RBcf: An VCF API for R.

Pierre Lindenbaum / @yokofakun/ Institut du Thorax . Nantes.

April 24, 2020

1 Abstract

RBcf uses the Htslib C API for parsing VCF and BCF files. This API was written by a regular user of the htsjdk library who doesn't like R.

A list of functions is available at: https://github.com/lindenb/rbcf/blob/master/R/rbcf.R

2 Examples

2.1 Htslib and Rbcf versions

Code:

```
# load the library
library(rbcf)
#print the version of the associated htslib
htslib.version()
#print the version of rbcf
rcbf.version()
```

Output:

```
> # load the library
> library(rbcf)
> #print the version of the associated htslib
> htslib.version()
[1] "1.10.2-dirty"
> #print the version of rbcf
> rcbf.version()
[1] "0.0-1"
>
```

2.2 Open and close a VCF file

Code:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
# dispose the vcf reader
bcf.close(fp)
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
>
```

2.3 Print the INFOs in the VCF header

Code:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
info <- bcf.infos(fp)
# dispose the vcf reader
bcf.close(fp)
# print the table
info
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
> info <- bcf.infos(fp)
> # dispose the vcf reader
> bcf.close(fp)
```

```
[1] TRUE
> # print the table
> info
         ID Number    Type
INDEL INDEL      0    Flag
IDV     IDV      1 Integer
IMF     IMF      1   Float
DP       DP      1 Integer
VDB     VDB      1   Float
RPB     RPB      1   Float
MQB     MQB      1   Float
BQB     BQB      1   Float
MQSB   MQSB      1   Float
SGB     SGB      1   Float
MQOF   MQOF      1   Float
ICB     ICB      1   Float
HOB     HOB      1   Float
AC       AC      A Integer
AN       AN      1 Integer
DP4     DP4      4 Integer
MQ       MQ      1 Integer

INDEL                                                          "Indicates␣that␣the␣v
IDV                                                          "Maximum␣number␣of␣reads␣
IMF                                                        "Maximum␣fraction␣of␣reads␣
DP
VDB   "Variant␣Distance␣Bias␣for␣filtering␣splice-site␣artefacts␣in␣RNA-seq␣data␣
RPB                                      "Mann-Whitney␣U␣test␣of␣Read␣Position␣Bias␣
MQB                                    "Mann-Whitney␣U␣test␣of␣Mapping␣Quality␣Bias␣
BQB                                      "Mann-Whitney␣U␣test␣of␣Base␣Quality␣Bias␣
MQSB                         "Mann-Whitney␣U␣test␣of␣Mapping␣Quality␣vs␣Strand␣Bias␣
SGB                                                                        "Segreg
MQOF                                                     "Fraction␣of␣MQ0␣reads␣
ICB                                             "Inbreeding␣Coefficient␣Binomial␣test␣
HOB                                               "Bias␣in␣the␣number␣of␣HOMs␣number␣
AC                        "Allele␣count␣in␣genotypes␣for␣each␣ALT␣allele,␣in␣the␣s
AN                                                   "Total␣number␣of␣alleles␣
DP4              "Number␣of␣high-quality␣ref-forward␣,␣ref-reverse,␣alt-forward␣an
MQ                                                                        "Aver
>
```

2.4 Print the FORMATs in the VCF header

Code:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
fmts <- bcf.formats(fp)
# dispose the vcf reader
bcf.close(fp)
# print the table
fmts
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
> fmts <- bcf.formats(fp)
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # print the table
> fmts
   ID Number    Type                                      Description
PL PL      G Integer "List␣of␣Phred-scaled␣genotype␣likelihoods"
GT GT      1  String                                       "Genotype"
>
```

2.5 Print the FILTERs in the VCF header

Code:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/gnomad.exomes.r2.0.1.sites.vcf",FALSE)
flt <- bcf.filters(fp)
# dispose the vcf reader
bcf.close(fp)
# print the table
flt
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/gnomad.exomes.r2.0.1.sites.vcf",FALSE)
> flt <- bcf.filters(fp)
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # print the table
> flt
                              ID
PASS                        PASS
AC0                          AC0
InbreedingCoeff InbreedingCoeff
LCR                          LCR
RF                            RF
SEGDUP                    SEGDUP

PASS
AC0             "Allele␣Count␣is␣zero␣(i.e.␣no␣high-confidence␣genotype␣(GQ␣>=␣2
InbreedingCoeff
LCR
RF                                                  "Failed␣random␣forests␣filte
SEGDUP
>
```

2.6 Print the Samples in the VCF header

The samples are defined in the '#CHROM' line of the VCF **Code**:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
# print the number of samples
bcf.nsamples(fp)
# get the name for the 1st sample
bcf.sample.at(fp,1)
# get the 1-based index for the samples
bcf.sample2index(fp,c("S1","S2","S3","missing"))
```

```
# get all the samples
bcf.samples(fp)
# dispose the vcf reader
bcf.close(fp)
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
> # print the number of samples
> bcf.nsamples(fp)
[1] 5
> # get the name for the 1st sample
> bcf.sample.at(fp,1)
[1] "S1"
> # get the 1-based index for the samples
> bcf.sample2index(fp,c("S1","S2","S3","missing"))
     S1      S2      S3 missing
      1       2       3       0
> # get all the samples
> bcf.samples(fp)
[1] "S1" "S2" "S3" "S4" "S5"
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
>
```

2.7 Print the Dictionary in the VCF header

Code:

```
# load rbcf
library(rbcf)
# we don't need the index for this file
fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
dict <- bcf.dictionary(fp)
# dispose the vcf reader
bcf.close(fp)
# print the table
dict
```

Output:

```
> # load rbcf
> library(rbcf)
> # we don't need the index for this file
> fp <- bcf.open("../tests/data/rotavirus_rf.01.vcf",FALSE)
> dict <- bcf.dictionary(fp)
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # print the table
> dict
     chrom size
RF01  RF01 3302
RF02  RF02 2687
RF03  RF03 2592
RF04  RF04 2362
RF05  RF05 1579
RF06  RF06 1356
RF07  RF07 1074
RF08  RF08 1059
RF09  RF09 1062
RF10  RF10  751
RF11  RF11  666
>
```

2.8 Print the Indexed Chromosomes

Code:

```
# load rbcf
library(rbcf)
# Open the indexed VCF
fp <- bcf.open("../tests/data/rotavirus_rf.02.vcf.gz")
# get the indexed contigs
contigs <- bcf.contigs(fp)
# dispose the vcf reader
bcf.close(fp)
# print the table
contigs
```

Output:

```
> # load rbcf
> library(rbcf)
> # Open the indexed VCF
> fp <- bcf.open("../tests/data/rotavirus_rf.02.vcf.gz")
> # get the indexed contigs
> contigs <- bcf.contigs(fp)
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # print the table
> contigs
 [1] "RF01" "RF02" "RF03" "RF04" "RF05" "RF06" "RF07" "RF08" "RF09" "RF10"
[11] "RF11"
>
```

2.9 Scanning the variants

Code:

```
# load rbcf
library(rbcf)

# create a function counting variants in a VCF
count.variants<-function(filename) {
        # we don't need the index for this file
        fp <- bcf.open(filename,FALSE)
        # number of variants
        n<-0
        # loop while we can read a variant
        while(!is.null(vc<-bcf.next(fp))) {
                # increment the count
                n<-n+1
        }
        # dispose the vcf reader
        bcf.close(fp)
        # return the number of variant
        n
}

# filenames
vcfs<-c(
```

```
        "../tests/data/gnomad.exomes.r2.0.1.sites.vcf",
        "../tests/data/rotavirus_rf.01.vcf",
        "../tests/data/rotavirus_rf.02.vcf.gz",
        "../tests/data/rotavirus_rf.03.vcf.gz",
        "../tests/data/rotavirus_rf.04.bcf"
        )
# print the number of variants for each vcf
for(f in vcfs) {
        cat(paste(f,"␣",count.variants(f),"\n"))
        }
```

Output:

```
> # load rbcf
> library(rbcf)
>
> # create a function counting variants in a VCF
> count.variants<-function(filename) {
+       # we don't need the index for this file
+       fp <- bcf.open(filename,FALSE)
+       # number of variants
+       n<-0
+       # loop while we can read a variant
+       while(!is.null(vc<-bcf.next(fp))) {
+               # increment the count
+               n<-n+1
+       }
+       # dispose the vcf reader
+       bcf.close(fp)
+       # return the number of variant
+       n
+ }
>
> # filenames
> vcfs<-c(
+       "../tests/data/gnomad.exomes.r2.0.1.sites.vcf",
+       "../tests/data/rotavirus_rf.01.vcf",
+       "../tests/data/rotavirus_rf.02.vcf.gz",
+       "../tests/data/rotavirus_rf.03.vcf.gz",
+       "../tests/data/rotavirus_rf.04.bcf"
+       )
> # print the number of variants for each vcf
```

```
> for(f in vcfs) {
+       cat(paste(f,"␣",count.variants(f),"\n"))
+       }
../tests/data/gnomad.exomes.r2.0.1.sites.vcf   50
../tests/data/rotavirus_rf.01.vcf   45
../tests/data/rotavirus_rf.02.vcf.gz   45
../tests/data/rotavirus_rf.03.vcf.gz   45
../tests/data/rotavirus_rf.04.bcf   45
>
```

2.10 Scanning the variants

Code:

```
# load rbcf
library(rbcf)

# create a function counting variants in a VCF
count.variants<-function(filename,predicate) {
        # we don't need the index for this file
        fp <- bcf.open(filename,FALSE)
        # number of variants
        n<-0
        # loop while we can read a variant
        while(!is.null(vc<-bcf.next(fp))) {
                # test the variant
                if(predicate(vc)) {
                        # increment the count
                        n<-n+1
                        }
        }
        # dispose the vcf reader
        bcf.close(fp)
        # return the number of variant
        n
}

# A vcf
filename <- "../tests/data/gnomad.exomes.r2.0.1.sites.vcf"
# filters
filters<-list(
```

```
        list("desc"="accept all","predicate"=function(ctx) {TRUE} ),
        list("desc"="accept none","predicate"=function(ctx) {FALSE} ),
        list("desc"="CHROM is '1'","predicate"=function(ctx) { variant.contig(ct
        list("desc"="POS is even","predicate"=function(ctx) { (variant.pos(ctx)%
        list("desc"="PASS filter","predicate"=function(ctx) {!variant.is.filtere
        list("desc"="count(FILTER)>1","predicate"=function(ctx) {length(variant.
        list("desc"="FILTER contains SEGDUP","predicate"=function(ctx) {variant.
        list("desc"="SNP","predicate"=function(ctx) {variant.is.snp(ctx)} ),
        list("desc"="POS!=END","predicate"=function(ctx) { variant.pos(ctx)!=var
        list("desc"="not diallelic","predicate"=function(ctx) {variant.nalleles(
        list("desc"="REF is 'A'","predicate"=function(ctx) {variant.reference(ct
        list("desc"="any allele is 'A'","predicate"=function(ctx) {"A"  %in% var
        list("desc"="any ALT allele is 'A'","predicate"=function(ctx) {"A"  %in%
        list("desc"="No QUAL","predicate"=function(ctx) {!variant.has.qual(ctx)}
        list("desc"="variant has ID","predicate"=function(ctx) {variant.has.id(c
        list("desc"="variant ID match 'rs1*' ","predicate"=function(ctx) {grepl(
        list("desc"="variant has INFO/AF_NFE","predicate"=function(ctx) {variant
        list("desc"="variant has INFO/AF_NFE > 1E-5","predicate"=function(ctx) {
        list("desc"="Missense in PLEKHN1 (VEP)","predicate"=function(ctx) {
                # NO VEP annotation ?
                if(!variant.has.attribute(ctx,"CSQ")) return(FALSE);
                # get VEP annotation
                predictions <- variant.vep(ctx)
                # In SCN5A
                predictions <- predictions[which(predictions$SYMBOL=="PLEKHN1"),
                # Consequence must contain missense
                predictions <- predictions[grep("missense_variant",predictions$C
                nrow(predictions)>0
                })
        )

# count the variant for each filter
for(flt in filters) {
        cat(paste(flt[["desc"]]," ",count.variants(filename,flt[["predicate"]]),
        }
```

Output:

```
> # load rbcf
> library(rbcf)
>
> # create a function counting variants in a VCF
```

```
> count.variants<-function(filename,predicate) {
+       # we don't need the index for this file
+       fp <- bcf.open(filename,FALSE)
+       # number of variants
+       n<-0
+       # loop while we can read a variant
+       while(!is.null(vc<-bcf.next(fp))) {
+               # test the variant
+               if(predicate(vc)) {
+                       # increment the count
+                       n<-n+1
+                       }
+       }
+       # dispose the vcf reader
+       bcf.close(fp)
+       # return the number of variant
+       n
+ }
>
> # A vcf
> filename <- "../tests/data/gnomad.exomes.r2.0.1.sites.vcf"
> # filters
> filters<-list(
+       list("desc"="accept␣all","predicate"=function(ctx) {TRUE} ),
+       list("desc"="accept␣none","predicate"=function(ctx) {FALSE} ),
+       list("desc"="CHROM␣is␣'1'","predicate"=function(ctx) { variant.contig(ct
+       list("desc"="POS␣is␣even","predicate"=function(ctx) { (variant.pos(ctx)%
+       list("desc"="PASS␣filter","predicate"=function(ctx) {!variant.is.filtere
+       list("desc"="count(FILTER)>1","predicate"=function(ctx) {length(variant.
+       list("desc"="FILTER␣contains␣SEGDUP","predicate"=function(ctx) {variant.
+       list("desc"="SNP","predicate"=function(ctx) {variant.is.snp(ctx)} ),
+       list("desc"="POS!=END","predicate"=function(ctx) { variant.pos(ctx)!=var
+       list("desc"="not␣diallelic","predicate"=function(ctx) {variant.nalleles(
+       list("desc"="REF␣is␣'A'","predicate"=function(ctx) {variant.reference(ct
+       list("desc"="any␣allele␣is␣'A'","predicate"=function(ctx) {"A"  %in% var
+       list("desc"="any␣ALT␣allele␣is␣'A'","predicate"=function(ctx) {"A"  %in%
+       list("desc"="No␣QUAL","predicate"=function(ctx) {!variant.has.qual(ctx)}
+       list("desc"="variant␣has␣ID","predicate"=function(ctx) {variant.has.id(c
+       list("desc"="variant␣ID␣match␣'rs1*'␣","predicate"=function(ctx) {grepl(
+       list("desc"="variant␣has␣INFO/AF_NFE","predicate"=function(ctx) {variant
+       list("desc"="variant␣has␣INFO/AF_NFE␣>␣1E-5","predicate"=function(ctx) {
```

```
+       list("desc"="Missense␣in␣PLEKHN1␣(VEP)","predicate"=function(ctx) {
+               # NO VEP annotation ?
+               if(!variant.has.attribute(ctx,"CSQ")) return(FALSE);
+               # get VEP annotation
+               predictions <- variant.vep(ctx)
+               # In SCN5A
+               predictions <- predictions[which(predictions$SYMBOL=="PLEKHN1"),]
+               # Consequence must contain missense
+               predictions <- predictions[grep("missense_variant",predictions$C
+               nrow(predictions)>0
+               })
+       )
>
> # count the variant for each filter
> for(flt in filters) {
+       cat(paste(flt[["desc"]],"␣",count.variants(filename,flt[["predicate"]]),
+       }
accept all   50
accept none   0
CHROM is '1'   50
POS is even   24
PASS filter   48
count(FILTER)>1   2
FILTER contains SEGDUP   1
SNP   47
POS!=END   3
not diallelic   8
REF is 'A'   6
any allele is 'A'   27
any ALT allele is 'A'   21
No QUAL   1
variant has ID   34
variant ID match 'rs1*'    2
variant has INFO/AF_NFE   50
variant has INFO/AF_NFE > 1E-5   14
Missense in PLEKHN1 (VEP)   13
>
>
>
```

2.11 Print a VEP table for a Variant

Code:

```
# load rbcf
library(rbcf)
# A vcf
filename <- "../tests/data/gnomad.exomes.r2.0.1.sites.vcf"
# we don't need the index for this file
fp <- bcf.open(filename,FALSE)
# current variant
vc <- NULL
while(!is.null(vc<-bcf.next(fp))) {
        #find the first variant having an INFO/CSQ attribute
        if(variant.has.attribute(vc,"CSQ")) break;
        }

if(!is.null(vc)) {
        # get the VEP table for the variant
        predictions<-variant.vep(vc)
        }

# dispose the vcf reader
bcf.close(fp)
# show
predictions
```

Output:

```
> # load rbcf
> library(rbcf)
> # A vcf
> filename <- "../tests/data/gnomad.exomes.r2.0.1.sites.vcf"
> # we don't need the index for this file
> fp <- bcf.open(filename,FALSE)
> # current variant
> vc <- NULL
> while(!is.null(vc<-bcf.next(fp))) {
+       #find the first variant having an INFO/CSQ attribute
+       if(variant.has.attribute(vc,"CSQ")) break;
+       }
>
> if(!is.null(vc)) {
```

```
+       # get the VEP table for the variant
+       predictions<-variant.vep(vc)
+       }
>
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # show
> predictions
   Allele             Consequence   IMPACT   SYMBOL            Gene
1       C downstream_gene_variant MODIFIER   KLHL17 ENSG00000187961
2       A downstream_gene_variant MODIFIER   KLHL17 ENSG00000187961
3       C downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
4       A downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
5       C          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
6       A          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
7       C          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
8       A          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
9       C          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
10      A          intron_variant MODIFIER  PLEKHN1 ENSG00000187583
11      C downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
12      A downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
13      C downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
14      A downstream_gene_variant MODIFIER C1orf170 ENSG00000187642
15      C   upstream_gene_variant MODIFIER  PLEKHN1 ENSG00000187583
16      A   upstream_gene_variant MODIFIER  PLEKHN1 ENSG00000187583
17      C   upstream_gene_variant MODIFIER  PLEKHN1 ENSG00000187583
18      A   upstream_gene_variant MODIFIER  PLEKHN1 ENSG00000187583
   Feature_type         Feature         BIOTYPE EXON INTRON
1    Transcript ENST00000338591  protein_coding
2    Transcript ENST00000338591  protein_coding
3    Transcript ENST00000341290  protein_coding
4    Transcript ENST00000341290  protein_coding
5    Transcript ENST00000379407  protein_coding        2/14
6    Transcript ENST00000379407  protein_coding        2/14
7    Transcript ENST00000379409  protein_coding        2/14
8    Transcript ENST00000379409  protein_coding        2/14
9    Transcript ENST00000379410  protein_coding        2/15
10   Transcript ENST00000379410  protein_coding        2/15
11   Transcript ENST00000433179  protein_coding
12   Transcript ENST00000433179  protein_coding
```

```
13   Transcript ENST00000479361 retained_intron
14   Transcript ENST00000479361 retained_intron
15   Transcript ENST00000480267 retained_intron
16   Transcript ENST00000480267 retained_intron
17   Transcript ENST00000491024  protein_coding
18   Transcript ENST00000491024  protein_coding
                          HGVSc HGVSp cDNA_position CDS_position
1
2
3
4
5  ENST00000379407.3:c.184-51G>C
6  ENST00000379407.3:c.184-51G>A
7  ENST00000379409.2:c.184-51G>C
8  ENST00000379409.2:c.184-51G>A
9  ENST00000379410.3:c.184-51G>C
10 ENST00000379410.3:c.184-51G>A
11
12
13
14
15
16
17
18
   Protein_position Amino_acids Codons Existing_variation ALLELE_NUM DISTANCE
1                                              rs540662886          1     4511
2                                              rs540662886          2     4511
3                                              rs540662886          1     4978
4                                              rs540662886          2     4978
5                                              rs540662886          1
6                                              rs540662886          2
7                                              rs540662886          1
8                                              rs540662886          2
9                                              rs540662886          1
10                                             rs540662886          2
11                                             rs540662886          1     4973
12                                             rs540662886          2     4973
13                                             rs540662886          1     4979
14                                             rs540662886          2     4979
15                                             rs540662886          1      649
```

```
16                                          rs540662886         2      649
17                                          rs540662886         1     3286
18                                          rs540662886         2     3286
   STRAND        FLAGS VARIANT_CLASS MINIMISED SYMBOL_SOURCE HGNC_ID CANONICAL
1       1                      SNV                    HGNC   24023       YES
2       1                      SNV                    HGNC   24023       YES
3      -1                      SNV                    HGNC   28208
4      -1                      SNV                    HGNC   28208
5       1                      SNV                    HGNC   25284
6       1                      SNV                    HGNC   25284
7       1                      SNV                    HGNC   25284
8       1                      SNV                    HGNC   25284
9       1                      SNV                    HGNC   25284       YES
10      1                      SNV                    HGNC   25284       YES
11     -1                      SNV                    HGNC   28208       YES
12     -1                      SNV                    HGNC   28208       YES
13     -1                      SNV                    HGNC   28208
14     -1                      SNV                    HGNC   28208
15      1                      SNV                    HGNC   25284
16      1                      SNV                    HGNC   25284
17      1 cds_start_NF         SNV                    HGNC   25284
18      1 cds_start_NF         SNV                    HGNC   25284
   TSL APPRIS        CCDS            ENSP SWISSPROT        TREMBL       UNIPARC
1            CCDS30550.1 ENSP00000343930    Q6TDP4 Q0VGE6&B3KXL7 UPI00001DFBF0
2            CCDS30550.1 ENSP00000343930    Q6TDP4 Q0VGE6&B3KXL7 UPI00001DFBF0
3                        ENSP00000343864                         UPI000022DAF4
4                        ENSP00000343864                         UPI000022DAF4
5            CCDS53256.1 ENSP00000368717    Q494U1        J3KSM5 UPI00005764FF
6            CCDS53256.1 ENSP00000368717    Q494U1        J3KSM5 UPI00005764FF
7                        ENSP00000368719    Q494U1        J3KSM5 UPI0000D61E06
8                        ENSP00000368719    Q494U1        J3KSM5 UPI0000D61E06
9               CCDS4.1 ENSP00000368720    Q494U1        J3KSM5 UPI00001416D8
10              CCDS4.1 ENSP00000368720    Q494U1        J3KSM5 UPI00001416D8
11                       ENSP00000414022    Q5SV97                 UPI0000418FB0
12                       ENSP00000414022    Q5SV97                 UPI0000418FB0
13
14
15
16
17                       ENSP00000462558                J3KSM5 UPI000268AE1F
18                       ENSP00000462558                J3KSM5 UPI000268AE1F
```

	GENE_PHENO	SIFT	PolyPhen	DOMAINS	HGVS_OFFSET	GMAF	AFR_MAF	AMR_MAF
1						C:0.0008	C:0	C:0
2						C:0.0008	C:0	C:0
3						C:0.0008	C:0	C:0
4						C:0.0008	C:0	C:0
5						C:0.0008	C:0	C:0
6						C:0.0008	C:0	C:0
7						C:0.0008	C:0	C:0
8						C:0.0008	C:0	C:0
9						C:0.0008	C:0	C:0
10						C:0.0008	C:0	C:0
11						C:0.0008	C:0	C:0
12						C:0.0008	C:0	C:0
13						C:0.0008	C:0	C:0
14						C:0.0008	C:0	C:0
15						C:0.0008	C:0	C:0
16						C:0.0008	C:0	C:0
17						C:0.0008	C:0	C:0
18						C:0.0008	C:0	C:0

	EAS_MAF	EUR_MAF	SAS_MAF	AA_MAF	EA_MAF	ExAC_MAF	ExAC_Adj_MAF	ExAC_AFR_MAF
1	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
2	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
3	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
4	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
5	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
6	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
7	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
8	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
9	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
10	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
11	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
12	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
13	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
14	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
15	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
16	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
17	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04
18	C:0	C:0.004	C:0	C:0			C:0	C:2.146e-04

	ExAC_AMR_MAF	ExAC_EAS_MAF	ExAC_FIN_MAF	ExAC_NFE_MAF	ExAC_OTH_MAF
1	C:0	C:0.0002281	C:0.002986	C:0	C:1.606e-05
2	C:0	C:0.0002281	C:0.002986	C:0	C:1.606e-05

```
3           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
4           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
5           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
6           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
7           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
8           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
9           C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
10          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
11          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
12          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
13          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
14          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
15          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
16          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
17          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
18          C:0  C:0.0002281   C:0.002986          C:0  C:1.606e-05
   ExAC_SAS_MAF CLIN_SIG SOMATIC PHENO PUBMED MOTIF_NAME MOTIF_POS HIGH_INF_POS
1           C:0
2           C:0
3           C:0
4           C:0
5           C:0
6           C:0
7           C:0
8           C:0
9           C:0
10          C:0
11          C:0
12          C:0
13          C:0
14          C:0
15          C:0
16          C:0
17          C:0
18          C:0
   MOTIF_SCORE_CHANGE LoF LoF_filter LoF_flags LoF_info"
1
2
3
4
5
```

```
6
7
8
9
10
11
12
13
14
15
16
17
18
>
```

2.12 Print a SNPEFF table for a Variant

Code:

```
# load rbcf
library(rbcf)
# A vcf
filename <- "../tests/data/rotavirus_rf.ann.vcf.gz"
# we don't need the index for this file
fp <- bcf.open(filename,FALSE)
# current variant
vc <- NULL
while(!is.null(vc<-bcf.next(fp))) {
        #find the first variant having an INFO/ANN attribute
        if(variant.has.attribute(vc,"ANN")) break;
        }
if(!is.null(vc)) {
        # get SNPEFF table
        predictions<-variant.snpeff(vc)
        }
# dispose the vcf reader
bcf.close(fp)
# show
predictions
```

Output:

```
> # load rbcf
> library(rbcf)
> # A vcf
> filename <- "../tests/data/rotavirus_rf.ann.vcf.gz"
> # we don't need the index for this file
> fp <- bcf.open(filename,FALSE)
> # current variant
> vc <- NULL
> while(!is.null(vc<-bcf.next(fp))) {
+       #find the first variant having an INFO/ANN attribute
+       if(variant.has.attribute(vc,"ANN")) break;
+       }
> if(!is.null(vc)) {
+       # get SNPEFF table
+       predictions<-variant.snpeff(vc)
+       }
> # dispose the vcf reader
> bcf.close(fp)
[1] TRUE
> # show
> predictions
  Allele       Annotation Annotation_Impact    Gene_Name      Gene_ID
1      C missense_variant          MODERATE Gene_18_3284 Gene_18_3284
  Feature_Type Feature_ID Transcript_BioType Rank   HGVS.c      HGVS.p
1   transcript AAA47319.1     protein_coding  1/1 c.952A>C p.Lys318Gln
  cDNA.pos / cDNA.length CDS.pos / CDS.length AA.pos / AA.length Distance
1              952/3267            952/3267           318/1088
  ERRORS / WARNINGS / INFO'"
1
>
```