# RAREsim Vignette

Megan Null

This vigette describes how to use the RAREsim R package to simulate rare variant genetic data. A start to finish simulation with RAREsim can be found online at the RAREsim Example Code github page. Here, R functions within the package are described in more detail.

The example below simulates a one cM block on chromosome 19. Here, RAREsim simulates haplotypes to match target data from the African ancestry group from gnomAD v2.1 (Karczewski, et al., 2020).

## Install the package

```r
library(RAREsim)
```

The source code for all functions within the RAREsim package can be found at https://github.com/meganmichelle/RAREsim.

RAREsim has three main steps:

(1) Simulate genetic data with an abundance of rare variants using HAPGEN2 (Su, 2011). By simulating with default parameters and input haplotypes with information at all sequencing bases, including monomorphic sites, HAPGEN2 simulates an abundance of rare variants.

(2) Estimate the expected number of variants in minor allele count (MAC) bins. Users may fit target data, manually enter parameters, or use default parameters to estimate the number of variants per MAC bin; all are described in detail below.

(3) Probabilistically prune the rare variants to match the estimated number of variants in each MAC bin. RAREsim prunes the simulated variants by returning all or a subset of alternate alleles back to reference.

## Simulate genetic data with an abundance of rare variants

An example simulation with HAPGEN2 can be found on the RAREsim Example Code github page. The simulation is done using HAPGEN2 default parameters and haplotype files with information at every sequencing base within the region of interest.

After haplotypes are simulated with HAPGEN2, all the bases that are monomorphic in the simulated data will be removed in the haplotype and legend files (see the example code).

A MAC file will be created - a file with a single column counting the number of alternate alleles for that variant. The MAC file will be compared with the expected number of variants per MAC bin in the pruning step.

## Estimate the number of variants per MAC bin

The number of variants is estimated by combining the *Number of Variants* function and the *Allele Frequency Spectrum (AFS)* function.

### The *Number of Variants* function

For a given region, the *Number of Variants* function estimates the number of variants per Kb, $f_{nvariant}(n)$, for a sample size $n$. Estimating the number of variants can be achieved by 1) fitting target data to estimate parameters, 2) using default parameters, or 3) directly inputting parameters to the function. Additionally, a user may directly input the number of variants expected in the region (e.g. 1000 variants).

#### 1) Fitting Target Data

Target data is used to estimate $\phi$ and $\omega$ to optimize the function $f_{nvariant}(n) = \phi n^{\omega}$ to fit the target data.

The Number of Variants target data consists of various sample sizes ($n$) and the observed number of variants per Kb in the region of interest. Ancestry specific data is advised. Data should be formatted with the first column as the number of individuals ($n$) and the second column as the observed number of variants per Kb in the region of interest ($per_{kb}$).

Here we will fit the example target data for the African ancestry population.

```
# load the target data
data("nvariant_afr")
print(nvariant_afr, row.names  =  FALSE)
```

```
##     n      per_kb
##    10  0.2627568
##    20  0.6831678
##    50  1.5239897
##   100  2.7326712
##   200  4.3092123
##   500  7.6199485
##  1000 12.1919176
##  2000 19.3914551
##  3070 25.2246571
##  5000 33.4226707
##  5040 33.7905302
##  8128 45.1941773
```

The target data is used to estimate $\phi$ and $\omega$ within a least squares loss function, optimizing using sequential quadratic programming (SQP). This optimization is implemented via the *fit_nvariant* function.

```
nvar <- fit_nvariant(nvariant_afr)
nvar
```

```
## $phi
## [1] 0.1638108
##
## $omega
## [1] 0.6248848
```

The output of the *fit_nvariant* function are the parameters phi ($\phi$) and omega ($\omega$), respectively. The estimated parameters can then be used to determine the expected number of variants per Kb, given the number of individuals to be simulated, $N_{sim}$.

To simulate the sample size observed in the target data used here, ($N_{sim} = 8128$), we calculate $\hat{f_{nvariant}}(N_{sim}) = \hat{\phi} N_{sim}^{\hat{\omega}}$. This can be done with the *nvariant* function.

Parameter values estimated from the target data are used here, as well as the sample size.

```
nvariant(phi = nvar$phi, omega = nvar$omega, N = 8128)
```

```
## [1] 45.46027
```

### 2) Using Default Parameters

RAREsim also provides ancestry specific default parameters for phi ($\phi$), omega ($\omega$). To use the default parameters, the ancestry must be specified: African (AFR), East Asian (EAS), Non-Finnish European (NFE), or South Asian (SAS).

```
nvariant(N=8128, pop = 'AFR')
```

```
## [1] 43.66395
```

### 3) Directly Inputting Parameters

Finally, parameters can be directly input into the *Number of Variants* function.

```
nvariant(phi = 0.1638108, omega = 0.6248848, N = 8128)
```

```
## [1] 45.46026
```

### Total Number of Variants in the Region

The example data here is a cM block with 19,029 bp. Thus, to calculate the total expected number of variants in the region, we multiple the expected number of variants per Kb by 19.029.

```
19.029*nvariant(nvar$phi, omega = nvar$omega, N = 8128)
```

```
## [1] 865.0634
```

### The *Allele Frequency Spectrum (AFS)* Function

The *AFS* function inputs a MAC ($z$) and outputs the proportion of variants at MAC = z, ($f_{afs}(z)$). This is done by estimating $\alpha$ and $\beta$ to optimize the function $f_{afs}(z) = \frac{b}{(\beta+z)^{\alpha}}$. Here $b$ ensures that the sum of the individual rare allele count proportions equals the total proportion of rare variants, $p_{rv}$.

The *AFS* function inputs a data frame with the upper and lower boundaries for each bin and proportion of variants within each respective bin. The default bins used here and within the evaluation of RAREsim are:

MAC = 1
MAC = 2
MAC = 3 - 5
MAC = 6 - 10
MAC = 11 - 20
MAC = 21 - MAF = 0.5%
MAF = 0.5% - MAF = 1%

These are the recommended bins when simulated sample sizes above 3,500.

When simulating sample sizes between 2000 and 3500, the recommended MAC bins are:

MAC = 1
MAC = 2
MAC = 3 - 5
MAC = 6 - MAF = 0.25%
MAF = 0.25% - MAF = 0.5%
MAF = 0.5% - MAF = 1%

If a sample size below 2000 is desired, we recommend simulating 2000 individuals and taking a random sample to reach the desired sample size.

Estimating the AFS can be achieved by 1) fitting target data to estimate parameters, 2) using default parameters, or 3) directly inputting parameters to the function. Additionally, a user may directly input the proportion of variants in each MAC bin.

### 1) Fitting Target Data

Here we will fit the example target data for the African ancestry population.

The first two columns in the target data identify the lower and upper boundaries of each MAC bin. The third column specifies the observed proportion of variants within each MAC bin in the target data.

```
# load the data
data("afs_afr")
print(afs_afr)
```

```
##   Lower Upper       Prop
## 1     1     1 0.50257998
## 2     2     2 0.16305470
## 3     3     5 0.08255934
## 4     6    10 0.05882353
## 5    11    20 0.03715170
## 6    21    81 0.05675955
## 7    82   162 0.01754386
```

The *fit_afs* function estimates the parameters alpha ($\alpha$), beta ($\beta$), and $b$.

```
af <- fit_afs(Observed_bin_props = afs_afr)
print(af)
```

```
## $alpha
## [1] 1.594622
##
## $beta
## [1] -0.2846474
##
## $b
## [1] 0.297495
##
## $Fitted_results
##   Lower Upper       Prop
## 1     1     1 0.50753380
## 2     2     2 0.12582725
## 3     3     5 0.12226962
## 4     6    10 0.06152310
## 5    11    20 0.04187244
## 6    21    81 0.04709594
## 7    82   162 0.01235050
```

### 2) Using Default Parameters

As with the *Number of Variants* function, default parameters can be used to estimate the parameters for the *AFS* function with the *afs* function. As the default parameters are ancestry specific, the ancestry needs to be specified as pop = 'AFR', 'EAS', 'NFE', or 'SAS'. The function requires a MAC bin dataframe, with the bins specified.

This is the first two columns of the AFS target data.

```
mac <- afs_afr[,c(1:2)]
```

Using the MAC bins as input and specifying an African ancestry, the default parameters are used below to estimate the proportion of variants within each bin.

```r
afs(mac_bins = mac, pop = 'AFR')
```

```
##   Lower Upper       Prop
## 1     1     1 0.51575451
## 2     2     2 0.12460586
## 3     3     5 0.12032463
## 4     6    10 0.06046027
## 5    11    20 0.04121670
## 6    21    81 0.04656603
## 7    82   162 0.01228838
```

#### 3) Directly Inputting Parameters

The *AFS* function can also directly input the parameters alpha, beta, and b.

```r
afs(alpha = 1.594622, beta =  -0.2846474, b  = 0.297495, mac_bins = mac)
```

```
##   Lower Upper       Prop
## 1     1     1 0.50753386
## 2     2     2 0.12582721
## 3     3     5 0.12226954
## 4     6    10 0.06152304
## 5    11    20 0.04187238
## 6    21    81 0.04709586
## 7    82   162 0.01235047
```

### Expected Number of Variants per MAC bin

Once the total number of variants (*nvariant*) and proportion of variants per bin (*afs*) have been estimated, the expected number of variants per MAC bin can be estimated. An example using the total number of varants and estimated proportion of variants per MAC bin is shown below.

The MAC bin boundaries should be defined based on the sample size that will be simulated.

Continuing our example, we will imput the total number of variants we expect in the region (Total_num_var = 865.0634) and the estimated proportion of variants per MAC bin (af$Fitted_results).

```r
bin_estimates <- expected_variants(Total_num_var = 865.0634, mac_bin_prop = af$Fitted_results)
print(bin_estimates)
```

```
##   Lower Upper Expected_var
## 1     1     1    439.04891
## 2     2     2    108.84855
## 3     3     5    105.77097
## 4     6    10     53.22138
## 5    11    20     36.22231
## 6    21    81     40.74098
## 7    82   162     10.68397
```

The output of the *expected_variants* function is the exected number of variants in each MAC bin within the simulation region. This output (shown above) is input for the pruning function.

The *Number of Variants* and *AFS* function can also be calculated within the *expected_variants* function.

```r
bin_estimates <- expected_variants(Total_num_var = 19.029*nvariant(phi = 0.1638108, omega = 0.6248848, 
print(bin_estimates)
```

```
##   Lower Upper Expected_var
## 1     1     1    446.16029
```

```
## 2      2      2    107.79195
## 3      3      5    104.08842
## 4      6     10     52.30196
## 5     11     20     35.65505
## 6     21     81     40.28256
## 7     82    162     10.63023
```

## Pruning Variants

To prune, the expected number of variants are compared to a MAC file. Below is an example MAC file created from the haplotypes simulated for the African ancestry group and the region of interest. Each row represents one variant in the haplotype file.

```r
data("MAC_afr")
```

Pruning happens in two stages: 1) RAREsim theoretically decides which variants should be pruned and 2) the haplotype and legend files are edited. The theoretical pruning takes place in R, within the RAREsim R package. The implementation of the pruning is done via a bash script - see the RAREsim Example Code github page.

### 1) Theoretically Prune

Pruning variants requires a MAC file from the simulated data and the expected number of variants within each MAC bin (product of the *expected_variants* function).

```r
ToPrune <- prune_variants(MAC = MAC_afr, expected = bin_estimates)
head(ToPrune$ToRemove, row.names = FALSE)
```

```
##      line Current_mac New_mac
## 382   382         137       0
## 616   616          99       0
## 1736 1736         105       0
## 2018 2018         114       0
## 2060 2060         117       0
## 3856 3856          93       0
```

```r
head(ToPrune$ToChange)
```

```
## NULL
```

The output from *prune_variants* includes the variants to prune, notated by what line they are in the haplotype file, the current allele count, and the new minor allele count. Variants that will either have all (ToRemove) or a subset of minor alleles (ToChange) removed.

### 2) Pruning Implementation

See the RAREsim Example Code github page for a bash script to prune the variants, given the output from the *prune_variants* function.

### Simulated Genetic Data Complete!

RAREsim simulated genetic data is complete once the pruning has been implemented. The resulting files are haplotype and legend files - the same format as the input reference datasets.