barplot provided a genome file is available.
- `trackInfo`: Show information on tracks.
- `gruler`: Toggle the display of the genomic coordinates as ruler.
- `pctRuler`: Toggle the display of the column number of the terminal (useful for navigation within the current genomic window).

`arg` can be just a prefix of the argument name, e.g. `show ge` will be recognized as `show genome`.

7.5.5 recentlyOpened

`recentlyOpened [-grep = .*]`

List recently opened files. Files are listed with their absolute path.

- `-n INT` Return only the last INT files.
- `-grep <pattern>` Filter for files (strings) matching pattern. Use single quotes to define patterns containing spaces, e.g. `-grep 'goto chr1'`.

7.5.6 open

`open [files | URLs | indexes]...`

Add tracks from local or remote files. The list of files to open can be a list of file names or URLs. For local files, glob characters (wildcard) are expanded as in Bash (but note that currently globs in directory names are not expanded.)

Alternatively, the files to open can be given as numeric indexes of recently opened files (see command `recentlyOpened`). The last opened file has index 1, the second last 2, etc.

Examples:

```
open peaks.bed genes.*.gtf         <- Note use of wildecard
open http://remote/host/peaks.bed <- From URL
open 1 2 3                         <- The three most recent files
```

7.5.7 reload

`reload [track_regex = .*]...`

Reload track files. *reload* is useful when an input track file is edited by external actions and you want to reload it in the current session. This is easier than dropping and re-opening tracks with *dropTracks ... && open ...* since track formattings and filters are preserved.

A track is dropped if it cannot be reloaded, for example when the sequence disctionary has become incompatible with the current one.

Examples:

```
reload       <- reload all tracks
reload .bam  <- reload files matching '.bam'
```

7.5.8 dropTracks

`dropTracks [-t] [-v] track_regex [track_regex]...`

Drop tracks matching any of the listed regexes. * `-t` (test) flag only shows which tracks would be removed but do not remove them.

- `-v` Invert selection: apply changes to the tracks not selected by list of track_regex

Examples:

```
dropTracks bam
```

7.5.9 orderTracks

`orderTracks [track_regex]...`

Reorder tracks according to the list of regexes or sort by name. Not all the tracks need to be listed, the missing ones follow the listed ones in unchanged order. Without arguments sort track by tag name. For example, given the track list: *[hela.bam#1, hela.bed#2, hek.bam#3, hek.bed#4]*:

```
orderTracks #2 #1   -> [hela.bed#2, hela.bam#1, hek.bam#3, hek.bed#4]
orderTracks bam bed -> [hela.bam#1, hek.bam#3, hela.bed#2, hek.bed#4]
orderTracks . bam   -> 'bam' tracks go last
orderTracks         -> name sort [hela.bam#1, hela.bed#2, hek.bam#3, hek.bed#4]
```

7.5.10 posHistory

`posHistory [-n INT=10]`

List the visited positions. Recorded positions include the current and the previous sessions of ASCIIGenome.

`-n INT` Show only the last INT positions. Show all if <= 0.

7.5.11 history

`history [-n INT] [-grep = .*]`

List the executed commands. Commands executed in previous sessions of ASCIIGenome are in /.asciigenome_history

- `-n INT` Return only the last INT commands.
- `-grep <pattern>` Filter for commands (strings) matching pattern. Use single quotes to define patterns containing spaces, e.g. `-grep 'goto chr1'`

7.5.12 save

`save [>>] [filename = chrom_start_end.txt']`

Save screenshot to file as text or pdf format. The default file name is generated from the current coordinates and the default format is plain text. If the file name has extension '.pdf' then save as pdf. To append to an existing file use >>. The string `%r` in the file name is replaced with the current coordinates. Examples:

```
save mygene.txt     -> Save to mygene.txt as text
save >> mygene.txt  -> Append to mygene.txt
save                -> Save to chrom_start-end.txt as text
save .pdf           -> Save to chrom_start-end.pdf as pdf
save mygene.%r.pdf  -> Save to mygene.chr1_100-200.pdf as pdf
```

7.5.13 sys

`sys [-L] command`

Execute a system command. By default the given `command` is executed as a string passed to Bash as `bash -c string`. With the `-L` option the command is executed literally as it is. Note that with the `-L` option globs are not expanded by Java. Examples:

```
sys pwd                          <- Print working directory name
sys ls *.bam                     <- List files ending in .bam
sys bcftools view -h vars.vcf.gz <- Print vcf header
```

7.5.14 q

`q`

Quit

7.5.15 h

`h`

`help`, `h`, `-h`, and `?` show this help. For help on individual commands use one of:

```
command -h
?command
help command
```

e.g. *ylim -h*

CHAPTER 8

Indices and tables

- genindex
- modindex
- search