The Variation Toolkit

Pierre Lindenbaum PhD.

Institut du Thorax
INSERM UMR-915
44000 Nantes - France

plindenbaum@yahoo.fr
http://plindenbaum.blogspot.com
https://twitter.com/yokofakun

January 5, 2012

1 Introduction

The Variation Toolkit is a set of C/C++ programs to handle Variant Call Format (VCF). The programs have been preliminary designed for knime4bio (`http://code.google.com/p/knime4bio/`) [3] but here, the bioinformaticians are the preliminary audience for this toolkit.

2 Building

2.1 Dependencies

Samtools : Utilities for post-processing alignments in the SAM format. [2]

Tabix : fast retrieval of sequence features from generic TAB-delimited files. [1]

mysql-dev : the files and libraries for mysql.

libxml2 : the C library for xml `http://xmlsoft.org/`.

libxslt : the C library for xslt `http://xmlsoft.org/XSLT/`.

libcurl : the C library for downloading URLs.

jkentsrc : (optional) Jim Kent's C library / Ucsc parsing bigbed, bigwig...

sqlite3 : (optional) SQL database engine.

2.2 Building

Download the latest version of varking using `subversion`:

> $ svn checkout http://variationtoolkit.googlecode.com/svn/trunk/ variationtoolkit-read-only

or update your current version by calling `"svn update"` in the variationtoolkit folder.

> $ svn update

Edit the file variationtoolkit/config.mk . You'll have to set the path to the sources of tabix (¿=0.2.5), samtools (¿=-0.1.18), etc..

config.mk example:

```
##path to SAMTOOLS
SAMDIR=${HOME}/samtools-0.1.18
##path to TABIX
TABIXDIR=/home/lindenb/tabix-0.2.5
##optional path to UCSC kent's src
KENTDIR=${HOME}/src/kent/kent
##optional path to google leveldb
LEVELDBDIR=${HOME}/tmp/leveldb-read-only
SQLITE_LIB=-lsqlite3
SQLITE_CFLAGS=
```

then type:

```
$ make
```

the binaries will be generated in the "bin" folder.

2.3 Colors

Some programs use colors (e.g; 'manhattan'). Colors can be specified as a rgb triple (e.g: 'rgb(100,10,100), rgb(0.5)', as a name (e.g: 'red'), as an hexadecimal '#AAAAAA', etc...

2.4 Base index

The tools reading and writing VCF-like files use '+1' as the index of the first base.

2.5 Input

The tools read from files or from the standard input. The streams can be gzipped.

3 Tools

All the tools have the option "-h" that provides some help.

3.1 scanvcf

Reads a strean with two columns: a Sample-Name and the file path to a VCF(.gz) . It then prints all the VCF, adding an extra column with the sample name to the output.

Options:

- –sample or -S (column-index) column for the path to SAMPLE
- –vcf or -V (column-index) column for the path to VCF(.gz)

Example:

```
$ head -n3 input.txt

#Sample VCF
Sample1 data/sample1.vcf.gz
Sample2 data/sample1.vcf.gz
Sample2 data/sample1.vcf.gz


$ cat input.txt |scanvcf

#CHROM POS ID REF ALT QUAL FILTER . FORMAT Call SAMPLE
1 879317 rs7523549 C T 71 0 . GT:GQ:DP:FLT 0/1:34:8:0 Sample1
1 880238 rs3748592 A G 51 0 . GT:GQ:DP:FLT 1/1:51:8:0 Sample1
1 880390 rs3748593 C A 99 0 . GT:GQ:DP:FLT 1/0:99:30:0 Sample1
1 881627 rs2272757 G A 99 0 . GT:GQ:DP:FLT 1/0:59:20:0 Sample1
(...)
Y 13524507 . C T 99 0 . GT:GQ:DP:FLT 1/1:99:233:0 Sample20
Y 21154323 rs10465459 G A 99 0 . GT:GQ:DP:FLT 1/1:99:215:0 Sample20
Y 21154426 rs52812045 G A 99 0 . GT:GQ:DP:FLT 1/0:99:143:0 Sample20
Y 21154466 rs10465460 T A 99 0 . GT:GQ:DP:FLT 1/1:99:134:0 Sample20
Y 21154529 . G A 51 0 . GT:GQ:DP:FLT 1/1:51:8:0 Sample20
```

3.2 extractinfo

Extract a field from the INFO column of a VCF file.

Options:

- -c (info-column-infex) (7)
- –delim (column-delimiter) (default:tab)
- -t (tag) (required)
- -N (string) symbol for NOT-FOUND. default:N/A
- -i ignore line if tag was not found

Example: The following script extract the GN(gene name) field from the column INFO. We keep the lines for the gene NOTCH2 and we display the associated SNP.

```
$ gunzip -c data.vcf.gz |\
  extractinfo -t GN -i | \
  awk -F '       ' '($11 =="NOTCH2")' |\
  cut -d ' ' -f 3 | grep rs

rs6685892
rs2493392
rs2493420
rs7534585
rs7534586
rs2493409
rs2453040
rs2124109
```

3.3 extractformat

Extract a field from the FORMAT column of a VCF file.

Options:

- -f (format-column-infex) (8)
- -c (call-column-infex) (9)

- –delim (column-delimiter) (default:tab)
- -t (tag) (required)
- -N (string) symbol for NOT-FOUND. default:N/A

Example: The following command line extract the field 'GT' from the VCF and we count the occurence of the values.

```
$ gunzip -c data.vcf.gz |\
   extractformat -t GT |\
   cut -d '        ' -f 11 |\
   sort |\
    uniq -c

     29
  10729 0/1
  10800 1/0
  13518 1/1
     11 1/2
```

3.4 ncbiefetch

Fetch a record from NCBI database. Currently supported databases: pubmed , nucleotide, proteine, snp , gene and taxonomy.

Options:

- -D (database) (default pubmed)
- -d (delimiter) (default:tab)
- -c (column=int)

Example: The following example generates a sequences of 6 pubmed ID and we call ncbiefetch to download the records.

```
$  (echo "#GI"; seq 1000 2 1010)   |\
      ncbiefetch -c 1 |\
```

```
      cut -c 1-100

#GI pubmed.year pubmed.title pubmed.journal pubmed.abstract
1000 1976 The amino acid sequence of Neurospora NADP-specific glutamate dehyd
1002 1976 The amino acid sequence of Neurospora NADP-specific glutamate dehyd
1004 1976 Properties of 5-aminolaevulinate synthetase and its relationship to
1006 1976 The attachment of glutamine synthetase to brain membranes.
Biochemical medicine ...
1008 1976 Nature and possible origin of human serum ribonuclease.
Biochemical and biophysical resear
1010 1976 Formation of non-amidine products in the chemical modification of h
```

The following example creates a sequence of 3 gi, we fetch each record (the gi is in the 1st column) and we cut the result down to 80 characters.

```
$ (echo "#GI"; seq 5 2 10)  | ncbiefetch -c 1 -D nucleotide | cut -c 1-80
#GI nucleotide.type nucleotide.accver nucleotide.taxid nucleotide.orgname
nucleo
5 nucleotide X60065.1 9913 Bos taurus B.bovis beta-2-gpI mRNA for beta-2-glyc
7 nucleotide X51700.1 9913 Bos taurus Bos taurus mRNA for bone Gla protein
437 G
9 nucleotide X68321.1 9913 Bos taurus B.taurus mRNA for cyclin A
1512 GAATTCCAGG
```

Let's download some data for 3 rs## from dbsnp.

```
echo -e "#RS\nrs25\nrs26\nrs27"  | ncbiefetch -c 1 -D snp

#RS snp.het snp.bitField snp.seq5 snp.obs snp.seq3 snp.map
rs25 0 0501000800010305001201 01 AGTAAGAGGAATCAATGTCATAGGCTTTAGATAGCATTTATGACT
rs26 0 0501000800110001000007 00 AAATGTGTGACCAAGAAAATGACtttttttttttccgactgtgtc
rs27 0 0501000800010301001001 00 TCTATGTCCAGAACTATGGATATATATTGACCTTAACTGTCAAGT
```

Taxonomy

```
$ echo -e "#Taxon-id\n9606\n9605"  | ncbiefetch -c 1 -D taxonomy

#Taxon-id taxon.name taxon.lineage
9606 Homo sapiens cellular organisms; Eukaryota; Opisthokonta; Metazoa; Eumet
9605 Homo cellular organisms; Eukaryota; Opisthokonta; Metazoa; Eumetazoa; Bi
```

Gene:

```
$ (echo "#Gene-Id"; seq 105 2 110)  | ncbiefetch -c 1 -D gene

#Gene-Id gene.locus gene.desc gene.maploc gene.ids gene.summary
105 ADARB2 adenosine deaminase, RNA-specific, B2 HGNC=227|Ensembl=ENSG0000018
107 ADCY1 adenylate cyclase 1 (brain) HGNC=232|Ensembl=ENSG00000164742|HPRD=0
109 ADCY3 adenylate cyclase 3 HGNC=234|Ensembl=ENSG00000138031|HPRD=02620|MIM
```

3.5 samplespersnp

Appends a column with the number of Samples per Variation.

Options:

- –delim (char) or -d (char) (delimiter) default:tab
- –norefalt : don't look at REF and ALT
- –sample SAMPLE column index
- –chrom CHROM column index: default 1
- –pos POS position column index: default 2
- –ref REF reference allele column index: default 4
- –alt ALT alternate allele column index: default 5
- -e (query) (optional) filters by samples using boolean request eg. '((S1 && S2) —— (!(S3) —— "S4"))'.

Example: The following command line scans the VCF, sort the variations by CHROM/POS/REF/ALT/SAMPLE, counts the number of samples/variation

```
$ cat list.tsv | scanvcf  |\
  sort -t'  ' -k1,1 -k2,2n -k4,4 -k5,5 -k11,11 |\
  samplespersnp --sample 11 | awk '($8=".")'

1 753269 rs61770172 C G 99 0 . GT:GQ:DP:FLT 1/1:99:116:0 Sample16 1
```

```
1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:63:31:0 Sample10 7
1 753405 rs61770173 C A 81 0 . GT:GQ:DP:FLT 1/1:51:19:0 Sample12 7
1 753405 rs61770173 C A 35 0 . GT:GQ:DP:FLT 1/0:35:66:0 Sample19 7
1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:99:35:0 Sample3 7
1 753405 rs61770173 C A 90 0 . GT:GQ:DP:FLT 1/1:90:21:0 Sample5 7
1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:99:36:0 Sample6 7
1 753405 rs61770173 C A 90 0 . GT:GQ:DP:FLT 1/1:90:21:0 Sample9 7
1 876499 rs4372192 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample12 6
1 876499 rs4372192 A G 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample16 6
1 876499 rs4372192 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample17 6
1 876499 rs4372192 A G 45 0 . GT:GQ:DP:FLT 1/1:45:6:0 Sample18 6
1 876499 rs4372192 A G 45 0 . GT:GQ:DP:FLT 1/1:45:6:0 Sample4 6
1 876499 rs4372192 A G 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample6 6
1 877831 rs6672356 T C 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample14 2
1 877831 rs6672356 T C 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample4 2
1 878601 . C T 98 0 . GT:GQ:DP:FLT 0/1:50:11:0 Sample14 1
```

This tool also takes an option '-e' for a query over the samples. e.g: "variation must contains Sample11, Sample12 BUT NOT Sample1 to Sample5:"

```
-e '(Sample11  && Sample12 && (!(Sample1 || Sample2 || Sample3 || Sample4 ||
```

3.6 groupbysnp

Creates a pivot table with the data(samples)=f(SNP)

Options:

- –delim (char) or -d (char) (delimiter) default:tab
- –norefalt : don't look at REF and ALT
- –sample SAMPLE column index
- –chrom CHROM column index: default 1
- –pos POS position column index: default 2
- –ref REF reference allele column index: default 4
- –alt ALT alternate allele column index: default 5

- -T 1,2,3,4,... columns indexes on top.
- -L 5,6,7,... columns indexes on left.
- -n (name1,name2,name3,...) add this sample name.

Examples: Read the VCF data and generate a pivot table.

```
$  cat sample2vcf.tsv | scanvcf | grep -v "##"  |\
   sed 's/^#CHROM/#/' |\
   sort -t '       ' -k1,1 -k2,2n -k4,4 -k5,5 -k11,11 |\
   sed 's/^#/#CHROM/' |\
   groupbysnp -L 1,2,3,4,5 -T 6,7,8,9,10 --sample 11  -n Sample1,Sample2,Samp
   verticalize

>>>2
$1 #CHROM        1
$2 POS           753405
$3 ID            rs61770173
$4 REF           C
$5 ALT           A
$6 Sample1          .
$7 Sample1:QUAL     .
$8 Sample1:FILTER   .
$9 Sample1:INFO     .
$10 Sample1:FORMAT   .
$11 Sample1:CALL     .
$12 Sample2          .
$13 Sample2:QUAL     .
$14 Sample2:FILTER   .
$15 Sample2:INFO     .
$16 Sample2:FORMAT   .
$17 Sample2:CALL     .
$18 Sample3          Sample3
$19 Sample3:QUAL     99
$20 Sample3:FILTER   0
$21 Sample3:INFO     AC=2;DB=3;ST=0:0,3:32;DP=35;NC=-0.76;UM=3;CQ=...
$22 Sample3:FORMAT   GT:GQ:DP:FLT
$23 Sample3:CALL     1/1:99:35:0
```

```
$24 Sample4          .
$25 Sample4:QUAL     .
$26 Sample4:FILTER   .
$27 Sample4:INFO     .
$28 Sample4:FORMAT   .
$29 Sample4:CALL     .
$30 count.samples 1
<<<2

>>>3
$1 #CHROM        1
$2 POS           876499
$3 ID            rs4372192
$4 REF           A
$5 ALT           G
$6 Sample1          .
$7 Sample1:QUAL     .
$8 Sample1:FILTER   .
$9 Sample1:INFO     .
$10 Sample1:FORMAT   .
$11 Sample1:CALL     .
$12 Sample2          .
$13 Sample2:QUAL     .
$14 Sample2:FILTER   .
$15 Sample2:INFO     .
$16 Sample2:FORMAT   .
$17 Sample2:CALL     .
$18 Sample3          .
$19 Sample3:QUAL     .
$20 Sample3:FILTER   .
$21 Sample3:INFO     .
$22 Sample3:FORMAT   .
$23 Sample3:CALL     .
$24 Sample4          Sample4
$25 Sample4:QUAL     45
$26 Sample4:FILTER   0
$27 Sample4:INFO     AC=2;DB=1;ST=0:0,6:0;DP=6;NC=-3.05;UM=3;CQ=...
$28 Sample4:FORMAT   GT:GQ:DP:FLT
```

```
$29 Sample4:CALL     1/1:45:6:0
$30 count.samples 1
<<<3

>>>4
$1 #CHROM        1
$2 POS           877831
$3 ID            rs6672356
$4 REF           T
$5 ALT           C
$6 Sample1           .
$7 Sample1:QUAL      .
$8 Sample1:FILTER    .
$9 Sample1:INFO      .
$10 Sample1:FORMAT    .
$11 Sample1:CALL      .
$12 Sample2           .
$13 Sample2:QUAL      .
$14 Sample2:FILTER    .
$15 Sample2:INFO      .
$16 Sample2:FORMAT    .
$17 Sample2:CALL      .
$18 Sample3           .
$19 Sample3:QUAL      .
$20 Sample3:FILTER    .
$21 Sample3:INFO      .
$22 Sample3:FORMAT    .
$23 Sample3:CALL      .
$24 Sample4          Sample4
$25 Sample4:QUAL     39
$26 Sample4:FILTER   0
$27 Sample4:INFO     AC=2;DB=1;ST=0:0,2:2;DP=4;NC=0.40;UM=3;CQ=...
$28 Sample4:FORMAT   GT:GQ:DP:FLT
$29 Sample4:CALL     1/1:39:4:0
$30 count.samples 1
<<<4

>>>5
```

```
$1 #CHROM        1
$2 POS           879317
$3 ID            rs7523549
$4 REF           C
$5 ALT           T
$6 Sample1          CALL
$7 Sample1:QUAL     71
$8 Sample1:FILTER   0
$9 Sample1:INFO     AC=1;DB=1;ST=2:1,3:2;DP=8;NC=2.16;UM=3;CQ=...
$10 Sample1:FORMAT   GT:GQ:DP:FLT
$11 Sample1:CALL     0/1:34:8:0
$12 Sample2          .
$13 Sample2:QUAL     .
$14 Sample2:FILTER   .
$15 Sample2:INFO     .
$16 Sample2:FORMAT   .
$17 Sample2:CALL     .
$18 Sample3          .
$19 Sample3:QUAL     .
$20 Sample3:FILTER   .
$21 Sample3:INFO     .
$22 Sample3:FORMAT   .
$23 Sample3:CALL     .
$24 Sample4          .
$25 Sample4:QUAL     .
$26 Sample4:FILTER   .
$27 Sample4:INFO     .
$28 Sample4:FORMAT   .
$29 Sample4:CALL     .
$30 count.samples 1
<<<5

>>>6
$1 #CHROM        1
$2 POS           880238
$3 ID            rs3748592
$4 REF           A
$5 ALT           G
```

```
$6 Sample1          CALL
$7 Sample1:QUAL     51
$8 Sample1:FILTER   0
$9 Sample1:INFO     AC=2;DB=1;ST=0:0,4:4;DP=8;NC=-3.73;UM=3;CQ=...
$10 Sample1:FORMAT   GT:GQ:DP:FLT
$11 Sample1:CALL     1/1:51:8:0
$12 Sample2          Sample2
$13 Sample2:QUAL     54
$14 Sample2:FILTER   0
$15 Sample2:INFO     AC=2;DB=1;ST=0:0,3:6;DP=9;NC=-3.73;UM=3;CQ=..
$16 Sample2:FORMAT   GT:GQ:DP:FLT
$17 Sample2:CALL     1/1:54:9:0
$18 Sample3          Sample3
$19 Sample3:QUAL     54
$20 Sample3:FILTER   0
$21 Sample3:INFO     AC=2;DB=1;ST=0:0,4:5;DP=9;NC=-3.73;UM=3;CQ=...
$22 Sample3:FORMAT   GT:GQ:DP:FLT
$23 Sample3:CALL     1/1:54:9:0
$24 Sample4          Sample4
$25 Sample4:QUAL     72
$26 Sample4:FILTER   0
$27 Sample4:INFO     AC=2;DB=1;ST=0:0,5:10;DP=15;NC=-3.73;UM=3;CQ=...
$28 Sample4:FORMAT   GT:GQ:DP:FLT
$29 Sample4:CALL     1/1:72:15:0
$30 count.samples 4
<<<6
(...)
```

3.7 numericsplit

A simple numeric splitter.

Options:

- -c (column-infex) (-1)
- –delim (column-delimiter) (default:tab)
- -m (min-value)

- -M (max-value)
- -v Inverse

Example: The following command line extracts the number of samples/variation and only keep the variation carried by 5 to 9 samples.

```
$ cat list.tsv | scanvcf  |\
 sort -t'  ' -k1,1 -k2,2n -k4,4 -k5,5 -k11,11 |\
 samplespersnp --sample 11 |\
 numericsplit -c 12 -m 5 -M 9 | awk '($8=".")' | head

1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:63:31:0 Sample10 7
1 753405 rs61770173 C A 81 0 . GT:GQ:DP:FLT 1/1:51:19:0 Sample12 7
1 753405 rs61770173 C A 35 0 . GT:GQ:DP:FLT 1/0:35:66:0 Sample19 7
1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:99:35:0 Sample3 7
1 753405 rs61770173 C A 90 0 . GT:GQ:DP:FLT 1/1:90:21:0 Sample5 7
1 753405 rs61770173 C A 99 0 . GT:GQ:DP:FLT 1/1:99:36:0 Sample6 7
1 753405 rs61770173 C A 90 0 . GT:GQ:DP:FLT 1/1:90:21:0 Sample9 7
1 876499 rs4372192 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample12 6
1 876499 rs4372192 A G 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample16 6
1 876499 rs4372192 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample17 6
1 876499 rs4372192 A G 45 0 . GT:GQ:DP:FLT 1/1:45:6:0 Sample18 6
1 876499 rs4372192 A G 45 0 . GT:GQ:DP:FLT 1/1:45:6:0 Sample4 6
1 876499 rs4372192 A G 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample6 6
1 900285 rs4970435 C T 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 Sample11 9
1 900285 rs4970435 C T 42 0 . GT:GQ:DP:FLT 1/1:32:6:0 Sample12 9
1 900285 rs4970435 C T 66 0 . GT:GQ:DP:FLT 1/1:66:13:0 Sample13 9
1 900285 rs4970435 C T 42 0 . GT:GQ:DP:FLT 1/1:42:5:0 Sample14 9
1 900285 rs4970435 C T 48 0 . GT:GQ:DP:FLT 1/1:48:7:0 Sample15 9
1 900285 rs4970435 C T 66 0 . GT:GQ:DP:FLT 1/1:66:13:0 Sample16 9
1 900285 rs4970435 C T 51 0 . GT:GQ:DP:FLT 1/1:51:9:0 Sample17 9
```

3.8 groupbygene

transposes a VCF table with a "GENE" and a "SAMPLE" column and ouput a new table: count(Gene)=f(SAMPLE)

Options:

- –delim (char) delimiter default:tab
- –norefalt : don't look at REF and ALT
- –sample SAMPLE column index
- –gene GENE column index
- –chrom CHROM column index: default 1
- –pos POS position column index: default 2
- -ref REF reference allele column index: default 4
- –alt ALT alternate allele column index: default 5

Example: The following command line extracts the name of the GENE, sort the data on CHROM/POS/REF/ALT/SAMPLE and group the data by gene.

```
$  cat list.tsv | scanvcf  |\
   extractinfo -t GN | awk '($12!="N/A")' |\
   sort -t '       ' -k1,1 -k2,2n -k4,4 -k5,5 -k11,11 |\
   groupbygene --gene 12 --sample 11


GENE CHROM START END count(SAMPLES) count(distinct_MUTATION)
count(Sample1) count(Sample2) count(Sample3) count(Sample4)
count(Sample5)
A1 19 58862835 58864479 5 2 2 2 2 2 2
A1CF 10 52569637 52576068 5 3 1 3 1 1 1
A2M 12 9230038 9264946 5 8 2 2 2 7 3
A2ML1 12 8990937 9020912 5 17 12 7 12 13 10
A4GALT 22 43088971 43089849 5 3 3 3 3 3 1
A4GNT 3 137843106 137850003 4 3 3 3 3 3 0
(...)
```

3.9 normalizechrom

Normalizes the name of a chromosome to/from UCSC/ENSEMBL.

Options:

- -i (string) ignore lines starting with this string.
- -d (string) column delimiter (default:tab).
- -c (int) column index (+1) (default:1).
- -E convert to ENSEMBL syntax (default is UCSC).

Example:

```
$ echo -e "1\t10\nX\t20\nMT\t30" | normalizechrom

chr1 10
chrY 20
chrM 30

$ echo -e "chr1\t10\nX\t20\nchrM\t30" | normalizechrom -E

1 10
X 20
MT 30
```

3.10 ranges

Chromosomes region below/above a given values:

Options:

- -c (chrom Column) (1)
- -p (pos Column) (2)
- -v (value Column) (0)
- -d (column-delimiter) (default:tab)
- -L (double: treshold) (default:0)

Example: The following command lines prints the ranges of QUAL (column 6) below/abouve "50.0".

```
$ gunzip -c data.vcf.gz |\
  sort -t ' ' -k1,1 -k2,2n |\
  ranges -v 6 -L 50
```

```
#CHROM chromStart chromEnd length In/Out Mean Count
1 0 883624 883625 + 80 4
1 881628 887559 5932 -39 1
1 883626 889158 5533 + 89.4 5
1 889159 892459 3301 -48 1
1 889160 914332 25173 + 90.7143 7
1 912050 915226 3177 -40.5 2
1 914941 985265 70325 + 82.875 8
1 982995 986442 3448 -36 1
1 985267 1225611 240345 + 82.1667 6
(...)
```

3.11 dnacontext

Prints the DNA context of a variation using a genome indexed with samtools.

Options:

- -c (chrom Column) (1)
- -p (pos Column) (2)
- -d (column-delimiter) (default:tab)
- -x (segment-size) (default:10)
- -f (genome file indexed with tabix.

Example:

```
$ gunzip -c data.vcf.gz |\
  grep -v "##" | normalizechrom |\
  dnacontext -f hg19.fa  |\
  awk '($8=".")'

#CHROM POS ID REF ALT QUAL FILTER . FORMAT CALL LEFT(DNA) CONTEXT(DNA) RIGHT(
chr1 879317 rs7523549 C T 71 0 . GT:GQ:DP:FLT 0/1:34:8:0 GAGTTTTCTA C GTGGCCA
chr1 880238 rs3748592 A G 51 0 . GT:GQ:DP:FLT 1/1:51:8:0 AGCCAGCCTT A GAGGTTA
chr1 880390 rs3748593 C A 99 0 . GT:GQ:DP:FLT 1/0:99:30:0 TGCCCTCCCG C CAGATG
chr1 881627 rs2272757 G A 99 0 . GT:GQ:DP:FLT 1/0:59:20:0 TACAAGGTCA G GGGTGT
chr1 883625 rs4970378 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 GAAGAGCAGG A GAGAGGG
chr1 887560 rs3748595 A C 99 0 . GT:GQ:DP:FLT 1/1:99:40:0 CCAGGCTGAC A AGTCAG
(...)
```

3.12 prediction

variation predictor.

Options:

- -d (column-delimiter) (default:tab)
- -f genome file indexed with samtools faidx.
- -c (CHROM col) (default:0)
- -p (POS col) (default:1)
- -r (REF col) (default:3)
- -a (ALT col) (default:4)
- –host (mysql host) default:genome-mysql.cse.ucsc.edu
- –user (mysql user) default:genome
- –password (mysql password) default:
- –database (mysql database) default:hg19
- –port (mysql password) default:0

Example:

```
$ mysql -e 'select chrom as "#CHROM",cdsStart+1 as "POS","." as "ID","A" as "
   prediction -f hg19.fa  | \
   verticalize | cut -c 1-80

>>>2
$1 #CHROM                    chr1
$2 POS                       12190
$3 REF                       A
$4 ALT                       G
$5 knownGene.name            uc001aaa.3
$6 knownGene.strand          +
$7 knownGene.txStart         11873
$8 knownGene.txEnd           14409
$9 knownGene.cdsStart        11873
$10 knownGene.cdsEnd          11873
$11 prediction.type           UTR3
$12 prediction.pos_in_cdna    .
$13 prediction.pos_in_protein .
$14 prediction.exon           .
$15 prediction.intron         .
$16 prediction.wild.codon     .
$17 prediction.mut.codon      .
$18 prediction.wild.aa        .
$19 prediction.mut.aa         .
$20 prediction.wild.prot      .
$21 prediction.mut.prot       .
$22 prediction.wild.rna       .
$23 prediction.mut.rna        .
$24 prediction.splicing       .
<<<2

>>>3
$1 #CHROM                    chr1
$2 POS                       12190
$3 REF                       A
$4 ALT                       G
```

```
$5 knownGene.name            uc010nxq.1
$6 knownGene.strand          +
$7 knownGene.txStart         11873
$8 knownGene.txEnd           14409
$9 knownGene.cdsStart        12189
$10 knownGene.cdsEnd          13639
$11 prediction.type           EXON|EXON_CODING_NON_SYNONYMOUS
$12 prediction.pos_in_cdna    0
$13 prediction.pos_in_protein 1
$14 prediction.exon           Exon 1
$15 prediction.intron         .
$16 prediction.wild.codon     ATG
$17 prediction.mut.codon      GTG
$18 prediction.wild.aa        M
$19 prediction.mut.aa         V
$20 prediction.wild.prot      MSESINFSHNLGQLLSPPRCVVMPGMPFPSIRSPELQKTTADLDHTL
$21 prediction.mut.prot       VSESINFSHNLGQLLSPPRCVVMPGMPFPSIRSPELQKTTADLDHTL
$22 prediction.wild.rna       ATGAGTGAGAGCATCAACTTCTCTCACAACCTAGGCCAGCTCCTGTC
$23 prediction.mut.rna        GTGAGTGAGAGCATCAACTTCTCTCACAACCTAGGCCAGCTCCTGTC
$24 prediction.splicing       .
<<<3

>>>4
$1 #CHROM                     chr1
$2 POS                        12190
$3 REF                        A
$4 ALT                        G
$5 knownGene.name             uc010nxr.1
$6 knownGene.strand           +
$7 knownGene.txStart          11873
$8 knownGene.txEnd            14409
$9 knownGene.cdsStart         11873
$10 knownGene.cdsEnd           11873
$11 prediction.type            UTR3
$12 prediction.pos_in_cdna     .
$13 prediction.pos_in_protein  .
$14 prediction.exon            .
$15 prediction.intron          .
```

```
$16 prediction.wild.codon     .
$17 prediction.mut.codon      .
$18 prediction.wild.aa        .
$19 prediction.mut.aa         .
$20 prediction.wild.prot      .
$21 prediction.mut.prot       .
$22 prediction.wild.rna       .
$23 prediction.mut.rna        .
$24 prediction.splicing       .
<<<4

>>>5
$1 #CHROM                    chr1
$2 POS                       69091
$3 REF                       A
$4 ALT                       G
$5 knownGene.name            uc001aal.1
$6 knownGene.strand          +
$7 knownGene.txStart         69090
$8 knownGene.txEnd           70008
$9 knownGene.cdsStart        69090
$10 knownGene.cdsEnd          70008
$11 prediction.type           EXON|EXON_CODING_NON_SYNONYMOUS
$12 prediction.pos_in_cdna    0
$13 prediction.pos_in_protein 1
$14 prediction.exon           Exon 1
$15 prediction.intron         .
$16 prediction.wild.codon     ATG
$17 prediction.mut.codon      GTG
$18 prediction.wild.aa        M
(...)
```

3.13 manhattan

plots a manhattan plot as postscript.

Options:

- -c (int) CHROM column default:1
- -p (int) POS column default:2
- -v (int) value column default:0
- -r (int) COLOR column (optional)
- -s (int) SAMPLE column (optional)
- -m (double) optional user's min value
- -M (double) optional user's max value

Example: the following command lines creates a Manhattan plot for the QUALities of a VCF file.

```
$ gunzip -c data.vcf.gz | grep -v "##" | \
   normalizechrom | cut -d '     ' -f 1,2,6 |\
   manhattan > result.ps
$ evince result.ps
```

Example2: plotting with Samples and Colors.

```
 cat sample2vcf.tsv |\
     scanvcf | awk '($3==".")' |grep NON_SYNO |\
     cut -d '     ' -f 1,2,6,11 |\
     awk '{printf("%s\trgb(10,%d,%d)\n",$0,255-(int($3)/100.0)*255.0,(int($3)
     manhattan -v 3 -r 5 -s 4 > result.ps
$ evince result.ps
```

3.14 ncbiesearch

Search NCBI/Entrez:

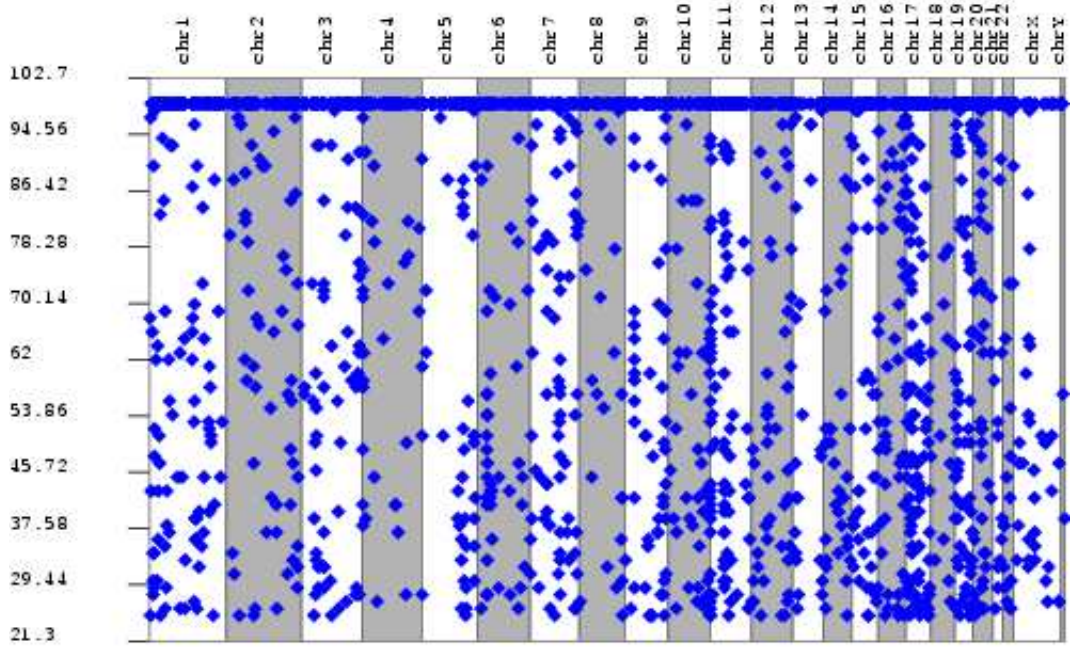

Figure 1: An Manhattan plot

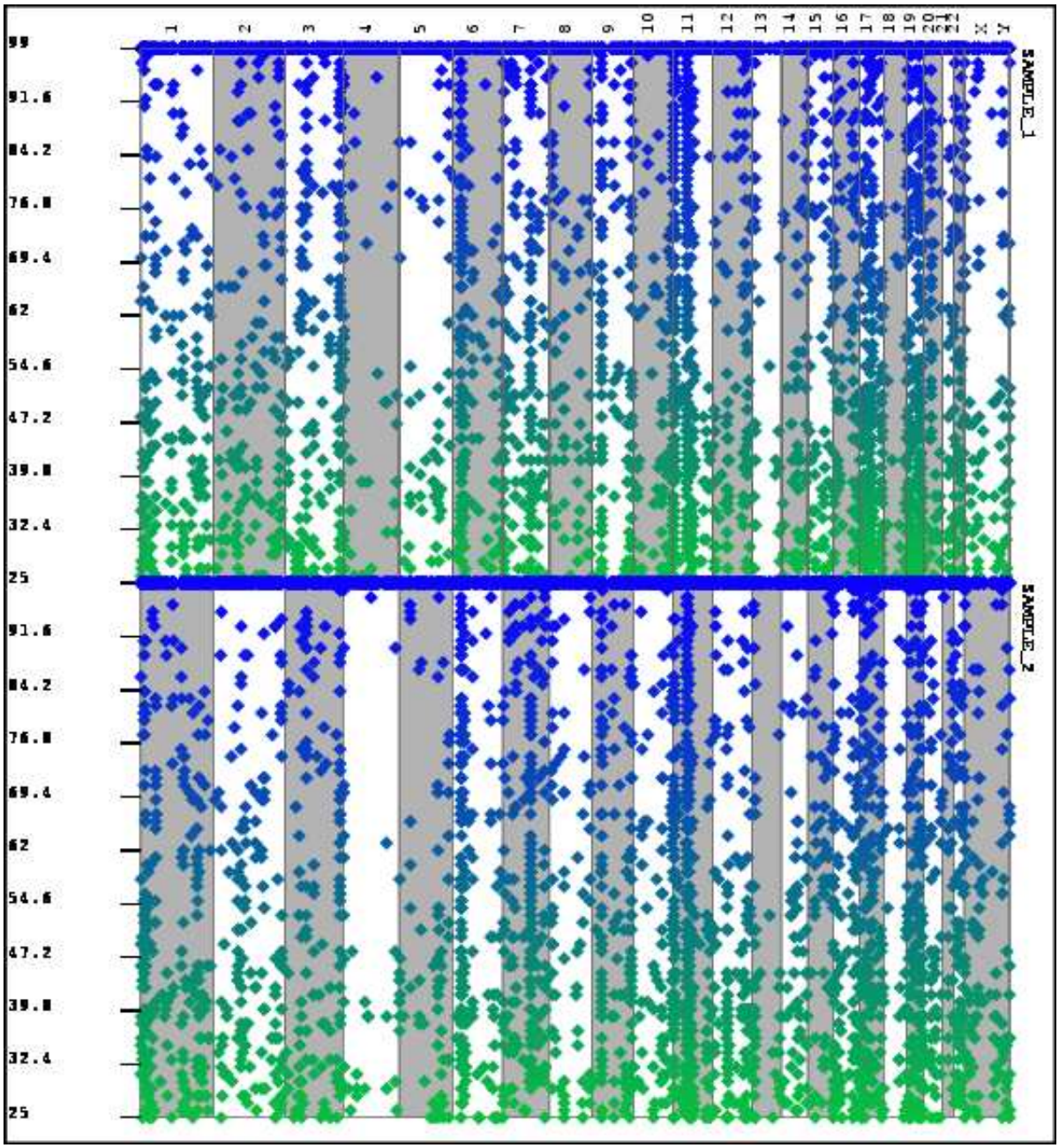

Figure 2: An Manhattan plot with colors and samples

Options:

- -D (database) (default pubmed)
- -q (query) [required]
- -d (delimiter) (default:tab)
- -L (limit=int) (default:10)

Example: The following example creates a sequence of 3 names, we search the NCBi for each name and the word "Rotavirus" in the title, limit to 2 record, we fetch each record (the PMID is in the 2nd column) and we cut the result down to 80 characters.

```
$ echo -e "#subject\nPiron\nLindenbaum\nPoncet" |\
   ncbiesearch -q '$1 "Rotavirus"[TITL]' -L 2  |\
   ncbiefetch -c 2 |\
   cut -c 1-80
#subject pubmed.id pubmed.year pubmed.title pubmed.journal pubmed.abstract
Piron 10888646 2000 Efficient translation of rotavirus mRNA requires simultan
Piron 10364288 1999 Identification of the RNA-binding, dimerization, and eIF4
Lindenbaum 15047801 2004 RoXaN, a novel cellular protein containing TPR, LD,
Lindenbaum 8985320 1997 In vivo and in vitro phosphorylation of rotavirus NSP
Poncet 21864538 2011 Structural Organisation of the Rotavirus Nonstructural P
Poncet 20935207 2010 Rapid generation of rotavirus-specific human monoclonal
```

3.15 vcfttview

Prints the BAM context around variations. Original code from samtools ttview : Heng Li, Bob Handsaker, Jue Ruan, Colin Hercus, Petr Danecek

Options:

- -c (chrom Column) (1)
- -p (pos Column) (2)
- -s (sample Column) (0) [optional]

- -d (column-delimiter) (default:tab)
- -B (bam-file) [defines one main bam for all data]
- -f (file) loads a file tab delimited with SAMPLE-NAME tPATH-TO-BAM
- -F (SAMPLE) (FILE) push a SAMPLE-NAME/PATH-TO-BAM in the current list
- -a for one position, print all BAM
- -x (int) shift x bases to the right: default10
- -w (int) screen width default:80
- -R (fasta) reference file indexed with samtools faidx

Example:

```
$ echo -e "ref\t3\nref2\t2" |\
  vcfttview -x 3 -B toy.bam -R toy.fa

>ref:3

1         11              21        31         41        51        61
AGCATGTTAGATAA****GATA**GCTGTGCTAGTAGGCAG*TCAGCGCCATNNNNNNNNNNNNNNNNNNNNNNNN
      ........    ....  ......K.K......K. ..........
      ........AGAG....***...      ,,,,,    ,,,,,,,,,
        ......GG**....AA
        ..C...**** ...**...>>>>>>>>>>>>>>>T.....



>ref2:2

1         11            21        31        41        51        61
aggttttataaaac****aattaagtctacagagcaactacgcgNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
.............Y    ..W..................
```

```
.............****..A...
 .............****..A...T.
     .........AAAT.............
         C...T****.....................
           ..T****.....................
             T****.....................
```

3.16 vcftabix

Intersection VCF/Tabix

Options:

- -d (char) column delimiter. default: TAB
- -c (int) chromosome column (1).
- -p (int) pos column (2).
- -f (filename) tabix file (required).
- -1 remove 1 to the VCF coodinates.
- -S (NOT-FOUND-String) default:!N/A.
- -m (int=mode) 0)=all 1:only-matching 2:only-non-matching default:0.

Example: download some 1000G data:

```
curl  -s "ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20100804/ALL.
          gunzip -c | grep -v "##" |\
          head -n 10000 > ~/1000G.vcf
```

Index this file with tabix.

```
$ bgzip 1000G.vcf
$ tabix -p vcf 1000G.vcf.gz
\begin{verbatim}
```

Compute the intersection of this file with our VCF. Only retain the matching line (option -m 1)

```
$ gunzip -c data.vcf.gz |\
  grep -v "##"  |  normalizechrom -E | vcftabix  -f 1000G.vcf.gz -m 1 |\
  awk '($8=".")' |  awk '($18=".")'
#CHROM POS ID REF ALT QUAL FILTER . FORMAT Call #CHROM POS ID REF ALT QUAL FI
1 879317 rs7523549 C T 71 0 . GT:GQ:DP:FLT 0/1:34:8:0 1 879317 rs7523549 C T
1 880238 rs3748592 A G 51 0 . GT:GQ:DP:FLT 1/1:51:8:0 1 880238 rs3748592 A G
1 880390 rs3748593 C A 99 0 . GT:GQ:DP:FLT 1/0:99:30:0 1 880390 rs3748593 C A
1 881627 rs2272757 G A 99 0 . GT:GQ:DP:FLT 1/0:59:20:0 1 881627 rs2272757 G A
1 883625 rs4970378 A G 39 0 . GT:GQ:DP:FLT 1/1:39:4:0 1 883625 rs4970378 A G
1 887560 rs3748595 A C 99 0 . GT:GQ:DP:FLT 1/1:99:40:0 1 887560 rs3748595 A C
1 887801 rs3828047 A G 99 0 . GT:GQ:DP:FLT 1/1:99:32:0 1 887801 rs3828047 A G
1 888639 rs3748596 T C 99 0 . GT:GQ:DP:FLT 1/1:99:32:0 1 888639 rs3748596 T C
1 888659 rs3748597 T C 99 0 . GT:GQ:DP:FLT 1/1:99:26:0 1 888659 rs3748597 T C
(...)
```

3.17 mysqlquery

Sends a mysql query for each row:

Options:

- -d (char) delimiter default:tab
- –host mysql host (genome-mysql.cse.ucsc.edu)
- –user mysql user (genome)
- –password mysql password ( )
- –database mysql db (hg19)
- –port (int) mysql port (default)
- -e or -q (SQL query)
- -L (int) limit number or rows returned

Example:

```
$  echo -e "#Gene\nuc001aaa.3\nHello\nuc001aac.3" |\
      mysqlquery --host localhost --user anonymous --port 3316  \
             -q 'select mRNA,description from kgXref where kgId="$1"'  |\
      verticalize
>>>2
$1 #Gene       uc001aaa.3
$2 mRNA        BC032353
$3 description Homo sapiens mRNA for DEAD/H box polypeptide 11 like 1 (DDX11L
<<<2

>>>3
$1 #Gene       Hello
$2 mRNA        .
$3 description .
<<<3

>>>4
$1 #Gene       uc001aac.3
$2 mRNA        BC063459
$3 description Homo sapiens cDNA FLJ31670 fis, clone NT2RI2004984.
<<<4
```

3.18 mysqlucsc

Intersection VCF/UCSC mysql data.

Options:

- –delim (char) delimiter default:tab
- –host mysql host (genome-mysql.cse.ucsc.edu)
- –user mysql user (genome)
- –password mysql password ( )
- –database mysql db (hg19)

- –port (int) mysql port (default)
- –table or -T (string)
- -C (int) chromosome column (first is 1).
- -S (int)start column (first is 1).
- -E (int) end column (first is 1).
- -f first column is not header.
- -1 data are +1 based.
- –limit (int) limit number or rows returned
- –field (string) set custom field. Can be used several times
- –type (int) type of selection: 0 any (default), 1 user data IN mysql data,2 user data embrace mysql data. (stdin—files)

Example: Compute the intersection of our data with ucsc.snp132 keep the lines containing the word 'syn'.

```
$ gunzip -c data.vcf.gz |\
   grep -v "##"  |  normalizechrom |\
   mysqlucsc --host myhost --user mypassword -C 1 -S 2 -E 2 --table snp132
   awk '($8=".")' | grep -i syn |\
   head

chr1 16375063 rs45612832 C G 67 0 . GT:GQ:DP:FLT 0/1:67:53:0 709 chr1 1637506
chr1 16375063 rs45612832 C G 67 0 . GT:GQ:DP:FLT 0/1:67:53:0 709 chr1 1637506
chr1 16890671 rs55951643 T C 99 0 . GT:GQ:DP:FLT 1/0:99:1177:0 713 chr1 16890
chr1 16890671 rs55951643 T C 99 0 . GT:GQ:DP:FLT 1/0:99:1177:0 713 chr1 16890
chr1 22176831 rs2290500 C T 87 0 . GT:GQ:DP:FLT 0/1:47:9:0 754 chr1 22176683
chr1 26361669 rs61742342 C A 99 0 . GT:GQ:DP:FLT 1/0:99:34:0 786 chr1 2636166
chr1 26608828 rs17838088 G A 36 0 . GT:GQ:DP:FLT 1/1:28:4:0 788 chr1 26608828
chr1 27210721 rs3170660 T C 99 0 . GT:GQ:DP:FLT 1/1:99:63:0 792 chr1 27210721
chr1 64643277 rs7527017 C T 99 0 . GT:GQ:DP:FLT 0/1:99:117:0 1078 chr1 646432
chr1 110709719 rs7527375 T C 99 0 . GT:GQ:DP:FLT 1/0:99:31:0 1429 chr1 110709
```

3.19 vcfbigwig

Intersection VCF/BigWig

Options:

- -f (bigwig file)
- -d (delimiter) (default:tab)
- -c (CHROM column=int) (default:1)
- -p (POS column=int) (default:2)
- -x (extend=int) extends window size (default:0)

Example: What's in the Big wig ?

```
$ kent/src/hg/encode/validateFiles/tests/test4.bw file.wig
$ cat file.wig

#bedGraph section chr1:1-1099
chr1    1       1000    54
chr1    1000    1099    53
```

let's get the intersection of a VCF file with this BIGWIG file.

```
$ echo -e "#CHROM\tPOS\nchr1\t500\nchr1\t1001"  |\
  vcfbigwig -f  kent/src/hg/encode/validateFiles/tests/test4.bw

#CHROM POS bigwig:min bigwig:max bigwig:mean bigwig:coverage
bigwig:stddev
chr1 500 54 54 54 1 0
chr1 1001 53 53 53 1 0

$ echo -e "#CHROM\tPOS\nchr1\t500\nchr1\t1001"  |\
  vcfbigwig -x 100 -f  kent/src/hg/encode/validateFiles/tests/test4.bw

#CHROM POS bigwig:min bigwig:max bigwig:mean bigwig:coverage
bigwig:stddev
```

```
chr1 500 54 54 54 1 0
chr1 1001 53 54 53.5025 0.99005 0.501255

$ echo -e "#CHROM\tPOS\nchrX\t500\nchrX\t1001"  |\
  vcfbigwig  -f  kent/src/hg/encode/validateFiles/tests/test4.bw

#CHROM POS bigwig:min bigwig:max bigwig:mean bigwig:coverage
bigwig:stddev
chrX 500 nan nan nan nan nan
chrX 1001 nan nan nan nan nan
```

3.20 vcfbigbed

Intersection VCF/BigBed

Options:

- -f (BigBed file)
- -d (delimiter) (default:tab)
- -c (CHROM column=int) (default:1)
- -p (POS column=int) (default:2)
- -x (extend=int) extends window size (default:0)
- -L (limit=int) limit to L records in bed (default:unbound)
- -S (NOT-FOUND-String) default:!N/A.
- -m (int=mode) 0)=all 1:only-matching 2:only-non-matching default:0.

Example: What's in the Big BED ?

```
$ cat test.bed
chr7    115000000       116000000       100.0
chr7    115500000       116500000       200.0
chr7    116000000       117000000       100.0
chr8    1000000         2000000         1000
```

```
chr8    1100000         1200000         1000
chr8    1100000         1200000         1000
chr9    100     200     10
chr9    100     300     10
chr9    100     400     10
chr9    1000    2000    10
chr9    1200    2000    10
chr9    1300    2000    10
```

let's get the intersection of a VCF file with this BigBed file.

```
$ echo -e "#CHROM\tPOS\nchr9\t1250\nchrX\t1"  |\
  vcfbigbed  -f  test.bb

#CHROM POS BigBed:chromStart BigBed:chromEnd BigBed:rest
chr9 1250 1000 2000 10
chr9 1250 1200 2000 10
chrX 1 !N/A !N/A !N/A

$ echo -e "#CHROM\tPOS\nchr9\t1250\nchrX\t1"  |\
  vcfbigbed  -x 1000 -f  test.bb

#CHROM POS BigBed:chromStart BigBed:chromEnd BigBed:rest
chr9 1250 100 300 10
chr9 1250 100 400 10
chr9 1250 1000 2000 10
chr9 1250 1200 2000 10
chr9 1250 1300 2000 10
chrX 1 !N/A !N/A !N/A
```

3.21 verticalize

Verticalize a table.

Options:

- -d or –delim (char) delimiter default:tab
- -n first line is NOT the header.

Example:

```
$ gunzip -c file.vcf.gz | grep -v "##" |\
   verticalize  |\
   head -n 100
>>>2
$1 #CHROM    1
$2 POS       880238
$3 ID        rs3748592
$4 REF       A
$5 ALT       G
$6 QUAL      48
$7 FILTER    0
$8 INFO      AC=2;DB=1;ST=0:0,3:4;DP=7;NC=-3.73;UM=3;CQ=INTRONIC;MQ=60;AN=2;P
$9 FORMAT    GT:GQ:DP:FLT
$10 162214_A1 1/1:48:7:0
<<<2

>>>3
$1 #CHROM    1
(...)
```

3.22 uniprot

Take as input a position on a protein and an uniprot ACN, connect to uniprot.org and answers wether a amino acid is contained in a 'feature'.

Options:

- -d (char) delimiter default:tab
- -p (column-index) column containing the amino acid index.
- -s (spId-index) column containing the swissprot-acn (e.g.: Q04721 or NOTC2_HUMAN).

Example:

```
$ echo -e "#POS\tID\n54\tQ04721\n1\tHELLO\n166\tP03536" |\
    uniprot -p 1 -s 2 |\
    verticalize
#warning: Cannot find record for HELLO
>>>2
$1 #POS             54
$2 ID               Q04721
$3 uniprot.beg      26
$4 uniprot.end      2471
$5 uniprot.type     chain
$6 uniprot.status   .
$7 uniprot.desc     Neurogenic locus notch homolog protein 2
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<2

>>>3
$1 #POS             54
$2 ID               Q04721
$3 uniprot.beg      26
$4 uniprot.end      1677
$5 uniprot.type     topological domain
$6 uniprot.status   potential
$7 uniprot.desc     Extracellular
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<3

>>>4
$1 #POS             54
$2 ID               Q04721
$3 uniprot.beg      26
$4 uniprot.end      63
$5 uniprot.type     domain
$6 uniprot.status   .
$7 uniprot.desc     EGF-like 1
$8 uniprot.evidence .
$9 uniprot.ref      .
```

```
<<<4

>>>5
$1 #POS             54
$2 ID               Q04721
$3 uniprot.beg      53
$4 uniprot.end      62
$5 uniprot.type     disulfide bond
$6 uniprot.status   by similarity
$7 uniprot.desc     .
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<5

>>>6
$1 #POS             1
$2 ID               HELLO
$3 uniprot.beg      .
$4 uniprot.end      .
$5 uniprot.type     .
$6 uniprot.status   .
$7 uniprot.desc     .
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<6

>>>7
$1 #POS             166
$2 ID               P03536
$3 uniprot.beg      1
$4 uniprot.end      315
$5 uniprot.type     chain
$6 uniprot.status   .
$7 uniprot.desc     Non-structural protein 3
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<7
```

```
>>>8
$1 #POS             166
$2 ID               P03536
$3 uniprot.beg      150
$4 uniprot.end      206
$5 uniprot.type     region of interest
$6 uniprot.status   .
$7 uniprot.desc     Dimerization
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<8

>>>9
$1 #POS             166
$2 ID               P03536
$3 uniprot.beg      166
$4 uniprot.end      237
$5 uniprot.type     coiled-coil region
$6 uniprot.status   potential
$7 uniprot.desc     .
$8 uniprot.evidence .
$9 uniprot.ref      .
<<<9
```

3.23 pfamscan

Take as input a position on a protein and an uniprot ACN, connect to pfam.sanger.ac.uk and answers wether a amino acid is contained in a 'match'.

Options:

- -d (char) delimiter default:tab
- -p (column-index) column containing the amino acid index.
- -a (acn) column containing the protein-acn (e.g.: Q04721 or IF4G1_HUMAN).

Example:

```
$ echo -e "#ACN\tPOS\nQ04721\t1650\nIF4G1_HUMAN\t1540\nQ04721\t300\nHello\t99
pfamscan -a 1 -p 2 |\
verticalize


>>>2
$1 #ACN       Q04721
$2 POS        1650
$3 pfam.beg   1617
$4 pfam.end   1677
$5 pfam.acn   PF07684
$6 pfam.id    NODP
$7 pfam.type  Pfam-A
$8 pfam.class .
<<<2

>>>3
$1 #ACN       IF4G1_HUMAN
$2 POS        1540
$3 pfam.beg   1518
$4 pfam.end   1599
$5 pfam.acn   PF02020
$6 pfam.id    W2
$7 pfam.type  Pfam-A
$8 pfam.class .
<<<3

>>>4
$1 #ACN       Q04721
$2 POS        300
$3 pfam.beg   298
$4 pfam.end   338
$5 pfam.acn   PF07645
$6 pfam.id    EGF_CA
$7 pfam.type  Pfam-A
$8 pfam.class .
```

```
<<<4

>>>5
$1 #ACN       Hello
$2 POS        9980
$3 pfam.beg   .
$4 pfam.end   .
$5 pfam.acn   .
$6 pfam.id    .
$7 pfam.type  .
$8 pfam.class .
<<<5

>>>6
$1 #ACN       .
$2 POS        .
$3 pfam.beg   .
$4 pfam.end   .
$5 pfam.acn   .
$6 pfam.id    .
$7 pfam.type  .
$8 pfam.class .
<<<6
```

3.24 vcf2bed

Generates a BED file from a VCF.

Options:

- -d (column-delimiter) (default:tab)
- -c (CHROM col) (default:0)
- -p (POS col) (default:1)
- -S (bed score col) (default:-1)
- -N (col) adds this column for the 'name'

- -D (char) name separator.
- -t print ucsc custom track header.

Example:

```
$ gunzip -c file.vcf.gz |\
  normalizechrom |\
  vcf2bed -t -S 6 -N 3,4,5 -D _

track name="__TRACK_NAME__" description="__TRACK_DESC__"
chr1 879316 879317 rs7523549_C_T 71 +
chr1 880237 880238 rs3748592_A_G 51 +
chr1 880389 880390 rs3748593_C_A 99 +
chr1 881626 881627 rs2272757_G_A 99 +
chr1 883624 883625 rs4970378_A_G 39 +
chr1 887559 887560 rs3748595_A_C 99 +
chr1 887800 887801 rs3828047_A_G 99 +
chr1 888638 888639 rs3748596_T_C 99 +
chr1 888658 888659 rs3748597_T_C 99 +
chr1 889157 889158 rs56262069_G_C 51 +
(...)
```

3.25 emblstringresolve

Calls the service: EMBL String resolve (`http://string-db.org/help/index.jsp?topic=/org.s`
).

Options:

- -d (char) delimiter default:tab
- -c column identifier
- -t (int) taxon id

Example:

```
$ echo -e "#Gene\nNOTCH2\nEIF4G1\nPABPC1" |\
  emblstringresolve -c 1 |verticalize
>>>2
$1 #Gene          NOTCH2
$2 stringId       9606.ENSP00000256646
$3 preferredName NOTCH2
$4 annotation     Notch homolog 2 (Drosophila); Functions as a receptor...
<<<2

>>>3
$1 #Gene          EIF4G1
$2 stringId       9606.ENSP00000316879
$3 preferredName EIF4G1
$4 annotation     eukaryotic translation initiation factor 4 gamma, 1; ...
<<<3

>>>4
$1 #Gene          PABPC1
$2 stringId       9606.ENSP00000313007
$3 preferredName PABPC1
$4 annotation     poly(A) binding protein, cytoplasmic 1; Binds the...
<<<4
```

3.26 emblstringinteractors

Calls the service: EMBL String interactors (`http://string-db.org/help/index.jsp?topic=/or`).

Options:

- -d (char) delimiter default:tab
- -c column identifier

Example:

```
$ echo -e "#Gene\nNOTCH2\nEIF4G1\nPABPC1" |\
  emblstringresolve -c 1 | \
  emblstringinteractors -c 2 | \
  emblstringresolve -c 5 | \
  verticalize

>>>2
$1 #Gene         NOTCH2
$2 stringId      9606.ENSP00000256646
$3 preferredName NOTCH2
$4 annotation    Notch homolog 2 (Drosophila); Functions as a receptor for me
$5 interactor    9606.ENSP00000256646
$6 stringId      9606.ENSP00000256646
$7 preferredName NOTCH2
$8 annotation    Notch homolog 2 (Drosophila); Functions as a receptor for me
<<<2

>>>3
$1 #Gene         NOTCH2
$2 stringId      9606.ENSP00000256646
$3 preferredName NOTCH2
$4 annotation    Notch homolog 2 (Drosophila); Functions as a receptor for me
$5 interactor    9606.ENSP00000345206
$6 stringId      9606.ENSP00000345206
$7 preferredName RBPJ
$8 annotation    recombination signal binding protein for immunoglobulin kapp
<<<3

>>>4
$1 #Gene         NOTCH2
$2 stringId      9606.ENSP00000256646
$3 preferredName NOTCH2
$4 annotation    Notch homolog 2 (Drosophila); Functions as a receptor for me
$5 interactor    9606.ENSP00000355718
$6 stringId      9606.ENSP00000355718
$7 preferredName DLL1
```

3.27 emblstringinteractions

Calls the service: EMBL String interactions (`http://string-db.org/help/index.jsp?topic=/o`).

Options:

- -d (char) delimiter default:tab
- -c column identifier

Example:

```
$ echo -e "#Gene\nNOTCH2\nEIF4G1\nPABPC1" | \
   emblstringresolve -c 1 | \
   emblstringinteractions -c 2 | \
   verticalize

>>>2
$1 #Gene           NOTCH2
$2 stringId        9606.ENSP00000256646
$3 preferredName   NOTCH2
$4 annotation      Notch homolog 2 (Drosophila); Functions as a receptor for
$5 interactorA     string:9606.ENSP00000355718
$6 interactorB     string:9606.ENSP00000326366
$7 labelA          DLL1
$8 labelB          PSEN1
$9 aliasesA        -
$10 aliasesB        -
$11 method          -
$12 firstAuthor     -
$13 publication     -
$14 taxonA          taxid:9606
$15 taxonB          taxid:9606
$16 types           -
$17 sources         -
$18 interaction.ids -
$19 score           score:0.999|escore:0.639|dscore:0.9|tscore:0.984
<<<2
```

```
>>>3
$1 #Gene            NOTCH2
$2 stringId         9606.ENSP00000256646
$3 preferredName    NOTCH2
$4 annotation       Notch homolog 2 (Drosophila); Functions as a receptor for
$5 interactorA      string:9606.ENSP00000345206
$6 interactorB      string:9606.ENSP00000292599
$7 labelA           RBPJ
```

3.28 vcfcut

Simple "cut this region".

```
$ curl -s "ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20100804/ALL.
   gunzip -c | grep -v "##" |\
   vcfcut -e '2:10kb+500bp;1:10000-20000'

#CHROM POS ID REF ALT QUAL FILTER INFO
1 10327 rs112750067 T C . PASS DP=65;AF=0.208;CB=BC,NCBI
1 10469 rs117577454 C G . PASS DP=2055;AF=0.020;CB=UM,BC,NCBI
1 10492 rs55998931 C T . PASS DP=231;AF=0.167;CB=BC,NCBI
(...)
1 16841 . G T . PASS DP=2906;AF=0.004;CB=UM,BI;EUR_R2=0.248
2 10038 . C A . PASS DP=73;AF=0.409;CB=BC,NCBI
2 10075 . C A . PASS DP=31;AF=0.150;CB=BC,NCBI
2 10144 . C A . PASS DP=33;AF=0.562;CB=BC,NCBI
2 10159 . C A . PASS DP=32;AF=0.222;CB=BC,NCBI
2 10205 . T G . PASS DP=582;AF=0.107;CB=UM,BC
2 10297 . G T . PASS DP=500;AF=0.246;CB=UM,BC
2 10363 . G A . PASS DP=788;AF=0.016;CB=UM,BI;EUR_R2=0.273;AFR_R2=0.034
2 10437 rs71337607 C T . PASS DP=324;AF=0.267;CB=UM,BC
```

3.29 ucscgenesps

Reads an ordered VCF (ordered by CHROM/POS and optionaly by SAMPLE) , connect to an UCSC database a print the variation in postscript.

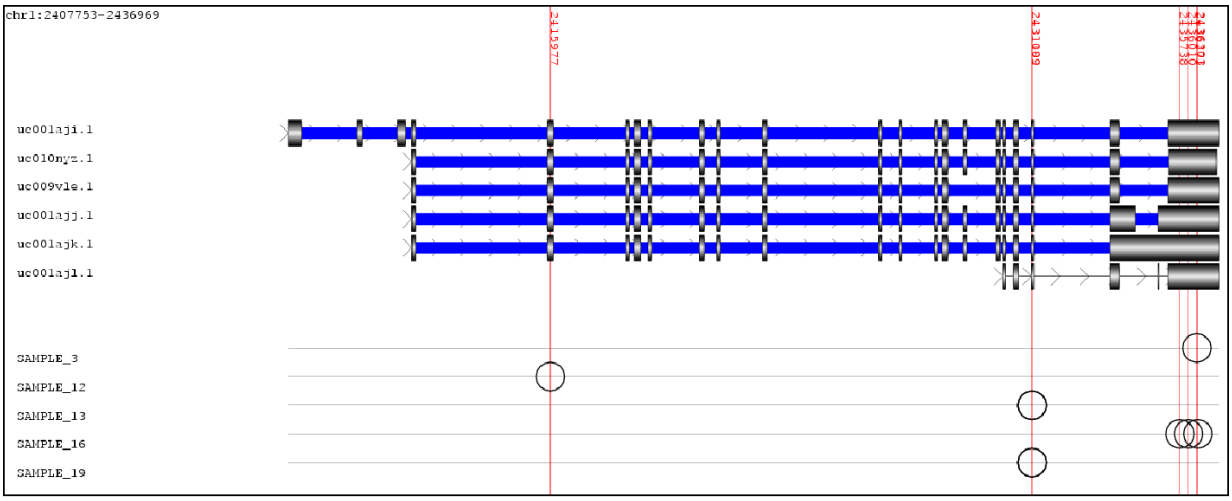

Figure 3: Structure of a gene

Options:

- -c (int) CHROM column default:1
- -p (int) POS column default:2
- -s (int) SAMPLE column (optional)
- -r (int) COLOR column (optional)
- –host (mysql host) default:genome-mysql.cse.ucsc.edu
- –user (mysql user) default:genome
- –password (mysql password) default:
- –database (mysql database) default:hg19
- –port (mysql password) default:0

Example:

```
$ cat sample2vcf.tsv | tr -d ' ' |\
  scanvcf |\
  awk -F '	' '($3==".")' |\
```

```
normalizechrom |\
sort -t '  ' -k1,1 -k2,2n -k11,11 |\
head -n 10000 |\
ucscgenesps --host localhost --user username --port 3316 -s 11 > result.ps
```

3.30 vcfintersect

Compute the intersection for an ordered VCF (ordered by CHROM/POS) with another source ordered by CHROM/POS.

Options:

- -f ¡external url/file¿ [required]
- -m (mode) 0:all 1:matching 2:unmatching default:0
- -n (string) no-match string default: NO_MATCH
- -c1 (CHROM col) (default:1)
- -s1 (START col) (default:2)
- -e1 (END col) (default:2)
- -d1 (delimiter) (default:tab)
- -h1 toggle: input is half open (default:0)
- -z1 toggle: input zero-based (default:0)
- -c2 (CHROM col) (default:1)
- -s2 (START col) (default:2)
- -e2 (END col) (default:2)
- -d2 (delimiter) (default:tab)
- -h2 toggle: input is half open (default:1)
- -z2 toggle: input zero-based (default:1)
- –http force database is a URL
- –gunzip force database is a gzipped stream

Example: annotate a VCF with the data from snp132 at the UCSC.

```
$ echo -e "#CHROM\tPOS\nchr1\t10519\nchr1\t10520\nchr1\t10828"|\
vcfintersect -n NO\_MATCH -c2 2 -s2 3 -e2 4  \
-f "http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/snp132.txt.gz" |\
verticalize -n

>>>1
$1   #CHROM
$2   POS
<<<1

>>>2
$1   chr1
$2   10519
$3   585
$4   chr1
$5   10518
$6   10519
$7   rs62636508
$8   0
$9   +
$10  G
$11  G
$12  C/G
$13  genomic
$14  single
$15  by-1000genomes
$16  0
$17  0
$18  unknown
$19  exact
$20  1
$21
$22  2
$23  1000GENOMES,BCMHGSC_JDW,
$24  2
$25  G,C,
```

```
$26  112.000000,8.000000,
$27  0.933333,0.066667,
$28
<<<2

>>>3
$1   chr1
$2   10520
$3   NO_MATCH
<<<3

>>>4
$1   chr1
$2   10828
$3   585
$4   chr1
$5   10827
$6   10828
$7   rs10218492
$8   0
$9   +
$10  G
$11  G
$12  A/G
$13  genomic
$14  single
$15  by-cluster
$16  0
$17  0
$18  unknown
$19  exact
$20  1
$21
$22  1
$23  WUGSC_SSAHASNP,
$24  0
$25
$26
```

```
$27
$28
<<<4
```

3.31 igvcontrol

simple curses-based VCF browser. It controls IGV `http://www.broadinstitute.org/igv/` to change the visualization to a defined view.

Options:

- -c (CHROM col) (default:1)
- -p (START col) (default:2)

Example:

3.32 vcfliftover

Use the UCSC C API to process the data with 'liftOver'

Options:

- -d (char) column delimiter. default: TAB
- -c (int) chromosome column (1).
- -p (int) pos column (2).
- -1 data are NOT +1 based.
- -f (path) liftOver map file (required).
- -b (double) liftOver minblocks.
- -m (double) liftOver minMatch.

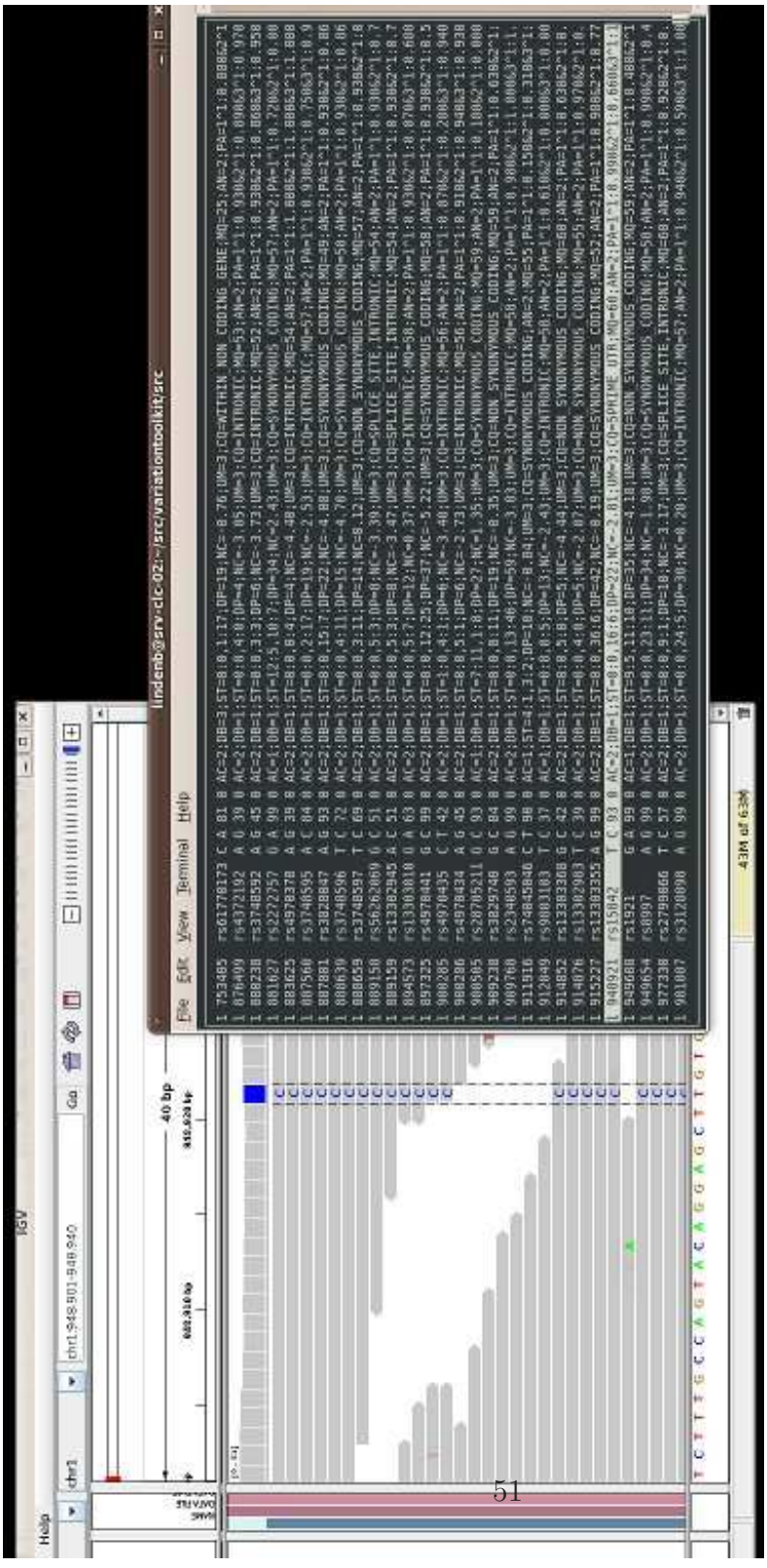

Figure 4: Screenshot IGV and igvcontrol

Example: Download the data of snp129 from UCSC hg18, remove some columns and convert to hg19.

```
$ curl  -s "http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/snp129.tx
  gunzip  -c |\
  cut -d '  ' -f 2,3,5 |\
  vcfliftover -1 -f /path/tp/hg18ToHg19.over.chain

chr1 433 rs56289060 chr1 10433 10434 .
chr1 491 rs55998931 chr1 10491 10492 .
chr1 518 rs62636508 chr1 10518 10519 .
chr1 582 rs58108140 chr1 10582 10583 .
chr1 690 rs10218492 chr1 10827 10828 .
chr1 766 rs10218493 chr1 10903 10904 .
chr1 789 rs10218527 chr1 10926 10927 .
chr1 800 rs28853987 chr1 10937 10938 .
chr1 876 rs28484712 chr1 11013 11014 .
chr1 884 rs28775022 chr1 11021 11022 .
(...)
chr1 1609710 rs61776794 . . . Deleted in new
chr1 1609743 rs61776795 . . . Deleted in new
chr1 1609758 rs61776796 . . . Deleted in new
chr1 1609849 rs7413891 . . . Deleted in new
chr1 1610719 rs3737622 . . . Deleted in new
chr1 1610719 rs45576038 . . . Deleted in new
chr1 1610763 rs3737624 . . . Deleted in new
chr1 2475133 rs3091278 . . . Deleted in new
chr1 2475134 rs3091239 . . . Deleted in new
(...)
```

3.33 backlocate

convert a protein variation to a genomic position.

Options:

- -g (column) gene name default:1
- -m (column) mutation in protein default:2

- -f (pasta to fasta reference indexed with faidx).
- -p print sequences.
- -d delimiter. Default:tab
- –host (mysql host) default:genome-mysql.cse.ucsc.edu
- –user (mysql user) default:genome
- –password (mysql password) default:
- –database (mysql database) default:hg19
- –port (mysql password) default:0

```
echo -e  "NOTCH2\tM1T\nEIF4G1\tD240Y" |\
    backlocate -f /path/to/hg19.fa

#User.Gene AA1 petide.pos.1 AA2 knownGene.name knownGene.strand
knownGene.AA index0.in.rna codon base.in.rna chromosome index0.in.genomic
exon
##uc001eik.2
NOTCH2 M 1 T uc001eik.2 -M 0 ATG A chr1 120612019 Exon 1
NOTCH2 M 1 T uc001eik.2 -M 1 ATG T chr1 120612018 Exon 1
NOTCH2 M 1 T uc001eik.2 -M 2 ATG G chr1 120612017 Exon 1
##uc001eil.2
NOTCH2 M 1 T uc001eil.2 -M 0 ATG A chr1 120612019 Exon 1
NOTCH2 M 1 T uc001eil.2 -M 1 ATG T chr1 120612018 Exon 1
NOTCH2 M 1 T uc001eil.2 -M 2 ATG G chr1 120612017 Exon 1
##uc001eim.3
NOTCH2 M 1 T uc001eim.3 -M 0 ATG A chr1 120548116 Exon 2
NOTCH2 M 1 T uc001eim.3 -M 1 ATG T chr1 120548115 Exon 2
NOTCH2 M 1 T uc001eim.3 -M 2 ATG G chr1 120548114 Exon 2
##Warning ref aminod acid for uc003fnp.2  [240] is not the same (I/D)
EIF4G1 D 240 Y uc003fnp.2 + I 717 ATC A chr3 184039089 Exon 10
EIF4G1 D 240 Y uc003fnp.2 + I 718 ATC T chr3 184039090 Exon 10
EIF4G1 D 240 Y uc003fnp.2 + I 719 ATC C chr3 184039091 Exon 10
##Warning ref aminod acid for uc003fnu.3  [240] is not the same (I/D)
EIF4G1 D 240 Y uc003fnu.3 + I 717 ATC A chr3 184039089 Exon 9
```

```
EIF4G1 D 240 Y uc003fnu.3 + I 718 ATC T chr3 184039090 Exon 9
EIF4G1 D 240 Y uc003fnu.3 + I 719 ATC C chr3 184039091 Exon 9
##Warning ref aminod acid for uc003fnq.2  [240] is not the same (V/D)
EIF4G1 D 240 Y uc003fnq.2 + V 717 GTA G chr3 184039350 Exon 7
EIF4G1 D 240 Y uc003fnq.2 + V 718 GTA T chr3 184039351 Exon 7
EIF4G1 D 240 Y uc003fnq.2 + V 719 GTA A chr3 184039352 Exon 7
##Warning ref aminod acid for uc003fnr.2  [240] is not the same (L/D)
EIF4G1 D 240 Y uc003fnr.2 + L 717 CTC C chr3 184039581 Exon 6
EIF4G1 D 240 Y uc003fnr.2 + L 718 CTC T chr3 184039582 Exon 6
EIF4G1 D 240 Y uc003fnr.2 + L 719 CTC C chr3 184039583 Exon 6
##Warning ref aminod acid for uc003fny.3  [240] is not the same (T/D)
EIF4G1 D 240 Y uc003fny.3 + T 717 ACC A chr3 184039677 Exon 3
EIF4G1 D 240 Y uc003fny.3 + T 718 ACC C chr3 184039678 Exon 3
EIF4G1 D 240 Y uc003fny.3 + T 719 ACC C chr3 184039679 Exon 3
##uc010hxx.2
EIF4G1 D 240 Y uc010hxx.2 + D 717 GAT G chr3 184038780 Exon 10
EIF4G1 D 240 Y uc010hxx.2 + D 718 GAT A chr3 184039069 Exon 11
EIF4G1 D 240 Y uc010hxx.2 + D 719 GAT T chr3 184039070 Exon 11
##Warning ref aminod acid for uc003fns.2  [240] is not the same (L/D)
EIF4G1 D 240 Y uc003fns.2 + L 717 CTC C chr3 184039209 Exon 10
EIF4G1 D 240 Y uc003fns.2 + L 718 CTC T chr3 184039210 Exon 10
EIF4G1 D 240 Y uc003fns.2 + L 719 CTC C chr3 184039211 Exon 10
(...)
```

3.34 genomesim

Generates two mutated homologous sequences from a fasta file.

Options:

- -o (file.tar) output tar file. contains chromosomes and mutations.
- -f (file) limit by genomic region (optional) read file:chrom(TAB)start(TAB)end
- -i (file) no mutation in those genomic regions (optional) read file:chrom(TAB)start(TAB)end
- -r (float) rate of mutations. default: 0.001
- -R (float) fraction of indels default: 0.1

- -X (float) probability an indel is extended default: 0.1
- -u (filename) read a file containing user-defined mutations (optional). Format: (CHROM)
  t(POS+1)
  t(BASE1)
  t(BASE2)
- -m (chrom) (POS+1) (BASE1) (BASE2) insert user defined substitution. use dot('.') to not change the base.

```
$ genomesim -o chrom.tar -m chrM 10 A T -f regions.bed chrM.fa
$ tar tvf chrom.tar
-rw-r--r-- 0/0           16795 2011-12-09 13:56 chrom/homologous1.fa
-rw-r--r-- 0/0           16793 2011-12-09 13:56 chrom/homologous2.fa
-rw-r--r-- 0/0             203 2011-12-09 13:56 chrom/mutations.txt
```

3.35 bam2wig

Creates a WIG file for the coverage of a BAM file.

Options:

- -z (int) number of depth=0 accepted before starting a new WIG file
- -o (filename-out) save as... (default:stdout).
- -t print a ucsc custom track header.

```
$ bam2wig -t  file.bam

track name="__TRACK_NAME__" description="__TRACK_DESC__" type="wiggle_0"
fixedStep chrom=chrM start=23 step=1 span=1
2
2
3
4
5
6
6
```

```
4
(...)
```

3.36 ttmap

prints an ASCII genomic map.

Options:

- -c (int) chrom column
- -s (int) start column
- -e (int) end column default: start column
- -o (int) strand column default (optional)
- -n (int) name column default (optional)
- -d (char) delimiter default:tab
- -C (int) fix the number of output columns.

```
$ curl -s "http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/knownGene.
  gunzip -c |  grep chrM |\
  ./ttmap -c 2 -s 4 -e 5 -o 3 -n 1 -C 50

>chrM:236-15998
<              .              .              .     uc004coq.3
    >          .              .              .     uc004cor.1
    >          .              .              .     uc004cos.3
               . >            .              .     uc011mfh.1
               .       >      .              .     uc004cou.3
               .          >   .              .     uc011mfi.1
               .              . >            .     uc004cov.3
               .              .      >       .     uc004cow.1
               .              .       >      .     uc004cox.3
               .              .              .>    uc004coy.2
```

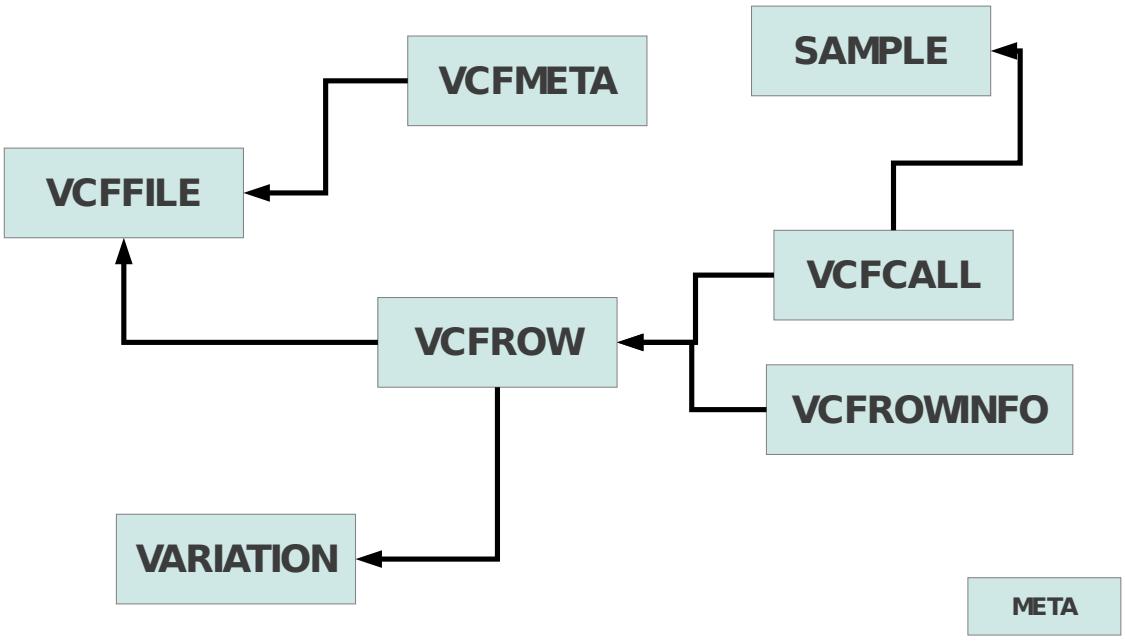

Figure 5: Sqlite schema

3.37 vcf2sqlite

Inserts a VCF in a sqlite3 database.

Options:

- -f (file) sqlite database (REQUIRED).

Schema:

Example:

```
$ vcf2sqlite -f db.sqlite file.vcf
$ sqlite3 -line db.sqlite  "select * from VCFCALL LIMIT 4"

        id = 1
    nIndex = 0
vcfrow_id = 1
sample_id = 1
```

```
      prop = GT
     value = 1/1

        id = 2
    nIndex = 1
vcfrow_id = 1
sample_id = 1
      prop = PL
     value = 46,6,0

        id = 3
    nIndex = 2
vcfrow_id = 1
sample_id = 1
      prop = GQ
     value = 10

        id = 4
    nIndex = 0
vcfrow_id = 2
sample_id = 1
      prop = GT
     value = 1/1
```

3.38 vcf2xml

Transforms a VCF to xml.

Example:

```
$ vcf2xml input.vcf | xmllint --format -

<?xml version="1.0" encoding="UTF-8"?>
<vcf>
  <head>
    <meta key="fileformat">VCFv4.1</meta>
    <meta key="samtoolsVersion">0.1.17 (r973:277)</meta>
```

```
    <infos>
      <info>
        <id>DP</id>
        <number>1</number>
        <type>Integer</type>
        <description>Raw read depth</description>
      </info>
      <info>
        <id>DP4</id>
        <number>4</number>
        <type>Integer</type>
        <description># high-quality ref-forward bases</description>
      </info>
      <info>
        <id>MQ</id>
(...)
      </calls>
    </variation>
    <variation>
      <chrom>chr1</chrom>
      <pos>112697</pos>
      <ref>T</ref>
      <alt>G</alt>
      <qual>10.4</qual>
      <infos>
        <info key="DP">1</info>
        <info key="AF1">1</info>
        <info key="AC1">2</info>
        <info key="DP4">0,0,0,1</info>
        <info key="MQ">60</info>
        <info key="FQ">-30</info>
      </infos>
      <calls>
        <call sample="input.bam">
          <prop key="GT">1/1</prop>
          <prop key="PL">40,3,0</prop>
          <prop key="GQ">5</prop>
        </call>
```

```
      </calls>
    </variation>
  </body>
</vcf>
```

3.39 ngsproject

A Web-Application (CGI) displaying a BAM alignment using the Samtools API.

Install on apache 2: add the path to th cgi directory in the file variation-toolkit/config.mk

```
CGI_BIN_DIR=/var/www/cgi-bin
```

Create the cgi-bin folder is needed

```
sudo mkdir -p /var/www/cgi-bin
sudo chmod 755 /var/www/cgi-bin
```

Create a XML file describing your project

```
cat /var/www/cgi-bin/ngsproject.xml
```

```
<?xml version="1.0"?>
<!DOCTYPE projects [
<!ENTITY samdir "/tmp">
]>
<projects>
  <reference id="ref1">
    <name>Samtools1</name>
    <description>Samtools example 1</description>
    <path>&samdir;/examples/ex1.fa</path>
  </reference>
  <bam id="b1">
    <sample>Huey</sample>
    <path>&samdir;/examples/ex1.bam</path>
```

```
  </bam>
  <bam id="b2">
    <sample>Dewey</sample>
    <path>&samdir;/examples/ex1.bam</path>
  </bam>
  <bam id="b3">
    <sample>Louie</sample>
    <path>&samdir;/examples/ex1.bam</path>
  </bam>
  <project id="p1">
    <name>Project P1</name>
    <description>This is my 1st project</description>
    <bam ref="b1"/>
    <bam ref="b2"/>
    <bam ref="b3"/>
    <reference ref="ref1"/>
  </project>
  <project id="p2">
    <name>Project P2</name>
    <description>This is my 2nd project</description>
    <bam ref="b1"/>
    <bam ref="b3"/>
    <reference ref="ref1"/>
  </project>
</projects>
```

Edit the apache2 config /etc/apache2/apache2.conf

```
sudo nano /etc/apache2/apache2.conf
```

and add the following lines, specify NGS_PROJECT_PATH, the path to the XML file

```
<VirtualHost *:80>
ServerName localhost
DocumentRoot /var/www/
AddHandler cgi-script .cgi .pl
SetEnv NGS_PROJECT_PATH /var/www/cgi-bin/ngsproject.xml
<Directory /cgi-bin/>
```

```
        AllowOverride None
        Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
</Directory>

</VirtualHost>
```

and restart apache:

```
sudo /etc/init.d/apache2 restart
```

Example:

3.40 fastaslice

Slice some FASTA sequences.

Options:

- -e (every) default:0
- -L (fragment size) default: (same as -e)

Example:

```
$ ../bin/fastaslice -e 10 -L 20 < nsp3.fasta | head

>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
MLKMESTQQMASSIINTSFE
>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
ASSIINTSFEAAVVAATSTL
>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
AAVVAATSTLELMGIQYDYN
>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
ELMGIQYDYNEIYTRVKSKF
>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
EIYTRVKSKFDYVMDDSGVK
```

Figure 6: Screenshot of ngsproject

3.41 fastasortuniq

sort/uniq on FASTA sequences.

Options:

- -u uniq
- -i ignore case

Example:

```
$ fastaslice -e 1 -L 3 < nsp3.fasta |\
  fastasortuniq -i -u | head

>gi|77020788|gb|ABA60396.1| non-structural protein NSP3 [Human rotavirus B219
AAF
>gi|288187218|gb|ADC42131.1| translation enhancer NSP3 [Bovine rotavirus A]|s
AAK
>gi|110558644|gb|ABG75781.1| NSP3 [Rotavirus A]|slice:20-23
AAL
>gi|284517165|gb|ADB92082.1| NSP3 [Human rotavirus A]|slice:48-51
AAR
>gi|256041817|gb|ACU64749.1| NSP3 protein [Rotavirus A AU32xUK reassortant (U
AAT
```

3.42 fastasortuniq

reverse complement FASTA sequences.

Options:

- -p print original sequence.
- -c DISABLE complement.
- -r DISABLE reverse.

Example:

```
$ fastaslice -e 1 -L 20 < rotavirus.fasta  | fastarevcomp  -p | head
>gi|27592135|slice:0-20
GGAAGGGCTGCCCCACCATT
>gi|27592135|slice:0-20|reverse-complement
AATGGTGGGGCAGCCCTTCC
>gi|27592135|slice:1-21
GAAGGGCTGCCCCACCATTC
>gi|27592135|slice:1-21|reverse-complement
GAATGGTGGGGCAGCCCTTC
>gi|27592135|slice:2-22
AAGGGCTGCCCCACCATTCA
```

3.43 fastatail

prints the last sequences of a list of FASTA sequences.

Options:

- -n number of sequences

Example:

```
$ fastaslice -e 50 -L 50 < rotavirus.fasta  |fastatail -n 2

>gi|14662886|slice:200-250
CATCATTTCATACCATCATATCGGCATCAATCAAAATGGTCCCATGACTT
>gi|14662886|slice:250-275
TTTGTAACCGGCCCCCCTTAAAACT
```

3.44 fasta2tsv

prints Fasta Sequences as Tab delimited values.

Options:

- -d ¡char¿ delimiter. default: tab
- -u convert sequence to upper case

Example:

```
$ fastaslice -e 50 -L 50 < rotavirus.fasta  | fastatail -n 5 | fasta2tsv

gi|14662886|slice:50-100 CACTCCTTTCACAAATCCCGAATTCTCTATCTAACTAACATTTGGCATAT
gi|14662886|slice:100-150 CAGGTTGCCCTTCTCTCAGCGCCAGTTACAGGCCCATTTCCCAGTCAAGT
gi|14662886|slice:150-200 CCTATTCCGCGCTCAGGTATATCTTTTCAACCCATCAATATTGCAGCCTT
gi|14662886|slice:200-250 CATCATTTCATACCATCATATCGGCATCAATCAAAATGGTCCCATGACTT
gi|14662886|slice:250-275 TTTGTAACCGGCCCCCCTTAAAACT
```

3.45 fastatac

Reverse the order of some fasta sequences.

Example:

```
$ fastaslice -e 50 -L 50 < rotavirus.fasta  | fastatac  | head

>gi|14662886|slice:250-275
TTTGTAACCGGCCCCCCTTAAAACT
>gi|14662886|slice:200-250
CATCATTTCATACCATCATATCGGCATCAATCAAAATGGTCCCATGACTT
>gi|14662886|slice:150-200
CCTATTCCGCGCTCAGGTATATCTTTTCAACCCATCAATATTGCAGCCTT
>gi|14662886|slice:100-150
CAGGTTGCCCTTCTCTCAGCGCCAGTTACAGGCCCATTTCCCAGTCAAGT
>gi|14662886|slice:50-100
CACTCCTTTCACAAATCCCGAATTCTCTATCTAACTAACATTTGGCATAT
```

3.46 ncbielink

Retrieves related NCBI records using NCBI-ELink

Options:

- -f (ncbi-database-from) REQUIRED.
- -t (ncbi-database-to) REQUIRED.

- -d (delimiter) (default:tab)
- -c (column) index of column containing gi identifier for database-from.

Example:

```
$ echo -e "#Genbank_GI\n256041817" | ncbielink  -c 1 -f protein -t nucleotide

#Genbank_GI protein:nucleotide.linkName protein:nucleotide.id
256041817 protein_nuccore 256041816
256041817 protein_nuccore_mrna 256041816
```

3.47 ncbicited

Retrieves NCBI/Pubmed records citing a pmid using NCBI-ELink.

Options:

- -p show pmids.
- -c (column) index of column containing gi identifier for database-from.

Example:

```
$ echo -e "#query\nNOTCH2"  |\
  ncbiesearch  -q 'Caignec $1' | \
  ncbicited -p -c 2

#query pubmed.id count.citing citing.pmid
NOTCH2 21793104 . .
NOTCH2 21718643 . .
NOTCH2 21378989 2 21984761
NOTCH2 21378989 2 22046109
```

References

[1] H. Li. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics*, 27(5):718–9, Mar. 2011.

[2] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. D. and. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–9, Aug. 2009.

[3] P. Lindenbaum, S. L. Scouarnec, V. Portero, and R. Redon. Knime4Bio: a set of custom nodes for the interpretation of Next Generation Sequencing data with KNIME. *Bioinformatics*, Oct. 2011.