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"
> #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
MQ0F   MQ0F      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
MQ0F                                                     "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"="FILTER␣contains␣SEGDUP","predicate"=function(ctx) {variant.
        list("desc"="SNP","predicate"=function(ctx) {variant.is.snp(ctx)} ),
        list("desc"="not␣diallelic","predicate"=function(ctx) {variant.nalleles(
        list("desc"="REF␣is␣’A’","predicate"=function(ctx) {variant.reference(ct
        list("desc"="REF␣is␣’A’","predicate"=function(ctx) {"A"  %in% variant.al
        )

# 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"="FILTER␣contains␣SEGDUP","predicate"=function(ctx) {variant.
+       list("desc"="SNP","predicate"=function(ctx) {variant.is.snp(ctx)} ),
+       list("desc"="not␣diallelic","predicate"=function(ctx) {variant.nalleles(
+       list("desc"="REF␣is␣'A'","predicate"=function(ctx) {variant.reference(ct
+       list("desc"="REF␣is␣'A'","predicate"=function(ctx) {"A"  %in% variant.al
+       )
>
> # 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
FILTER contains SEGDUP   1
SNP   47
not diallelic   8
REF is 'A'   6
REF is 'A'   27
>
>
>
```