FastEPRR: a Fast Estimator of Population Recombination Rate

Feng Gao, Chen Ming, Wangjie Hu, Haipeng Li

Contents

Introduction

This manual describes how to use FastEPRR, an R package for estimating the population recombination rate.

FastEPRR has the cross-platform compatibility since R can run on different operating systems. The next section describes how to install R environment and FastEPRR package on different operating systems. The following sections describe the parameters of main functions (see Main function and arguments specification) and several modes to run the main function under different operating systems (see Usage). The manual is divided into several subsections according to different operation systems and functions, and the user can read relevant sections directly.

All packages of FastEPRR and relevant example files are available at http://www.picb.ac.cn/evolgen/softwares/FastEPRR.html. If you have any questions about usage or report bugs, please feel free to contact gaofeng@picb.ac.cn or lihaipeng@picb.ac.cn.

How to cite

Please cite FastEPRR as following:

GAO Feng, MING Chen, HU Wangjie, LI Haipeng* (2016). New software for the fast estimation of population recombination rates (FastEPRR) in the genomic era. ***G3-Genes Genomes Genetics***, Early Online March 30, 2016, doi: 10.1534/g3.116.028233.

LIN Kao, FUTSCHIK Andreas, LI Haipeng* (2013). A fast estimate for the population recombination rate based on regression. ***Genetics*** 194:473-484.

Downloading and Installation

FastEPRR is a standard R package, and the user should install the R environment first. If not, here is an official link to download latest version of R software. And here is an official link about how to install R under different operating systems. The minimum required version of R for FastEPRR is 2.14.0.

The subsequent subsections describe how to install FastEPRR package, and the user can read relevant sections according to specific operating system.

Linux/Unix

There are two ways to install FastEPRR package in Linux/Unix. We recommend the first way, since it is convenient and it avoids opening the R console.

[1] Command Line

Once the R environment has been installed, the user can type the following command in the command line to install FastEPRR package. That is,

```
$ R CMD INSTALL "package_path/FastEPRR_1.0.tar.gz"
```

[2] Within R console

Typing "R" in the command line can start the R console. That is,

```
$ R
```

The results of the above command show the basic information about R version, platform and introductions and so on. Then in the R console, the user can type the following command at the R prompt (the > symbol) to install FastEPRR package.

```
>install.packages("package_path /FastEPRR_1.0.tar.gz", repos = NULL, type="source")
```

The *package_path* is the file path of FastEPRR_1.0.tar.gz

Windows

There are three ways to install FastEPRR package in Windows and the first way is recommended. Double clicking R icon in desktop can open the R console. Again, there is some information about R.

[1] Menu bar

"Packages" → "Install packages from local zip file…"

[2] Install package command

Typing the following command at the R prompt (the > symbol). That is,

```
>install.packages("package_path/FastEPRR_1.0.zip", repos = NULL)
```

[3] Command Line

This way avoids opening R console. Click “Start” → “All Programs” → “Accessories” → “Run” to open the Run box, then change to the R directory, and type the following command in the Run box.

```
> R CMD INSTALL "package_path/FastEPRR_1.0.zip"
```

The *package_path* is the file path of FastEPRR_1.0.zip.

Please ignore the warning messages during installation. Note that FastEPRR uses "mboost" package and "seqinr" package (if your input is alignment file). "mboost" package is used to obtain the regression model and "seqinr" package is used to read alignment file. These two are standard R packages. ***FastEPRR will install them automatically when there is a need.*** *This involves downloading packages to your computer; therefore you must have an active Internet connection.*

Input

FastEPRR supports three common alignment formats (fasta, clustal and phylip) and VCF specification. The input data should be phased, and they can have missing data (denoted by ‘?’) or gaps (denoted by ‘-’). FastEPRR considers sites with two or more alleles segregating and indels (insertions and deletions) are excluded for VCF specification.

Main function and arguments specification

Function and arguments for Alignment file

FastEPRR_ALN

The main function for analyzing alignment file is *FastEPRR_ALN()*. The usage of *FastEPRR_ALN()* is described as follows:

```
FastEPRR_ALN (alnFilePath=NULL, format=-1, erStart=NULL, erEnd=NULL, winLength=NULL,
stepLength=NULL, demoParameter=NULL, outputFilePath=NULL)
```

alnFilePath: a character value. It indicates the full path of the alignment file. The file name should end with [.fas|.aln|.phy|.txt].

format: an integer value (1, 2, 3). It is the format number of the input. 1:fasta, 2:clustal, 3:phylip.

erStart: a character value indicating the start position from which to estimate the recombination rate. The leftmost position of the alignment file is 1. If missing, the default value is *NULL*, which means starting from the beginning of the input.

erEnd: a character value indicating the end position relative to *erStart*. The maximum value of *erEnd* is the length of the alignment file. If missing, the default value is *NULL*, which means up to the end of the input. If the user want to estimate the whole input, let *erStart=NULL* and *erEnd=NULL*.

winLength: a character value. If using the sliding-window strategy, it indicates the size of the sliding-window. The unit is 1. If missing, the default value is *NULL*, which means no sliding-window.

stepLength: a character value indicating the length of the sliding step. The unit is 1. If missing, the default value is *NULL*, which means no sliding step. If *stepLength* equals to *NULL*, but *winLength* doesn't equal to *NULL*, *FastEPRR_ALN* will scan the input window by window with the step length equals to *winLength* successively. This equivalent to set *stepLength* equals to *winLength*.

demoParameter: a character value. It indicates the population demographic changes the input has experienced. The format of *demoParameter* is consistent with the Hudson's *ms* format.

outputFilePath: a character value. It indicates the full path of the output file.

Note

[1] *alnFilePath, format* and *outputFilePath* are mandatory.

[2] Except *format,* all of the other arguments are character values, and they should be within double quotation marks (see Examples).

[3] A period is used as a thousand separator for *erStart, erEnd, winLength* and *stepLength* (see Examples).

Examples

The following three examples show different parameter configurations. The user should read them carefully before using this function.

Example 1:

```
FastEPRR_ALN (alnFilePath="D:/exampleData/alignmentExampleFile.fas", format =1,
outputFilePath ="D:/exampleData/alignmentExampleFileRecRate.txt")
```

Comments:

[1] The input is fasta format alignment file, therefore *format =1*.

[2] Estimating the recombination rates of the whole input. And there is no need to set other arguments, such as *erStart* and *erEnd*.

[3] The output file *"alignmentExampleFileRecRate.txt"* contains the estimated recombination rate information.

Example 2:

```
FastEPRR_ALN (alnFilePath="D:/exampleData/alignmentExampleFile.fas", format =1,
winLength="100", stepLength="50", demoParameter ="-G 6.93 –eN 0.1 0.5", outputFilePath ="D:/
exampleData/alignmentExampleFileRecRate.txt")
```

Comments:

[1] The sliding-window strategy is used to estimate the recombination rate of whole input (*erStart=NULL* and *erEnd=NULL*). The size of the sliding-window is 100 (*winLength="100"*), and the step length is 50 (*stepLength="50"*).

[2] The population demographic parameter (*demoParameter ="-G 6.93 –eN 0.1 0.5"*) represent the population has experienced exponential growth. The time expansion time $t = 0.1$, and $N_0/N_1 = 5$, where N_0 and N_1 are the current and ancestral effective population size, respectively.

[3] The output file *"alignmentExampleFileRecRate.txt"* stores the recombination rate information of each window.

Example 3:

```
FastEPRR_ALN (alnFilePath="D:/exampleData/alignmentExampleFile.fas", format =1,
erStart="50", erEnd="1,000", winLength="100", stepLength="50", outputFilePath ="D:/
exampleData/alignmentExampleFileRecRate.txt")
```

Comments:

[1] Using the sliding-window strategy (*winLength="100", stepLength="50"*) to estimate the recombination rate of interval between the position 50 (*erStart="50"*) and the position 1,000 (*erEnd="1,000"*).

[2] *FastEPRR_ALN* uses a period as a thousand separator (*erEnd="1,000"*).

Function and arguments for VCF file

There are three steps for analyzing VCF file. Each step is represented by one function. That is, *FastEPRR_VCF_step1(), FastEPRR_VCF_step2(),* and *FastEPRR_VCF_step3().* The following sections describe the three functions in detail.

FastEPRR_VCF_step1

The first function is *FastEPRR_VCF_step1()*. This function is used to scan the input and store required information into files. The usage of *FastEPRR_VCF_step1()* is:

```
FastEPRR_VCF_step1 (vcfFilePath=NULL, erStart=NULL, erEnd=NULL, winLength= NULL,
stepLength=NULL, winSNPThreshold=30, idvlConsidered="all", idvlChrFormat="[1:1]",
qualThreshold=20, gapFilePath=NULL, srcOutputFilePath=NULL)
```

vcfFilePath: a character value. It indicates the full path of the VCF file. The file name should end with [.vcf|.gz]. The VCF file should include the polymorphic information of only one chromosome.

erStart: a character value indicating the start position from which you want to estimate the recombination rate. The position corresponds to 'POS' column in VCF file. The unit is kilobase (kb). If missing, the default value is *NULL*, which means starting from the beginning of the input.

erEnd: a character value indicating the end position relative to *erStart*. The unit is kb. If missing, the default value is *NULL*, which means up to the end of the input. If you want to estimate the whole input, let *erStart=NULL* and *erEnd=NULL*.

winLength: a character value. It indicates the size of the sliding-window. The unit is kb. Since the input contains polymorphic information of the whole chromosome, *winLength* is not allowed to be *NULL*.

stepLength: a character value indicating the length of the sliding step. The unit is kb. If missing, the default value is *winLength*, this equivalent to set *stepLength* equals to *winLength*. That is, *FastEPRR_VCF_step1* will scan the input window by window with the step length equals to *winLength* successively.

winSNPThreshold: an integer value. It is the minimum number of $\xi_2' + \xi_x'$ for one window. The windows are excluded if the number of $\xi_2' + \xi_x'$ of these windows is less than *winSNPThreshold*. If missing, the default value is 30.

idvlConsidered: a character value. It indicates which individuals in the input are considered. If missing, the default value is "all", which means containing all individuals. If you want to specify interested individuals, please separate each by a semicolon (";"), e.g., NA0007;NA0010;NA0016. This argument is very useful for specific population analysis. Note that you can specify considered chromosome for each individual, see *idvlChrFormat* below.

idvlChrFormat: a character value. It indicates whether one or two chromosomes of each individual are considered. If missing, the default value is "[1:1]", which means including two chromosomes of individuals in *idvlConsidered*. If you only want to consider the first chromosome of diploid or if individuals are haploid, you should set *idvlChrFormat* as "[1:0]". If you only want to consider the second chromosome of diploid, you should set *idvlChrFormat* as "[0:1]". Besides, you can specify considered chromosomes for each individual. For example, *idvlConsidered*

="NA0007[1:0];NA0010[0:1];NA0016", it means containing the first chromosome of NA0007, the second chromosome of NA0010, as to NA0016, it depends on the value of *idvlChrFormat.*

qualThreshold: an integer value indicating the threshold of quality. It corresponds to "QUAL" column in VCF. The SNPs are excluded if the quality of them is less than *qualThreshold*. The default value is 20.

gapFilePath: a character value. It indicates the full path of the gap file. The gap file stores the gap information of the input chromosome (see '*vcfFilePath'*), and it has specified format. Please follow the format in 'chr7_gap_example.txt' file.

srcOutputFilePath: a character value. It is the full path of the output file. The output file contains basic information of each window, such as the number of ξ_2', ξ_x', four summary statistics, start position and end position.

Note

[1] *vcfFilePath, winLength* and *srcOutputFilePath* must be assigned.

[2] Except *qualThreshold* and *winSNPThreshold, vcfFilePath, erStart, erEnd, winLength, stepLength, idvlConsidered, idvlChrFormat, gapFilePath* and *outputFilePath* are character values, and they should be within double quotation marks.

[3] Be careful of the format when you want to specify interested individuals with specific chromosomes. See "Examples".

[4] Please keep in mind that the unit of *erStart*, *erEnd*, *winLength* and *stepLength* are kb. A period is used as thousand separators for them (see Examples).

[5] Appropriate *qualThreshold* can be set to exclude low quality SNPs. The SNPs which belong to indels are excluded automatically during parsing.

[6] These windows which overlap with gap intervals in *gapFilePath* are excluded.

Examples

Example 1:

```
FastEPRR_VCF_step1(vcfFilePath="/exampleData/ALL.chr7.omni_example.vcf.gz  ", erStart="
1,591.421", winLength="50", winDXThreshold=30,
gapFilePath="/exampleData/chr7_gap_example.txt",  srcOutputFilePath="
/CHB/step1_srcData/chr7")
```

Comments:

[1] The above example shows the way to scan all individuals of chromosome 7 for all populations. The example file ALL.chr7.omni_example.vcf.gz is part of the 1000 Genomes phased OMNI data set (Altshuler, et al. 2012), which includes several populations.

[2] The start position is 1591421 (*erStart="1,591.421"*). The end position is the end of the input (*erEnd=NULL*). The parameter *erStart* is within double quotation marks and has a thousand separator.

[3] All individuals (*idvlConsidered="all"*) for all populations are considered.

[4] The sliding-window size is 50 kb (*winLength="50"*), and the step length also is 50 kb (*stepLength=NULL*). The windows are excluded if the number of $\xi_2' + \xi_x'$ of these windows is less than 30 (*winSNPThreshold=30*).

[5] The gap file is chr7_gap_example.txt *(gapFilePath="/exampleData/chr7_gap_example.txt").*

[6] The outputs are stored into file 'chr7' (*srcOutputFilePath="/CHB/step1_srcData/chr7*).

Example 2:

```
FastEPRR_VCF_step1(vcfFilePath="/exampleData/ALL.chr7.omni_example.vcf.gz",
erStart="1,591.421", winLength="50", winDXThreshold=30, idvlConsidered="NA18525[1:0];
NA18526[0:1]; NA18527; NA18528[1:0]; NA18530[0:1]; NA18532",
gapFilePath="/exampleData/chr7_gap_example.txt", srcOutputFilePath="/CHB
/step1_srcData/chr7")
```

Comments:

[1] The above example shows the way to scan specified individuals of chromosome 7 for CHB population.

[2] Six individuals are considered, that is, *idvlConsidered="NA18525[1:0]; NA18526[0:1]; NA18527; NA18528[1:0]; NA18530[0:1]; NA18532"*, which means including the first chromosome of NA18525 and NA18528, including the second chromosome of NA18526 and NA18530, including both chromosomes of NA18527 and NA18532 because *idvlChrFormat* equals to its default value "[1:1]".

Example 3:

```
FastEPRR_VCF_step1(srcFilePath="/exampleData/ALL.chr7.omni_example.vcf.gz",
erStart="50,941.421", erEnd="159,091.420", winLength="20", stepLength="10",
gapFilePath="/exampleData/chr22_gap.txt", srcOutputFilePath="/CHB/step1_srcData/chr22")
```

Comments:

[1] The above example shows the way to scan specified interval of all individuals of chromosome 7 for all populations.

[2] The start position of the interval is 50941421 (*erStart="50,941.421"*) and the end position of the interval is 159091420 (*erEnd="159,091.420"*).

[3] The size of the sliding-window is 20 kb (*winLength="20"*), and the step length is 10 kb (*stepLength="10"*). Since the window size is larger than step length, the sliding-windows are overlapping.

FastEPRR_VCF_step2

The second function is *FastEPRR_VCF_step2()*. This function is used to estimate the recombination rate for each window. Only one model is constructed for specific ξ_2', ξ_x' configuration, and this model is used to estimate the recombination rate of all windows which have the same number of ξ_2' and ξ_x'. The usage of *FastEPRR_VCF_step2()* is:

```
FastEPRR_VCF_step2 (srcFolderPath=NULL, jobNumber=1, currJob=1, demoParameter=NULL,
DXOutputFolderPath=NULL)
```

srcFolderPath: a character value. It is the parent directory of *"srcOutputFilePath"* in step 1. This means that it is the folder path which stores the basic information of each window for all chromosomes.

jobNumber: an integer value. It is the number of jobs which are used to perform estimating in parallel. If missing, the default value is 1, which means there is only one job. If *jobNumber=100*, then *FastEPRR_VCF_step2 ()* must be executed 100 times. See "Examples".

currJob: an integer value. It denotes the current job number. If missing, the default value is 1, which means the first job. If *jobNumber=100*, then *currJob* should be equal to 1, 2… 100 in turn. See "Examples".

demoParameter: a character value. It indicates the population demographic changes the input has experienced. The format of *demoParameter* is consistent with the Hudson's *ms* format.

DXOutputFolderPath: a character value. It stores the output files in this step. The output files are named by unique ξ_2' and ξ_x'. Note that all of the jobs should have the same value of *DXOutputFolderPath*. Since the output files in *DXOutputFolderPath* will be as the input in the next step.

Note

[1] *srcFolderPath* and *DXOutputFolderPath* are not allowed to be NULL.

[2] *jobNumber* and *currJob* are integer values. *srcFolderPath, demoParameter, DXOutputFolderPath* are character values, and they should be within double quotation marks.

[3] The total job number is *jobNumber,* therefore the minimum value of *currJob* is 1 and the maximum value of *currJob* is *jobNumber*.

Examples

Example 1:

```
FastEPRR_VCF_step2 (srcFolderPath="/CHB/step1_srcData", jobNumber=100, currJob=1,
DXOutputFolderPath="/CHB/step2_proData/DXoutput")
```

……..

```
FastEPRR_VCF_step2 (srcFolderPath="/CHB/step1_srcData", jobNumber=100, currJob=7,
DXOutputFolderPath="/CHB/step2_proData/DXoutput")
```

……..

```
FastEPRR_VCF_step2 (srcFolderPath="/CHB/step1_srcData", jobNumber=100, currJob=100,
DXOutputFolderPath="/CHB/step2_proData/DXoutput")
```

Comments:

[1] One hundred jobs (*jobNumber=100*) are used in total. Here only three jobs which numbered by 1, 7 and 100 are listed.

[2] The input for step 2 is the parent directory of *"srcOutputFilePath"* in step 1, that is *srcFolderPath="/CHB/step1_srcData"*.

[3] The three jobs must have the same output path (*DXOutputFolderPath = "/CHB/ step2_proData /DXoutput"*).

Example 2:

```
FastEPRR_VCF_step2 (srcFolderPath="/CHB/step1_srcData", demoParameter ="-G 6.93 –eN 0.1
0.5", DXOutputFolderPath="/CHB/step2_proData /DXoutput")
```

Comments:

[1] Only one job is available to perform estimating (*jobNumber* equals to its default vale 1).

[2] There is no need to set the *currJob* value since *jobNumber=1*.

[3] The population demographic parameter (*demoParameter ="-G 6.93 –eN 0.1 0.5"*) represent the population has experienced exponential growth. The time expansion time $t = 0.1$, and $N_0/N_1 = 5$, where N_0 and N_1 are the current and ancestral effective population size, respectively.

FastEPRR_VCF_step3

The third function is *FastEPRR_VCF_step3()*. This function is used to merge the files which generated by step 2 as the final file for each chromosome. The final file for each chromosome

includes the recombination rate information of all sliding windows. The usage of *FastEPRR_VCF_step3()* is:

```
FastEPRR_VCF_step3 (srcFolderPath=NULL, DXFolderPath=NULL,
finalOutputFolderPath=NULL)
```

srcFolderPath: a character value. It is the parent directory of *"srcOutputFilePath"* in step 1. This means that it is the folder path which stores the basic information of each window for all chromosomes.

DXFolderPath: a character value. It is the value of *"DXOutputFolderPath"* in step 2.

finalOutputFolderPath: a character value. It is the folder path which stores the recombination rate of all sliding windows for each chromosome.

Note

[1] All of the parameters (*i.e.*, *srcFolderPath*, *DXFolderPath* and *finalOutputFolderPath*) must be assigned, and they should be within double quotation marks since they are character values.

Examples

Example 1:

```
FastEPRR_VCF_step3 (srcFolderPath="/CHB/step1_srcData", DXFolderPath=
"/CHB/step2_proData/DXoutput", finalOutputFolderPath="/CHB/step3_outputData")
```

Comments:

[1] The parent directory of the output of step 1 is used as the value of *srcFolderPath* (*srcFolderPath="/CHB/step1_srcData"*). The output directory of step 2 is used as the value of *DXFolderPath* (*DXFolderPath= "/CHB/step2_proData/DXoutput"*).

Usage

Linux/Unix

There are three modes to run FastEPRR under Linux/Unix: bash script, batch mode and within R console. Running R commands from a bash script allows submitting them to a computer cluster, which is suitable for parallel executing. For the batch mode, R provides a standard script (R CMD BATCH) for running R commands from a file and capturing the output in another file. Therefore, the batch mode allows executing a set of commands in sequence.

The following subsections describe batch mode and within R console mode for running alignment file and bash script mode for running vcf file, respectively.

ALN

The example corresponds to the 'Examples' in Function and arguments for Alignment file section.

[1] Batch mode

First, the following commands should be written in a file, for example, batchExample_ALN.R.

```
library(FastEPRR)
FastEPRR_ALN(alnFilePath="/exampleData/alignmentExampleFile.fas", format=1, erStart="50",
erEnd="1,000", winLength="100", stepLength="50",
outputFilePath ="/exampleData/alignmentExampleFileRecRate.txt")
```

Then, typing the following command in the command line to run the example.

```
$ R CMD BATCH batchExample_ALN.R
```

The outputs which generated by FastEPRR_ALN() are stored in the output file called alignmentExampleFileRecRate.txt automatically.

In addition, you can execute the above command in the background.

```
$ R CMD BATCH batchExample_ALN.R &
```

[2] Within R console

Typing "R" in the command line can open the R console. Then in the R console, the user can type the following commands at the R prompt (the > symbol).

```
>library(FastEPRR)
>FastEPRR_ALN(alnFilePath="/exampleData/alignmentExampleFile.fas", format =1, erStart="50",
erEnd="1,000", winLength="100", stepLength="50",
outputFilePath ="/exampleData/alignmentExampleFileRecRate.txt")
```

VCF

The following example shows how to execute three steps for VCF file on a Linux cluster. Suppose that a grid computing computer cluster software system (e.g. Oracle Grid Engine) is available, and jobs can be submitted via Linux commands (e.g. qsub).

The example corresponds to the 'Examples' in Function and arguments for VCF file section. The purpose of this example is estimating the recombination rate of all autosomes for CHB population.

[1] Step1

One bash script is created for each chromosome, for example, CHB_step1_chr1.sh.

```
#!/bin/bash
/R_Path/R --no-save <<EOF
library(FastEPRR)
FastEPRR_VCF_step1(vcfFilePath="/exampleData/ALL.chr1.omni_example.vcf.gz",
erStart="5,995.669", winLength="50", winDXThreshold=30, gapFilePath="/exampleData/
chr1_gap_example.txt", srcOutputFilePath="/CHB/step1_srcData/chr1")
EOF
```

The *R_Path* should be changed as the actual R software path.

And, the CHB_step1_chr22.sh is described as follows:

```
#!/bin/bash
/R_Path/R --no-save <<EOF
library(FastEPRR)
FastEPRR_VCF_step1(vcfFilePath="/exampleData/ALL.chr22.omni_example.vcf.gz",
erStart="20,412.698", winLength="50", winDXThreshold=30, gapFilePath="/exampleData/
chr22_gap_example.txt", srcOutputFilePath="/CHB/step1_srcData/chr22")
EOF
```

Then change the permission of CHB_step1_chrx.sh to be executable. And submit the above files using qsub command, as described in CHB_step1_submit.sh.

```
#!/bin/bash
for((i=1;i<23;i=i+1));
do
chmod +x CHB_step1_chr${i}.sh
qsub -cwd -q linux.q CHB_step1_chr${i}.sh
done
```

linux.q is the name of destination queue and it should be changed.

At last, execute CHB_step1_submit.sh using the following command.

```
$ sh CHB_step1_submit.sh
```

[2] Step 2

When all bash scripts have been done in step1, the output file for each chromosome has been stored into "/CHB/step1_srcData". Next, CHB_step2.sh which contains 100 jobs is submitted by CHB_step2_submit.sh. The file content of CHB_step2.sh is shown as:

```
#!/bin/bash
/R_Path/R --no-save <<EOF
library(FastEPRR)
FastEPRR_VCF_step2(srcFolderPath="/CHB/step1_srcData", jobNumber=100, currJob= ${1},
DXOutputFolderPath="/CHB/step2_proData/DXoutput")
EOF
```

The CHB_step2_submit.sh is described as follows:

```
#!/bin/bash
for((i=1;i<101;i=i+1));
do
qsub -cwd -q linux.q CHB_step2.sh ${i}
done
```

Then, execute CHB_step2_submit.sh via the following command.

```
$ sh CHB_step2_submit.sh
```

[3] Step 3

The outputs of step 2 are stored into "/CHB/step2_proData/DXoutput" when one hundred jobs have been done. The CHB_step3.sh is described as follows:

```
#!/bin/bash
/R_Path/R --no-save <<EOF
library(FastEPRR)
FastEPRR_VCF_step3(srcFolderPath="/CHB/step1_srcData", DXFolderPath=
"/CHB/step2_proData/DXoutput", finalOutputFolderPath="/CHB/step3_outputData")
EOF
```

Now, submit CHB_step3.sh:

```
$ qsub –cwd –q linux.q CHB_step3.sh
```

When CHB_step3.sh has been done, the final recombination rates of sliding-windows for each chromosome are stored into "/CHB/step3_outputData".

Windows

There are two modes to run functions in FastEPRR package under Windows: batch mode and within R console. For the batch mode, R provides a standard script (R CMD BATCH) for running R commands from a file and capturing the output in another file. Therefore, the batch mode allows executing a set of commands in sequence. For within R console mode, the R software must be started first.

The following subsections describe within R console mode for running alignment file and batch mode for running vcf file, respectively.

ALN

The example corresponds to the 'Examples' in Function and arguments for Alignment file section.

First double clicking R icon in desktop to open the R console, then in the R console, the user can type the following commands at the R prompt (the > symbol).

```
>library(FastEPRR)
>FastEPRR_ALN(alnFilePath="D:/exampleData/alignmentExampleFile.fas", format =1,
erStart="50", erEnd="1,000", winLength="100", stepLength="50",
outputFilePath ="D:/exampleData/alignmentExampleFileRecRate.txt")
```

VCF

The following example shows how to execute the three steps for VCF file in batch mode. Suppose that there are two computers A and B available, and the first two steps are executed on

the two computers simultaneously. The example corresponds to the 'Examples' in Function and arguments for VCF file section. The purpose of this example is estimating the recombination rate of all autosomes for CHB population.

[1] Step1

Computer A is used to scan the first eleven chromosomes (CHB_step1_firEleChrs.R), and computer B is used to scan the rest (CHB_step1_lastEleChrs.R).

CHB_step1_firEleChrs.R is described as follows:

```
library(FastEPRR)
FastEPRR_VCF_step1(vcfFilePath="D:/exampleData/ALL.chr1.omni_example.vcf.gz",
erStart="5,995.669", winLength="50", winDXThreshold=30, gapFilePath=" D:/exampleData/
chr1_gap_example.txt", srcOutputFilePath="D:/CHB/step1_srcData_A/chr1")

…# omitting nine chromosomes

FastEPRR_VCF_step1(vcfFilePath=" D:/exampleData/ALL.chr11.omni_example.vcf.gz",
erStart="5,189.256", winLength="50", winDXThreshold=30, gapFilePath= "D:/exampleData/
chr11_gap_example.txt", srcOutputFilePath="D:/CHB/step1_srcData_A/chr11")
```

CHB_step1_ lastEleChrs.R is described as follows:

```
library(FastEPRR)
FastEPRR_VCF_step1(vcfFilePath=" D:/exampleData/ALL.chr12.omni_example.vcf.gz",
erStart="5,064.079", winLength="50", winDXThreshold=30, gapFilePath="D:/exampleData/
chr12_gap_example.txt", srcOutputFilePath="D:/CHB/step1_srcData_B/chr12")

…# omitting nine chromosomes

FastEPRR_VCF_step1(vcfFilePath=" D:/exampleData/ALL.chr22.omni_example.vcf.gz",
erStart="20,412.698", winLength="50", winDXThreshold=30, gapFilePath="D:/exampleData/
chr22_gap_example.txt", srcOutputFilePath="D:/CHB/step1_srcData_B/chr22")
```

Click "Start" → "All Programs" → "Accessories" → "Run" to open the Run box, and then change to the R directory, and type the following command in the Run box for computer A and computer B separately.

For computer A:

```
> R CMD BATCH CHB_step1_firEleChrs.R
```

For computer B:

```
> R CMD BATCH CHB_step1_ lastEleChrs.R
```

When the two commands have been done, the user must put the output files together manually. That is, merge the folder "D: /CHB/step1_srcData_A" of computer A and "D: /CHB/step1_srcData_B" of computer B as "D:/CHB/step1_srcData".
The folder "D:/CHB/step1_srcData" should contain output files for all chromosomes.

[2] Step 2

In this step, each computer can be viewed as a job; therefore the number of jobs is 2 (*i.e.* jobNumber=2). Computer A can serve as job 1 (CHB_step2_job1.R) and computer B can serve as job 2 (CHB_step2_job2.R).

CHB_step2_job1.R is described as follows:

```
library(FastEPRR)
FastEPRR_VCF_step2(srcFolderPath="D:/CHB/step1_srcData", jobNumber=2, currJob=1,
DXOutputFolderPath="D:/CHB/step2_proData_A/DXoutput")
```

CHB_step2_job2.R is described as follows:

```
library(FastEPRR)
FastEPRR_VCF_step2(srcFolderPath="D:/CHB/step1_srcData", jobNumber=2, currJob=2,
DXOutputFolderPath="D:/CHB/step2_proData_B/DXoutput")
```

Next, typing the following command in the Run box for computer A and computer B separately.

For computer A:

```
> R CMD BATCH CHB_step2_job1.R
```

For computer B:

```
> R CMD BATCH CHB_step2_job2.R
```

[3] Step 3

The role of step3 is merging. This work can be done by one computer, for example, computer A. Again, when the step2 has been done, the user must put the output files in the output folder of computer B ("D:/CHB/step2_proData_B/DXoutput") into the output folder of computer A ("D:/CHB/step2_proData_A/DXoutput") manually.

CHB_step3.R is used to merge in step3, and it is described as follows:

```
library(FastEPRR)
FastEPRR_VCF_step3(srcFolderPath="D:/CHB/step1_srcData", DXFolderPath=
"D:/CHB/step2_proData_A/DXoutput", finalOutputFolderPath="D:/CHB/step3_outputData")
```

Now, for computer A, execute CHB_step3.R:

```
> R CMD BATCH CHB_step3.R
```

When CHB_step3.R has been done, the final recombination rates of sliding-windows for each chromosome are stored into "D:/CHB/step3_outputData".

References

Altshuler DM, Durbin RM, Abecasis GR, Bentley DR, Chakravarti A, Clark AG, Donnelly P, Eichler EE, Flicek P, Gabriel SB, et al. 2012. An integrated map of genetic variation from 1,092 human genomes. Nature 491:56-65.

K. Lin, A. Futschik, and H. Li (2013) A fast estimate for the population recombination rate based on regression, *Genetics*, 194, 473-484.

P. Buehmann and T. Hothorn (2007) Boosting algorithms: regularization, prediction, and model fitting. *Stat. Sci.*, 22, 477-505.

D. Charif and J. R. Lobry (2007) SeqinR 1.0-2: A Contributed Package to the R Project for Statistical Computing Devoted to Biological Sequences Retrieval and Analysis. Structural Approaches to Sequence Evolution. *Biological and Medical Physics, Biomedical Engineering*, pp 207-232

Hudson, R.R. (2002) Generating samples under a Wright-Fisher neutral model of genetic variation, *Bioinformatics*, 18, 337-338.