Description:

search_ dbNSFP20.class is a java program for querying dbNSFP2.0 on your local machine. This program is developed by Dr. Xiaoming Liu at the University of Texas Health Science Center at Houston. It is available for use without charge and without warranty. This document is a simple instruction of the usage of the program.

Release:

Version 2.0, released February 25, 2013

Files:

search_dbNSFP20.class	- the java program for querying database
tryhg19.in	- an example input file with hg19 genome positions
tryhg18.in	- an example input file with hg18 genome positions
search_ dbNSFP20.readme.pdf	- this file

Prerequisite:

A proper Java Runtime Environment should be installed on the machine which hosts dbNSFP20. To check the availability and version of the java on the machine, on the command-line type:

```
java -version
```

If the system found the Java Runtime Environment, a version number and other information will be shown on the screen. search_ dbNSFP20 is written with java version 1.7, although it should work fine with java 1.5 or upper.

Input file format:

Currently, the search program supports two formats of file: vcf format and custom format. The program automatically recognizes vcf file if its extension is ".vcf" and it will query the database by "chr pos ref alt" (see below). Custom input file contains one or more lines. Each line represents a query. A query can be a

1. A genome position: A genome position is represented by "chr pos", where chr is the chromosome number and pos is position on chromosome (default as to hg19). For example,
```
        10      93001
```
2. A non-synonymous SNP (NS): A NS can be represented by "chr pos ref alt", where ref is the reference allele and alt is the alternative allele. For example,

```
10      93001   G       A
```

That is, the SNP on chromosome 10 at position 93001 with reference allele G and alternative allele A. A NS can also be represented more specifically by "chr pos ref alt refAA altAA", where refAA is the reference amino acid and altAA is the alternative amino acid. For example,

```
10      93001   G       A       A       V
```

Alternatively, users may specify the NS based on amino acid change referring to a Uniprot ID or accession number. For example,

```
Uniprot:Q02040:I376F
Uniprot:SF17A_HUMAN:I376F
```

3. A gene ID: A gene ID can be a gene name (HGNC symbol), Entrez id, Uniprot id or accession number, or Ensembl gene id or transcript id. Database name is needed for Entrez, Uniprot and Ensembl. For examples,

```
ZFY
Entrez:132884
Uniprot:ADT3_HUMAN
Uniprot:Q02040
Ensembl:ENSG00000173876
Ensembl:ENST00000447903
```

Usage:

1. Download dbNSFP20.zip on to your machine and unzip it to a directory (for example, mydbNSFP). The total size of the unzipped database is about 27 Gb.
2. Prepare an input file, for example, "tryhg19.in". See above for file format. Put it into mydbNSFP.
3. Enter command-line environment, change directory to mydbNSFP. Type:

```
java search_ dbNSFP20 -i tryhg19.in -o tryhg19.out
```

This command specifies the input file as "tryhg19.in" and the output file as "tryhg19.out". If you put the input file in working directory other than the directory where dbNSFP files locate, i.e. mydbNSFP, you need to put that working directory into the "PATH" variable of your system, or replace try.in with <directory>try.in. Similarly, you can use <directory>try.out to specify the destination directory of the output file.

4. When the input file is large, Java may report memory insufficiency. In that case, try specify a larger memory for Java, for example:

```
java -Xmx5g search_ dbNSFP20 -i tryhg19.in -o tryhg19.out
```

Advanced usage:

1. Specify the human genome reference sequence: By default, search_ dbNSFP20 use human genome reference sequence version hg19 to interpret the chromosome position. To query genome positions or NSs according to version hg18, you can specify the human genome reference sequence by using the option "-v hg18". For example,

   ```
   java search_ dbNSFP20 -i tryhg18.in -o tryhg18.out -v hg18
   ```

2. Specify the chromosomes to search: By default, search_ dbNSFP20 searches all chromosomes if some queries do not contain the chromosome information (for example, when querying a gene id). You can specify the chromosomes to search by using the option "-c list_of_chromosomes_to_search", where list_of_chromosomes_to_search is a list of chromosome numbers separated by commas. For example,

   ```
   java search_ dbNSFP20 -i tryhg19.in -o tryhg19.out -c 1,2,3,10,Y
   java search_ dbNSFP20 -i tryhg18.in -o tryhg18.out -v hg18 -c 1,2,3,10,Y
   ```

3. Specify the columns to output: By default, search_ dbNSFP20 outputs all columns. You can specify the columns to output by using the option "-w list_of_columns_to_write", where list_of_columns_to_write is a list of column numbers separated by commas, and continuous number block can be simplified by begin-end. For example,

   ```
   java search_ dbNSFP20 -i tryhg19.in -o tryhg19.out -w 1-6,8
   ```

   will output columns 1 to 6 and 8.

Output:

The output file contains all NSs that match the query. By default all columns of dbNSFP2.0_variant and most columns of dbNSFP2.0_gene (except the first three columns). User can specify the columns to output (see above). All queries that do not have a match in dbNSFP2.0_variant will be written to a ".err" file.

Contact:

Xiaoming Liu, Ph.D.

Assistant Professor,
Human Genetics Center,
School of Public Health,
The University of Texas Health Science Center at Houston.

Email: xmliu.uth@gmail.com