An introduction to VPA

Qiang Hu

October 6, 2011

Contents

1 Introduction

`VPA` (Variant Pattern Analyzer) is an R package for prioritizing variants with user-specified frequency pattern from multiple study subjects in next generation sequencing study. The package starts from individual files of sequence and variant calls and extract variants with user-specified frequency pattern across the study subjects of interest. Positional level quality criteria including phred-like quality score and sequencing depth can be incorporated into variant extraction. It can be used in studies with matched pair design as well as studies with multiple distinct groups of subjects. Written in open source R environment, it provides the flexibility for users to adopt, extend and customize the functionality for their specific needs.

The input for VPA consists basically of two types of files for each sample subject, containing position-level information of variant calls (required) and sequence calls (optional). The specific format of input files is Variant Call Format (VCF). VCF format has been widely used in the next-generation sequencing studies. The variant and sequence call data in VCF format can be generated using popular SAMtools or GATK. It contains quality information of all sequenced positions, such as sequencing depth and phred-like quality score.

The `VPA` package has been implemented in both sequential mode and parallel mode.

First of all, the package should be installed and loaded.

```
> library(VPA)
```

A test data set is available in the sub-directory 'extdata' of package 'VPA'.

```
> dirpath <- system.file("extdata", package = "VPA")
```

2 Work flow

For a next-generation sequencing study, samples from different sources can often be classified into groups with distinct phenotype. For example, In an application of three-group design, samples can assigned to group with aggressive phenotype, group with benign phenotype, and group with normal phenotype. Variants with certain frequency pattern relating to phenotype status of groups could be of interest for prioritization, such as recurrent in group with aggressive phenotype, not recurrent in group with benign phenotype, and not observed in group with normal phenotype.

It takes three simple steps to extract the variants with user-specified frequency pattern and quality criteria.

2.1 Quality filtering

The first step is to load variant and sequence call data in VCF format and perform position-level quality filtering. The function `LoadFiltering` is designed to conduct this preprocessing procedure. The `index1.txt` file contains the annotation information of each sample. Each row is for one sample. The four columns are separated by tab, including sample name (required), group status (required), variant call file name (required) and sequence call file name (optional). Sample name column lists the sample name. Group status column lists status (e.g., aggressive, benign or normal) of group each sample belongs to. Variant call file name column lists the path of VCF formatted variant call file. The optional sequence call file name column lists the path of compressed VCF sequence call file. *Due to its high volume nature, the optional sequence call data in tab-delimited VCF formats is generally compressed by bgzip program and can be efficiently retrieved through tabix program from open-source SAMtool package (http://samtools.sourceforge.net/tabix.shtml). As a result, tabix should be installed if the optional sequence call checking is performed. The installation path of tabix could be specified in the optional argument of* `LoadFiltering` *function if it is not in the PATH system environment.*

```
> read.table(file.path(dirpath, "index1.txt"), sep = "\t")
```

```
          V1 V2                V3                V4
1 1151HZ0001  A 1151HZ0001.flt.vcf 1151HZ0001.vcf.gz
2 1151HZ0002  A 1151HZ0002.flt.vcf 1151HZ0002.vcf.gz
3 1151HZ0004  A 1151HZ0004.flt.vcf 1151HZ0004.vcf.gz
4 1151HZ0005  A 1151HZ0005.flt.vcf 1151HZ0005.vcf.gz
5 1151HZ0006  B 1151HZ0006.flt.vcf 1151HZ0006.vcf.gz
6 1151HZ0007  B 1151HZ0007.flt.vcf 1151HZ0007.vcf.gz
```

```
> varflt <- LoadFiltering(file.path(dirpath, "index1.txt"), datadir = dirpath,
+     filtering = TRUE, alter.PL = 20, alter.AD = 3, alter.ADP = NULL,
+     QUAL = 20, DP = c(10, 500), GQ = 20, tabix = "tabix", parallel = FALSE)

Load variants for 1151HZ0001:
Read data...
Format to vcf object...
Load variants for 1151HZ0002:
Read data...
Format to vcf object...
Load variants for 1151HZ0004:
Read data...
Format to vcf object...
Load variants for 1151HZ0005:
Read data...
Format to vcf object...
Load variants for 1151HZ0006:
Read data...
Format to vcf object...
Load variants for 1151HZ0007:
Read data...
Format to vcf object...
Extract sequence level data ...
retrieve calls from 1151HZ0001 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0002 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0004 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0005 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0006 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0007 with tabix...
Format to vcf object...
```

The arguments "filtering" is used to decide whether position-level quality filtering should be conducted. For each genomic position of variant call, VPA will retrieve variant call information from the subject(s) with variant called at this position. If the optional sequence call files are specified in the `index1.txt` file, it will take advantage of installed tabix function to retrieve sequence call information overlapping variant call position from the subject(s) without variant called at the same position. It will filter out the variant positions which don't

reach the user-specified quality criteria from further analysis. The quality criteria, including sequencing depth and phred-like quality score, can be set in the optional argument of `LoadFiltering` function.

By default, the phred-like quality score is set at $\geq$ 20 and the sequencing depth is set at $\geq$ 10x and $\leq$ 500x. PL (Phred-scaled genotype likelihoods) is a score for possible genotype. The alter.PL=20 is used to find possible variants as control to reduce false positive results. The arguments 'alter.AD' and 'alter.AP' define the threshold of altered allele for the variants. In this example, 'alter.AD=3' refers to the variants should at least have 3 altered alleles. More details about the filtering criteria can be find at 'help(filtervcf)'.

```
> varflt
```

```
$vcflist
$vcflist$`1151HZ0001`
VCF data
Calls: 49 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$vcflist$`1151HZ0002`
VCF data
Calls: 45 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$vcflist$`1151HZ0004`
VCF data
Calls: 41 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$vcflist$`1151HZ0005`
VCF data
Calls: 38 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$vcflist$`1151HZ0006`
VCF data
Calls: 42 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$vcflist$`1151HZ0007`
VCF data
Calls: 46 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...


$VarIndex
         1151HZ0001 1151HZ0002 1151HZ0004 1151HZ0005 1151HZ0006 1151HZ0007
1:567825 NA         NA         NA         NA         NA         NA
1:800383 NA         NA         NA         NA         NA         NA
1:808631 "TRUE"     "FALSE"    "FALSE"    "FALSE"    "FALSE"    "TRUE"
```

```
... ...
$Samples
     V1           V2  V3                    V4
[1,] "1151HZ0001" "A" "1151HZ0001.flt.vcf" "1151HZ0001.vcf.gz"
[2,] "1151HZ0002" "A" "1151HZ0002.flt.vcf" "1151HZ0002.vcf.gz"
[3,] "1151HZ0004" "A" "1151HZ0004.flt.vcf" "1151HZ0004.vcf.gz"

... ...
```

2.2 Specifying Pattern

The pattern of variant frequency can be customized by user according to the experiment design. The minimum and maximum frequency value in each group are sufficient to make up of a pattern matrix. In the test data set, there are 4 case samples and 2 control samples respectively. It is an unpaired design. To prioritize variants that occur in at least one case sample but do not occur in any of the control samples, the minimum and maximum frequency in case group will be set as 1/4 and 1, respectively. The minimum and maximum frequency in control group will be both set as 0.

```
> pattern <- cbind(A = c(1/4, 1), B = c(0, 0))
> pattern

        A B
[1,] 0.25 0
[2,] 1.00 0
```

The column names of the resulted frequency matrix correspond to the group names in the `index1.txt` file. The first row of the matrix lists the minimum frequency values of each group and the second row of the matrix lists the maximum frequency values of each group.

2.3 Extracting variants

Once an object for the list of qualified genomic position of variant call is obtained and the frequency pattern is defined, the last step is to extract variants with desired pattern. The `Patterning` function is designed for this purpose, with the the object of qualified variants from the first step and the object of customized pattern from the second step as input for the function. The argument 'not.covered=NULL' will remove the positions with low depth reads. The 'var.PL=c(FALSE, TRUE)' is used to label possible variants as true variants in group B rather than group A when filtering positions with specified pattern. This two arguments can be used to reduce false positive results.

```
> varpat1 <- Patterning(varflt, pattern, paired = FALSE, not.covered = NULL,
+     var.PL = c(FALSE, TRUE))
> varpat1

$VarVCF
$VarVCF$`1151HZ0001`
VCF data
```

```
Calls: 12 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0002`
VCF data
Calls: 11 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0004`
VCF data
Calls: 8 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0005`
VCF data
Calls: 7 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0006`
VCF data
Calls: 0 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0007`
VCF data
Calls: 0 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...


$VarFrequency
            A   B
1:808631 0.25 0.5
1:808922 0.75 1.0
1:808928 0.25 0.5

... ...
$Pattern
        A B
[1,] 0.25 0
[2,] 1.00 0

... ...
$Samples
     V1           V2  V3                    V4
[1,] "1151HZ0001" "A" "1151HZ0001.flt.vcf" "1151HZ0001.vcf.gz"
[2,] "1151HZ0002" "A" "1151HZ0002.flt.vcf" "1151HZ0002.vcf.gz"
[3,] "1151HZ0004" "A" "1151HZ0004.flt.vcf" "1151HZ0004.vcf.gz"

... ...
```

The result is a `varlist` object, including `VarVCF`, `VarFrequency`, `Pattern` and `Samples`. `VarVCF` is a list of VCF format variant data fitting the specified frequency pattern and quality criteria. `VarFrequency` is a matrix recording every variant's frequency in each group.

3 Parallel mode

The first step of the `VPA` package, *i.e.*, performing thorough position-level quality filtering using both variant and sequence call files, is the most time-consuming part. By default, the function `LoadFiltering` conduct quality filtering in a sequential mode. To speed up this step, We also implement the function to run the filtering in a parallel model. To run the `LoadFiltering` function in a parallel mode, one can set argument 'parallel=TRUE'.

```
> varflt <- LoadFiltering(file.path(dirpath, "index1.txt"), datadir = dirpath,
+     filtering = TRUE, alter.PL = 20, alter.AD = 3, alter.ADP = NULL,
+     QUAL = 20, DP = c(10, 500), GQ = 20, tabix = "tabix", parallel = TRUE,
+     pn = 2)

Load variants for 1151HZ0001:
Read data...
Format to vcf object...
Load variants for 1151HZ0002:
Read data...
Format to vcf object...
Load variants for 1151HZ0004:
Read data...
Format to vcf object...
Load variants for 1151HZ0005:
Read data...
Format to vcf object...
Load variants for 1151HZ0006:
Read data...
Format to vcf object...
Load variants for 1151HZ0007:
Read data...
Format to vcf object...
Extract sequence level data ...
R Version:  R version 2.11.1 (2010-05-31)

Library VPA loaded.
```

The argument 'pn=2' set the number of CPUs to be used. The function in parallel mode is implemented by the open source R package 'snowfall'. More arguments and details to run the function in parallel can be found in the manual of function `sfInit`.

Note that `VPA` package also allows user to perform fast quality filtering using only the variant call files. This can be performed by supplying the variant call files only in the `index1.txt`.

4 Case study

4.1 Unpaired multi-groups design

In an application of three-group design (e.g., group with aggressive phenotype, group with benign phenotype, group with normal phenotype) with 2 subjects per group, one might want to extract variants recurrent in aggressive phenotype group (i.e., frequency ≥ 0.5), not recurrent in benign phenotype group (i.e., frequency ≤ 0.5), and not observed in normal phenotype group at all (i.e., frequency $=0.0$). The `index2.txt` file shows such a three-group design.

```
> read.table(file.path(dirpath, "index2.txt"), sep = "\t")
```

```
          V1 V2                 V3                V4
1 1151HZ0001  A 1151HZ0001.flt.vcf 1151HZ0001.vcf.gz
2 1151HZ0002  A 1151HZ0002.flt.vcf 1151HZ0002.vcf.gz
3 1151HZ0004  B 1151HZ0004.flt.vcf 1151HZ0004.vcf.gz
4 1151HZ0005  B 1151HZ0005.flt.vcf 1151HZ0005.vcf.gz
5 1151HZ0006  C 1151HZ0006.flt.vcf 1151HZ0006.vcf.gz
6 1151HZ0007  C 1151HZ0007.flt.vcf 1151HZ0007.vcf.gz
```

The first step is to load the VCF format data and perform position-level quality filtering. By default, the phred-like quality score is set at ≥ 20 and the sequencing depth is set at ≥ 10x. User can set the quality criteria in the optional argument of `LoadFiltering` function.

```
> varflt <- LoadFiltering(file = file.path(dirpath, "index2.txt"),
+     datadir = dirpath, filtering = TRUE, alter.PL = 20, alter.AD = 3,
+     QUAL = 20, DP = c(10, 500), GQ = 20, tabix = "tabix")
```

```
Load variants for 1151HZ0001:
Read data...
Format to vcf object...
Load variants for 1151HZ0002:
Read data...
Format to vcf object...
Load variants for 1151HZ0004:
Read data...
Format to vcf object...
Load variants for 1151HZ0005:
Read data...
Format to vcf object...
Load variants for 1151HZ0006:
Read data...
Format to vcf object...
Load variants for 1151HZ0007:
Read data...
Format to vcf object...
Extract sequence level data ...
retrieve calls from 1151HZ0001 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0002 with tabix...
```

```
Format to vcf object...
retrieve calls from 1151HZ0004 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0005 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0006 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0007 with tabix...
Format to vcf object...
```

For aggressive phenotype group, the minimum and maximum frequency will be set as 0.5 and 1, respectively. For benign phenotype group, the minimum and maximum frequency will be set as 0 and 0.5, respectively. For normal phenotype group, both the minimum and maximum frequency will be set as 0.

```
> pattern <- cbind(A = c(0.5, 1), B = c(0, 0.5), C = c(0, 0))
```

The variants with with such a frequency pattern can be extracted using `Patterning` function.

```
> varpat2 <- Patterning(varflt, pattern, not.covered = NULL, var.PL = c(FALSE,
+     FALSE, TRUE))
> varpat2

$VarVCF
$VarVCF$`1151HZ0001`
VCF data
Calls: 12 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0002`
VCF data
Calls: 11 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0004`
VCF data
Calls: 5 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0005`
VCF data
Calls: 4 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0006`
VCF data
Calls: 0 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0007`
```

```
VCF data
Calls: 0 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...


$VarFrequency
          A   B   C
1:808631 0.5 0.0 0.5
1:808922 1.0 0.5 1.0
1:808928 0.5 0.0 0.5

... ...
$Pattern
       A   B C
[1,] 0.5 0.0 0
[2,] 1.0 0.5 0

... ...
$Samples
     V1           V2  V3                    V4
[1,] "1151HZ0001" "A" "1151HZ0001.flt.vcf" "1151HZ0001.vcf.gz"
[2,] "1151HZ0002" "A" "1151HZ0002.flt.vcf" "1151HZ0002.vcf.gz"
[3,] "1151HZ0004" "B" "1151HZ0004.flt.vcf" "1151HZ0004.vcf.gz"

... ...
```

4.2 Paired design

Paired design sequencing experiment can be used to identify somatic type of variant (i.e., variants occurring in cancer case subject but not in matched cancer-free subject). Recurrent somatic variants can be of great interests for follow up analysis. In a paired design sequencing study, `VPA` package can be used to identify somatic type of variants with the user customized frequency pattern. The "index3.txt" file shows a example of paired design with two cases and two matched controls. The identity of case to its matched control can be determined by the sample name column.

```
> read.table(file.path(dirpath, "index3.txt"), sep = "\t")

          V1      V2                 V3                V4
1 1151HZ0001    case 1151HZ0001.flt.vcf 1151HZ0001.vcf.gz
2 1151HZ0004    case 1151HZ0004.flt.vcf 1151HZ0004.vcf.gz
3 1151HZ0001 control 1151HZ0006.flt.vcf 1151HZ0006.vcf.gz
4 1151HZ0004 control 1151HZ0007.flt.vcf 1151HZ0007.vcf.gz
```

The same three simple steps as described above can be used to extract the variants of interest.

```
> varflt <- LoadFiltering(file = file.path(dirpath, "index3.txt"),
+     datadir = dirpath, filtering = TRUE, alter.PL = 20, alter.AD = 3,
+     QUAL = 20, DP = c(10, 500), GQ = 20, tabix = "tabix")
```

```
Load variants for 1151HZ0001:
Read data...
Format to vcf object...
Load variants for 1151HZ0004:
Read data...
Format to vcf object...
Load variants for 1151HZ0001:
Read data...
Format to vcf object...
Load variants for 1151HZ0004:
Read data...
Format to vcf object...
Extract sequence level data ...
retrieve calls from 1151HZ0001 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0004 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0001 with tabix...
Format to vcf object...
retrieve calls from 1151HZ0004 with tabix...
Format to vcf object...
```

```
> pattern <- cbind(case = c(0.5, 1), control = c(0, 0))
> varpat3 <- Patterning(varflt, pattern, paired = TRUE, not.covered = NULL,
+     var.PL = c(FALSE, TRUE))
> varpat3
```

```
$VarVCF
$VarVCF$`1151HZ0001`
VCF data
Calls: 19 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$VarVCF$`1151HZ0004`
VCF data
Calls: 14 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...


$VarFrequency
         case
1:808631  0.5
1:808922  0.0
1:808928  0.5

... ...
$Pattern
     case
[1,]  0.5
[2,]  1.0
```

```
... ...
$Samples
     V1           V2        V3                   V4
[1,] "1151HZ0001" "case"    "1151HZ0001.flt.vcf" "1151HZ0001.vcf.gz"
[2,] "1151HZ0004" "case"    "1151HZ0004.flt.vcf" "1151HZ0004.vcf.gz"
[3,] "1151HZ0001" "control" "1151HZ0006.flt.vcf" "1151HZ0006.vcf.gz"

... ...
```

5 Summary results

5.1 Variant frequency

Several functions are also provided to output the variant analysis results. For example, the function `vcfreq` can be used to summarize the variant frequency of a `varlist` object. We take the result data "varpat1" as an example:

```
> vfreq <- vcfreq(varpat1, method = "fisher.test")
> head(vfreq)
```

```
         REF 1151HZ0001 1151HZ0002 1151HZ0004 1151HZ0005 1151HZ0006 1151HZ0007
1:949053 "C" "C/G"      "C/G"      "."        "C/G"      "."        "."
1:970836 "G" "G/A"      "."        "G/A"      "."        "."        "."
1:985239 "C" "C/T"      "."        "C/T"      "."        "."        "."
1:985900 "C" "C/T"      "."        "C/T"      "."        "."        "."
1:985999 "G" "G/T"      "."        "G/T"      "."        "."        "."
1:809700 "G" "."        "G/A"      "G/A"      "."        "."        "."
         A       B   p.value
1:949053 "0.375" "0" "0.490909090909091"
1:970836 "0.25"  "0" "0.515151515151515"
1:985239 "0.25"  "0" "0.515151515151515"
1:985900 "0.25"  "0" "0.515151515151515"
1:985999 "0.25"  "0" "0.515151515151515"
1:809700 "0.25"  "0" "0.515151515151515"
```

The resulted matrix lists reference, genotypes, altered allele frequencies and p-value across the study groups of all positions. The positions without variants are marked with ".". The method 'fisher.test' or 'chisq.test' can be used to test the frequency difference of altered alleles between groups.

5.2 Gene frequency

Variants can be mapped to gene level to find the genes with different frequencies between groups. The function `Pos2Gene` can be used to annotate a position in reference genome to 'gene' level or 'exon' level. The annotation data come from UCSC reference genome table "refFlat". For example,

```
> Pos2Gene("1", "949053", level = "exon", ref = "hg19")
```

```
[1] "intron" "ISG15"
```

This position is in the intron region of gene "ISG15". The altered frequency in gene level can be summerized by the function `gefreq`. We take the same data "varpat1" for example. The variants of each sample will be annotated and the gene frequencies will be calculated.

```
> gfreq <- gefreq(varpat1, level = "exon", ref = "hg19", method = "fisher.test")
> gfreq$frequency
```

```
                    1151HZ0001 1151HZ0002 1151HZ0004 1151HZ0005 1151HZ0006
exon:AGRN           "3"        "1"        "1"        "0"        "0"
exon:FAM41C         "0"        "1"        "1"        "0"        "0"
exon:KLHL17         "0"        "1"        "0"        "0"        "0"
exon:NOC2L          "0"        "1"        "0"        "0"        "0"
exon:SAMD11         "2"        "1"        "0"        "1"        "0"
exon:TTLL10         "0"        "0"        "2"        "0"        "0"
intergene:HES4;ISG15 "0"       "0"        "0"        "1"        "0"
intron:AGRN         "4"        "0"        "3"        "1"        "0"
intron:FAM41C       "0"        "1"        "0"        "0"        "0"
intron:ISG15        "1"        "1"        "0"        "1"        "0"
intron:KLHL17       "0"        "1"        "0"        "1"        "0"
intron:NOC2L        "0"        "3"        "0"        "2"        "0"
intron:PLEKHN1      "2"        "0"        "1"        "0"        "0"
                    1151HZ0007 A      B   p.value
exon:AGRN           "0"        "0.75" "0" "0.4"
exon:FAM41C         "0"        "0.5"  "0" "0.466666666666667"
exon:KLHL17         "0"        "0.25" "0" "1"
exon:NOC2L          "0"        "0.25" "0" "1"
exon:SAMD11         "0"        "0.75" "0" "0.4"
exon:TTLL10         "0"        "0.25" "0" "1"
intergene:HES4;ISG15 "0"       "0.25" "0" "1"
intron:AGRN         "0"        "0.75" "0" "0.4"
intron:FAM41C       "0"        "0.25" "0" "1"
intron:ISG15        "0"        "0.75" "0" "0.4"
intron:KLHL17       "0"        "0.5"  "0" "0.466666666666667"
intron:NOC2L        "0"        "0.5"  "0" "0.466666666666667"
intron:PLEKHN1      "0"        "0.5"  "0" "0.466666666666667"
```

```
> head(gfreq$annotation[[1]])
```

```
     position   annotation genename
[1,] "1:874707" "exon"     "SAMD11"
[2,] "1:879431" "exon"     "SAMD11"
[3,] "1:906016" "intron"   "PLEKHN1"
[4,] "1:908749" "intron"   "PLEKHN1"
[5,] "1:949053" "intron"   "ISG15"
[6,] "1:970836" "intron"   "AGRN"
```

5.3 Variant information

For position(s) of interest, the function `subvcf` can be used to retrieve the detailed variant information. For example, the above position "1:949053" and "1:970836" contain variant call in sample "1151HZ0001".

```
> vcf1 <- subvcf(varpat1$VarVCF[[1]], CHRPOS = c("1:949053", "1:970836"))
> write.vcf(vcf1, file = "")

     CHROM POS      ID  REF ALT QUAL  FILTER
[1,] "1"   "949053" "." "C" "G" "54"  "."
[2,] "1"   "970836" "." "G" "A" "121" "."
     INFO
[1,] "DP=40;DP4=0,19,1,16;MQ=57;FQ=57;AF1=0.5;CI95=0.5,0.5;PV4=0.47,3.2e-13,1,1"
[2,] "DP=33;DP4=1,15,0,15;MQ=60;FQ=124;AF1=0.5;CI95=0.5,0.5;PV4=1,0.031,1,1"
     FORMAT     1151HZ0001
[1,] "GT:PL:GQ" "0/1:84,0,167:87"
[2,] "GT:PL:GQ" "0/1:151,0,198:99"
```

The detailed sequence call information can also be retrieved from samples without variant called at the same position.

```
> vcf2 <- pos2seq(Pos = rbind(c("1", "949053"), c("1", "970836")),
+     Seqfile = file.path(dirpath, varpat1$Samples[5, 4]))
> vcf2$vcf

  V1  V2       V3  V4  V5  V6  V7
1 "1" "949053" "." "C" "X" "0" "."
2 "1" "970836" "." "G" "X" "0" "."
  V8                                                                  V9
1 "DP=27;I16=0,27,0,0,896,30624,0,0,1589,94441,0,0,288,4576,0,0" "PL"
2 "DP=22;I16=1,21,0,0,807,30109,0,0,1320,79200,0,0,299,5013,0,0" "PL"
  V10
1 "0,81,199"
2 "0,66,245"
```

5.4 Output

The function `write.vcf` can be used to export the identified variant with user-specified frequency pattern and quality criteria as a plain text file in VCF format. It can be used as input file for further analysis.

```
> write.vcf(varpat1$VarVCF[[1]], file = tempfile())
```

6 Database filtering

In order to find novel variants, the results can also be filtered by some public nucleotide variant databases, such as dbSNP and 1000 genome. The function `filterpos` can be used to filter variants with different type of data sets, including 'vcf', 'bed', 'gff' and user-specified position matrix. Here are examples to show how to filter variants with dbSNP and 1000 Genome.

6.1 dbSNP

The latest version of dbSNP database (dbSNP132) is available on UCSC genome. It can be downloaded at `http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/snp132.txt.gz`. The 2-5 columns of the data set can be extracted

to build a bed-format file. The bed file is still too big to retrieve annotation. Thus 'tabix' is used to index this file for extracting information rapiddly. Here is an example for data preparing.

```
$wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/snp132.txt.gz
$gzip -d -c snp132.txt.gz | cut -f 2-5 > snp132.bed
$bgzip snp132.bed
$tabix -p bed snp132.bed.gz
```

In this section, we only use part of the dbSNP132 dataset as an example because the data set 'snp132.bed.gz' is too large to demonstrate.

```
> vcf3 <- varpat1$VarVCF[[1]]
> vcf3F1 <- filterpos(vcf3, file = file.path(dirpath, "dbSNP.bed.gz"),
+     type = "bed", tbi = TRUE, chr = TRUE, tabix = "tabix")
> vcf3F1

$filtered
VCF data
Calls: 7 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$dropped
VCF data
Calls: 5 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...
```

The arguments 'type='bed" and 'tbi=TRUE' are used to define the format of input file, which is in bed format and indexed by 'tabix'. The chromosome names in dbSNP have the prefix of "chr", so the argument 'chr' is set as 'TRUE'. There are 5 variants filtered out because they are annotated in dbSNP database, and 7 variants are left.

6.2 1000 Genome

The variants data released by 1000 Genome project are in indexed VCF format. The lastest version can be downloaded at `ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20101123/interim_phase1_release/`. The dat sets can be used directly as input file to filter variants. We also only use part of the dataset for example to filter the variants.

```
> vcf3F2 <- filterpos(vcf3F1$filtered, file = file.path(dirpath,
+     "1KG.vcf.gz"), type = "vcf", tbi = TRUE, chr = FALSE, tabix = "tabix")
> vcf3F2

$filtered
VCF data
Calls: 2 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...

$dropped
VCF data
Calls: 5 postions, 1 sample(s)
Names: HEAD CHROM POS ID REF ALT...
```

There are 2 variants left and 5 variants are filtered out by 1000 Genome dataset. The left 2 variants meet our specified sequencing quality criteria and frequency pattern. Furthermore, they are novel variants without annotation in current databases.

7 Session information

```
> sessionInfo()

R version 2.11.1 (2010-05-31)
x86_64-unknown-linux-gnu

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] VPA_0.3.1     snowfall_1.84 snow_0.3-3
```