User manual for *CalcABS*

Xiaoheng Cheng and Michael DeGiorgio

November, 2017

Contents

1 Introduction

CalcABS is a python script for calculating the ancestral branch statistic (ABS; Cheng et al. (2017), Submitted) of a genomic region, to summarize the internal branch length of an unrooted four-population tree.

If you experience any issues when running the script, then please contact Xiaoheng Cheng at `xh.cheng@psu.edu` for further help.

Please cite it as:

X. Cheng, C. Xu, M. DeGiorgio. Fast and robust detection of ancestral selective sweeps. *Mol. Ecol.*. 2017. doi: 10.1111/mec.14416.

2 Operation

We distribute *CalcABS* in compressed (tar.gz) format. In addition to the *CalcABS* script, we also included the user manual and example data. The script included is designed to perform on a UNIX system. To unpack *CalcABS* from the command line, go to the directory where it is stored, and enter

```
tar -xzvf CalcABS.tar.gz
cd CalcABS/
```

The first command will decompress the file and release the content into folder `CalcABS` in the current directory. The second command will lead the user to the `CalcABS` directory, which contains the script, manual, and `test` directory.

To run `CalcABS.py`, format the dash commands and their arguments in a single line as

```
python CalcABS.py -i <input file> -o <output file> -n <c,p,x> -w <window size>
   -s <step size> [--check] [--fix <pair>]
```

In this command line, `<input file>` and `<output file>` are the file names (including their paths) to input and output files, respectively. They must follow their respective dash commands, `-i` and `-o`. The arguments of `<window size>`, following `-w`, and `<step>`, following `-s`, are the lengths of the sliding window and the step it takes while sliding through the data, respectively. Both numbers should be the length in the number of nucleotides (nt). When choosing window and step sizes, the user should consider the density of polymorphic sites in the input data. More discussion on optimizing window sizes can be found in **?** (Submitted).

The `<c,p,x>` argument is for column indices and must follow `-n`. For this argument, the user should provide the column indices of chromosome name (denoted as `c`), locus positions (`p`), and the allele count of the first population (`x`), separated by commas (*i.e.*, `c,p,x`, with no space). If the file does not contain chromosome name, or the user does not need it in the output, then leave out the first index `c`, but still keep the first comma (*i.e.*, `,p,x`).

2.1 Checking format

The command argument `--check` is optional. When the user wishes to check if the input file has the correct format before scanning, provide input file information and the column indices, and include `--check` in the command.

```
python CalcABS.py -i <input file> -n <c,p,x> --check
```

2.2 Fixing the topology

Calculating ABS does not need to assume a certain topology relating the four populations. However, if users need to fix the topology, then `--fix` command can be used. Following this command, the user should provide the pair of target populations in the input data. That is, if the four populations listed in the input file are numbered 1, 2, 3 and 4, respectively from left to right, then the target sister populations can be designated by the pair of numbers representing these populations, being “12”, “13”, “14”, “23”, “24”, or “34”.

```
python CalcABS.py -i <input file> -o <output file> -n <c,p,x> -w <window size>
   -s <step size> --fix <pair>
```

2.3 Help page

The user can also use `-h` or `--help` command to see instructions for each dash command.

```
python CalcABS.py -h
python CalcABS.py --help
```

Either of the two commands should print the following output in the command window.

```
Usage:  CalcABS.py -i <input file> -o <output file> -n <c,p,x> -w <window size>
-s <step size> [--check] [--fix <pair>]

Options:
-h, --help            show this help message and exit
-i INFILE, --input=INFILE
                      Path and name of your input file.
-o OUTFILE, --output=OUTFILE
                      Path and name of your output file.
-n INDEX, --indices=INDEX
                      Index numbers for the columns of chromosome name c,
                      locus positions p, and allele counts of the first
                      population x, respectively.  Format of this argument
                      should be ‘c,p,x’ or ‘,p,x’, without space.
-w WIN, --window=WIN  Length of the sliding window, in the number of
                      nucleotides (nt), for ABS scan.
-s STEP, --step=STEP  Length of each step the sliding window takes, in the
                      number of nucleotides (nt), while scanning the data
                      with ABS.
--check               Option to check the file format.
--fix=PAIR            Option to fix the topology formed by population 1, 2,
                      3 and 4.  Argument can be ‘12’, ‘13’, ‘14’, ‘23’, ‘24’,
                      or ‘34’, identifying the pair of target populations.
```

3 Input format

ABS calculation requires allele count data from all four populations at each of their polymorphic sites. To present such data, input files must be tab-delimited plain text files that contain information on no more than one chromosome.

Input and output files are recommended to be presented in absolute paths. If you are unsure about their absolute paths, then copy the script `CalcABS.py` to the same folder where the data are located, and follow the instructions in Section 2 or 5. If the path of the output file is not provided, then by default the output file will be written to the same path as the script.

If the user has used the input file in other operating environment previously, then it is advised to use the following command to make sure this file is readable in a UNIX environment:

```
dos2unix <file>
```

3.1 Input file format

In the input file, the user must provide, for each of the four populations, the number of a certain allele (denoted as `x`) and the total number of alleles sampled (denoted as `n`). Positions of loci must be sorted in ascending order. All the loci included in the file should be polymorphic among the populations, meaning they can either be segregating sites within populations, or fixed differences among different populations. When parsing data for *CalcABS*, the users must only consider bi-allelic loci. For each locus, the chosen allele, whose count is represented by `x`, could be the number of reference alleles, alternate alleles, ancestral alleles, derived alleles, major alleles, or minor alleles at this position. A locus should be discarded if the allele count data are completely absent in any population.

The input file must have a header as its first line, but users can customize the name of each column. Additional information about each locus, for example their IDs, can also be kept in input files, but the file must at least have the following columns.

```
position   x1   n1   x2   n2   x3   n3   x4   n4
```

where `position` shows the physical position of the locus, and `xk` and `nk` denote the counts of a certain allele and the total number of observed alleles, respectively, at this position in population `k`. As in the example above, `x1`, `x2`, `x3`, and `x4` could be the counts of a certain allele at the given position in population 1, 2, 3, and 4, respectively. Note that additional information can be inserted before `position`, between `position` and `x1`, and after `n4`, but cannot be placed between the columns of allele counts. For each population, denoted as population `k`, its total sample size `nk` must immediately follow its allele count `xk`.

3.2 Providing column indices

Users should provide column indices for chromosome name (denoted as `c`), locus positions (denoted as `p`), and `x1` (denoted as `x`) to inform *CalcABS* of the structure of the list. Indexing begins with one, and the indices must be integers, separated by commas (*i.e.*, `c,p,x`).

The number `c`, the column index for the chromosome name, can be left blank if the user does not have such column in the input, or does not need it in the output. For example, for the header provided in Section 5.2, its corresponding `<c,p,x>` argument would be `1,2,3` or `,2,3`. Note that the inclusion of chromosome name is solely for the convenience or preference of the user. All the loci included in the file should reside on the same chromosome, such that their positions are comparable. In other words, when included, it should only have one value throughout each input file.

When the input file and column indices are provided, the user can use `--check` command (see Section 2.1) to check if they are formated correctly.

4 Output format

CalcABS writes output to the path and file name provided in the `-o <output file>` argument. Dependent on whether the column index of the chromosome name is given, the output can be in two formats: with the chromosome name or without. When the input file contains the chromosome name and the user indicates its column index in the `-n <c,p,x>` argument, the output format is

```
chr   midPos   score   numSites   topology
```

When the column index of chromosome name is not given, the output format is

```
midPos   score   numSites   topology
```

In these tables, `chr` denotes the chromosome name, `midPos` is the center position of the sliding window, `score` is the ABS score of this window, `numSites` is the number of sites included in this window, and `topology` indicates the four-population tree topology suggested by the data in this window. The value for `topology` will be "`12|34`", "`13|24`", or "`14|23`", with the numbers representing the order that the four populations are listed in the input file, and "|" denotes the internal branch separating the two pairs of populations.

5 Examples

To familiarize the users with *CalcABS*, we include example input files in the subfolder `test`. Both examples are the parsed allele count data for a simulated 1 million base (or 1 Mb) sequence. A strongly advantageous allele (selective coefficient $s = 0.1$) was introduced at the center of this sequence on a single haplotype before the pair of sister populations, 1 and 2, diverged. We sampled 50 haplotypes from each population, and denote the counts of derived alleles in each population as `x`, and the total numbers of alleles observed as `n`.

5.1 Input file without chromosome name

In the `test` folder, you should see the file `ex1_input.txt` that has the following layout.

```
position  x1  n1  x2  n2  x3  n3  x4  n4
24.0      0   50  0   50  0   50  2   50
119.0     1   50  0   50  0   50  0   50
402.0     0   50  0   50  1   50  0   50
608.0     26  50  19  50  37  50  28  50
828.0     0   50  0   50  5   50  11  50
1064.0    0   50  0   50  2   50  0   50
1293.0    0   50  0   50  0   50  2   50
...
```

Follow instructions in Section 2 to unpack the `CalcABS.tar.gz` file and navigate to the `CalcABS` folder. Use the following command to scan this file with a window size of 20,000 bases (or 20 kb) at 1 kb step. This command will also write the output to file `test/ex1_out.txt`:

```
python CalcABS.py -i test/ex1_input.txt -n ,1,2 -o test/ex1_out.txt -w 2e4 -s 1000
```

Once the script finishes running, you will find the output file `ex1_out.txt` in the `test` folder. The file should begin with

```
midPos  score           numSites  topology
10000   0.238267230009  192       12|34
11000   0.239424139258  199       12|34
12000   0.237528418872  206       12|34
13000   0.228974064723  206       12|34
14000   0.208281589288  216       12|34
15000   0.18058099615   220       12|34
...
```

We can also fix the topology so that populations 1 and 2, the target population pair, are always on the same side of their ancestral branch. The following command can be used, and its output should be identical to the previous one.

```
python CalcABS.py -i test/ex1_input.txt -n ,1,2 -o test/ex1_fix.txt -w 2e4 -s 1000
   --fix 12
```

5.2 Input file with chromosome name

In addition to all the data in example 1, `ex2_input.txt` contains a chromosome name for this sequence. Because this is a simulated sequence, we name the chromosome by the order it was generated. This file begins with:

```
chr  position  x1  n1  x2  n2  x3  n3  x4  n4
12   24        0   50  0   50  0   50  2   50
12   119       1   50  0   50  0   50  0   50
12   402       0   50  0   50  1   50  0   50
12   608       26  50  19  50  37  50  28  50
12   828       0   50  0   50  5   50  11  50
12   1064      0   50  0   50  2   50  0   50
12   1293      0   50  0   50  0   50  2   50
...
```

For this input file, the user can use the following command to generate an output file `ex2_out.txt` with a chromosome name.

```
python CalcABS.py -i test/ex2_input.txt -n 1,2,3 -o test/ex2_out.txt -w 2e4 -s 1000
```

The output file should begin with:

```
chr  midPos  score           numSites  topology
12   10000   0.238267230009  192       12|34
12   11000   0.239424139258  199       12|34
12   12000   0.237528418872  206       12|34
12   13000   0.228974064723  206       12|34
12   14000   0.208281589288  216       12|34
12   15000   0.18058099615   220       12|34
...
```

If the user does not need the `chr` column in the output file, then by leaving blank the corresponding number in the `<c,p,x>` argument, the following command can be used to generate an output file identical to `ex1_out.txt`:

```
python CalcABS.py -i test/ex2_input.txt -n ,2,3 -o test/ex2_out.txt -w 2e4 -s 1000
```

To fix the topology so that populations 1 and 2 are always on the same side of their ancestral branch, the user can add `--fix 12` to the command, as described in Section 5.1.

References

X. Cheng, C. Xu, and M. DeGiorgio. Fast and robust detection of ancestral selective sweeps. *Mol. Ecol.*, 2017. doi: 10.1111/mec.14416.