SCCT Documentation

Release 1.0

(Wallace)wavefancy@gmail.com

February 21, 2014

Brief introduction

SCCT (**S**election detection by **C**onditional **C**oalescent **T**ree) is an efficiency computational software developed to detect recent positive selection using deep sequencing data. It's robust to various demographic events and also robust to the variations of mutation rates and recombination rates. This method is also a powerful method, which has power comparable to iHS [1] method. SCCT, however, improved the ability to pinpoint selective causal sites, facilitated with other variant annotations, which can greatly help geneticists to explore the mechanisms behind positive selection events.

[1] Voight, B. F., Kudaravalli, S., Wen, X., & Pritchard, J. K. (2006). A map of recent positive selection in the human genome. PLoS Biology, 4(3), e72. doi:10.1371/journal.pbio.0040072.

CHAPTER

ONE

INSTALLATION AND ENVIRONMENT SETTING

1.1 Environment Setting

SCCT software package was developed by JAVA, and also scripted with some PYTHON code. Before we run this program, we need to set the running environment for Java and Python, and also make them meet the minimum version we need.

1.1.1 Java installation

We recommend the Java version should be >=1.6. If you don't have Java installed or your Java version is lower than 1.6. Please install or update your Java, you may need help from [Java SE 6 Platform Installation] (http://www.oracle.com/technetwork/java/javase/index-137561.html)

After the installation, you can check Java version by (command starts by ">" prompt):

```
>java -version
java version "1.7.0_21"
Java(TM) SE Runtime Environment (build 1.7.0_21-b11)
Java HotSpot(TM) 64-Bit Server VM (build 23.21-b01, mixed mode)
```

1.1.2 Python installation

All our python code were tested based python version 2.7.5. We do not guarantee these code could be run on the python 3.0+ version. Python version 2.7+ can be downloaded from (http://www.python.org/).

CHAPTER

TWO

A TOY EXAMPLE TO PLAY THE SCCT PROGRAM

This section, we will use an example to show how to run the SCCT program, and what these parameters mean. Totally, we need five steps to get the final results.

2.1 STEP 1. Count mutations

First, we need to count the number of mutations for derived allele group and ancestral allele group.

```
>java -jar CountTwoGroupMutations.V1.0.jar  --help
--------------------------------
    CountTwoGroupMutationsFromMsData    version: V1.0     Author:wavefancy@gmail.com
--------------------------------
Usages:
Read ms format file directly from std., and output results to stdout.
Parameter1(int): Number of threads used for computing.
Parameter2(int): Number of SNPs for flanking size.
Notes:
1. Input file can contain multiple ms datasets. Output will be separated by title.
2. Multiple threads work at SNP level.
3. Frequency represents derived allele frequency in the output file.
4. Output File Title: #POS      DE_FRE  DerivedCount    AncestralCount  STATES
--------------------------------
```

Note: For all the programs or scripts, if you don't know how to run it, use the `--help` flag to get help.

An example for ms format input file

```
//
segsites: 10
positions: 0.1 0.11 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
1100001101
1000000101
0010000010
1101111000
0101111111
0010011101

//
segsites: 10
positions: 1 1.1 2 3 4 5 6 7 8 9
1100001101
1000000101
0010000010
```

```
1101111000
0101111111
0010011101
```

This example contains two dataset, each dataset contains 6 haplotypes and 10 SNP sites. Input file should be prepared in this format in order to feed the SCCT program. **Ancestral allele was coded as 0, derived allele was coded as 1.**

2.1.1 An example to count mutations

```
#Count the number of mutations for derived and ancestral group.
>zcat selectionMs.gz | java -jar CountTwoGroupMutations.V1.0.jar 2 300 | gzip >counts.gz
```

Count mutation using two CPUs with flanking size 300 SNPs.

Let's look at the output file, which contains 5 columns, which are `Position`, `Derived_allele_frequency`, `Mutation_number_for_derived_group`, `Mutation_number_for_ancestral_group`, `States`. `States` column could have four possible values: `L_END`, `R_END`, `MAF`, `OK`. `L_END` or `R_END` represents the sliding window meets the left or right end. `MAF` represents minor allele frequency at this site less than 5%, **we should exclude these sites for following analysis**.

Output file example

```
0.0127440524    0.0416666667    NA      MAF
0.0127621072    0.025   NA      MAF
0.0128805894    0.3833333333    321     276     L_END
0.0128864804    0.15    239     359     L_END
0.0130199967    0.075   180     419     L_END
0.0130305557    0.0166666667    NA      MAF
0.0131779926    0       NA      MAF
0.0133031297    0.475   353     247     OK
0.0134150400    0.0083333333    NA      MAF
0.0136366954    0.1     196     404     OK
0.0136937217    0.0166666667    NA      MAF
0.0137014226    0       NA      MAF
0.0137058251    0.425   355     245     OK
0.0137534133    0.0083333333    NA      MAF
0.0138180315    0.1     198     402     OK
0.0139230681    0.0083333333    NA      MAF
0.0139828531    0.3166666667    312     288     OK
0.0140655685    0       NA      MAF
0.0140828726    0.0083333333    NA      MAF
0.0141015802    0.0666666667    166     434     OK
0.0142287396    0       NA      MAF
0.0142809646    0.0166666667    NA      MAF
0.0143185954    0.2666666667    300     300     OK
0.0143264656    0.0666666667    177     423     OK
```

2.2 STEP 2. Prepare scale ratio file.

Two approaches can be used to generate the scale ratio [1] file, one estimates from empirical data, the other estimates from theoretical equation or by simulation. Here, we show the first approach.

Note: Please be aware that the number of SNPs in this example may not sufficient, the estimated ratio file may

[1] scale ratio is the parameter of (α_i) in the SCCT paper.

have some bias. It's better to used genome-wide data to do empirical estimation. If you don't have enough data to empirically estimate ratio, It's better to get ratio by theoretical equation or simulation. we will show these two approaches later. The power for these two approaches is nearly the same if data is sufficient.

```
>python ComputeScaleRatioV1.1.py

    ----------------------------------------
    Compute Scale Ratio
    ----------------------------------------

    @Author: wavefancy@gmail.com
    @Version: 1.1

    @Usages:
    para1: Column index for frequency
    para2: Column index for derived group values.
    para3: Column index for ancestral group values.

    @Note:
    1. Column index starts from 1.
    2. Read from stdin, and output to stdout.
    3. Tital [Fre   Ratio   Count]
    ----------------------------------------
```

2.2.1 An example to calculate scale ratio file

```
# Generate scale ratio file.
>zcat counts.gz | grep -i 'ok' | python ComputeScaleRatioV1.1.py 2 3 4 >scale_ratio.txt
```

We use the results from step 1 ("`counts.gz`") to select out only those SNPs with `State` labeled by `OK`. Next, use the filtered results to generate the scale ratio file.

A fraction of the scale ratio file

```
Fre     AveragedRatio   Count
0.05    0.3910087279    1215
0.0583333333    0.4023119828    999
0.0666666667    0.4166837593    915
0.075   0.4410834878    812
0.0833333333    0.4453102493    726
0.0916666667    0.4681617508    621
0.1     0.4936401926    602
......
```

- `Column1`: Derived allele frequency.
- `Column2`: the averaged ratio of all the SNPs with the frequency equals value in column 1.
- `Column3`: Number of sites with derived allele frequency equals value in column 1.

2.3 STEP 3. Get the unstandardized SCCT score

```
>python ComputeUnstandardizedSCCTV1.1.py --help

    ----------------------------------------------
```

```
    Compute unstandardized SCCT score
    ---------------------------------------------

    @Author: wavefancy@gmail.com
    @Version: 1.1

    @Usages:
    para1: Column index for allele frequency.
    para2: Column index for derived group values.
    para3: Column index for ancestral group values.
    para4: scale ratio file name.

    @Note:
    1. Add one column for the unstandardized SCCT score.
    2. Column index starts from 1.
    3. Read from stdin and output to stdout.
    ---------------------------------------------
```

2.3.1 An example to calculate the unstandardized SCCT score

```
>zcat counts.gz | grep -i 'OK' | \
python ComputeUnstandardizedSCCTV1.1.py 2 3 4 scale_ratio.txt | gzip >un.scct.gz
```

2.4 STEP 4. Generate profile to standardize SCCT score

We need to partition SNPs into different bins, and normalize them bin by bin. First we need to compute the mean and variance for each bin, then use these values to do normalization.

```
>java -jar StandardizeFileGenerator.V1.0.jar --help
---------------------------------
    StandardizeFileGenerator    version: 1.0     Author:wavefancy@gmail.com
---------------------------------
Usages:
parameter1[int]: Column index for frequency.
parameter2[int]: Column index for value.
parameter2[double]: frequency interval.
[Column index start from 1.]
---------------------------------
```

2.4.1 An example to generate profile for standardization

```
>zcat un.scct.gz | java -jar StandardizeFileGenerator.V1.0.jar 2 6 0.01 >std.profile.txt
```

Last parameter for `StandardizeFileGenerator` is to partition SNPs into 100 bins according to the derived allele frequency, each bin length is `1/100 = 0.01`.

A fraction of the standardization profile file

```
Fre_end Mean    SD.
0.05    -0.0111728071   0.1503143826
0.06    -0.0117042354   0.153914443
0.07    -0.0119474557   0.1555408665
0.08    -0.0121449281   0.1558833834
```

```
0.09    -0.013424928    0.163156773
0.1     -0.0131090985   0.1614803146
......
```

2.5 STEP 5. Get the final standardized SCCT score

This is the final step to get the standardized SCCT score.

```
>java -jar StandardizeFromFile.V1.0.jar --help
--------------------------------
    StandardizerFromFile    version: 1.0     Author:wavefancy@gmail.com
--------------------------------
Usages:
parameter1[int]: Column index for frequency.
parameter2[int]: Column index for value.
parameter2[file]: standardize file.
[Column index start from 1.]
--------------------------------
```

2.5.1 An example to calculate standardized SCCT score

```
>zcat un.scct.gz | java -jar StandardizeFromFile.V1.0.jar 2 6 std.profile.txt \
 | gzip >std.scct.gz
```

A fraction of the standardized SCCT results

```
......
0.4993749899    0.2333333333    305     295     OK      0.2811862567    1.6442567782
0.4994577499    0.55    103     497     OK      -0.9224959757   -2.896511775
0.4995161549    0.075   185     415     OK      0.0105984116    0.1458997053
0.4995349489    0.125   303     297     OK      0.6253335189    3.9171667595
0.4995549065    0.55    104     496     OK      -0.9108199654   -2.858247115
0.4998591260    0.0833333333    252     348     OK      0.4862106580    3.0623036777
0.5000000000    0.5416666667    93      507     OK      -1.1145171193   -3.5258041092
0.5000466282    0.0833333333    250     350     OK      0.4725118137    2.9783424418
0.5006451130    0.5416666667    90      510     OK      -1.1532066642   -3.6525976152
0.5006853125    0.2333333333    320     280     OK      0.3813812291    2.1980254948
0.5008912063    0.1     223     377     OK      0.1808749662    1.2012861455
0.5010037947    0.5583333333    108     492     OK      -0.9153815187   -4.1896412835
0.5019236478    0.075   197     403     OK      0.1027882735    0.7373024571
0.5019959965    0.1583333333    321     279     OK      0.6147690457    3.8160723801
0.5021548072    0.05    197     403     OK      0.2232925640    1.5598332445
0.5029516747    0.0916666667    252     348     OK      0.4361680291    2.7822408491
0.5037488961    0.1     265     335     OK      0.4715476764    3.0013365784
0.5037716610    0.1     265     335     OK      0.4715476764    3.0013365784
......
```

The last column is the standardized SCCT (`std_SCCT`) score. In the neutral circumstances, `std_SCCT` follows standard normal distribution. If we see a list of SNPs with `|std_SCCT| >=2.0` , we may suspect that this region may experienced positive nature selection.

For human genome-wide data, in practice, if we see more than 7 SNPs with `|std_SCCT| >=2.0` in a 50-SNP window, this region could show significant signature of positive selection.

In this toy example, we used msms [2] software to simulated a 3M sequences, and a positive selection event occurred at the middle point of this 3M sequences, therefore, around the position of `0.5000000000` , we can see a list of SNPs have extreme `std_SCCT` score.

That's the end of our toy example. Have a nice day!!!

[2] Ewing, G., & Hermisson, J. (2010). MSMS: a coalescent simulation program including recombination, demographic structure and selection at a single locus. Bioinformatics (Oxford, England), 26(16), 2064–5. doi:10.1093/bioinformatics/btq322.

CHAPTER

THREE

GENERATE SCALE RATIO BY THEORETICAL EQUATION OR BY NEUTRAL SIMULATION

We mentioned in the toy example section, when our empirical data set don't have sufficient number of SNPs, scale ratio (α_i) may be biased by our limited empirical data, this may reduce power to detect selection event. In this case, we can estimate scale ratio by theoretical equation (this theoretical equation was deduced from a constant population size model), or by simulation (sophisticated demography parameters can be incorporated).

3.1 Generate scale ratio file by theoretical equation

```
>java -jar TheoreticalRatioV1.1.jar --help
--------------------------------
    TheoreticalRatio    version: 1.1     Author:wavefancy@gmail.com
--------------------------------
Usages:
parameter1: Number of threads used for computing.
parameter2: Number of Haplotypes
--------------------------------
```

3.1.1 An example to generate scale ratio file by theoretical equation

```
>java -jar TheoreticalRatioV1.1.jar 2 20

derived ratio
0.1     0.014294946136966327
0.15    0.03400296845636232
0.2     0.0584244648909865
0.25    0.08734990527852876
0.3     0.12084393013866404
0.35    0.15916596957257678
0.4     0.20274431803019444
0.45    0.2521777931353374
0.5     0.3082566360638361
0.55    0.37200149966188795
0.6     0.44472369141814994
0.65    0.5281139387813419
0.7     0.6243724095256858
0.75    0.7364013584635845
0.8     0.8680966095070524
0.85    1.0248010001572212
0.9     1.214034247094156
0.95    1.446717004926677
```

Generate scale ratio file for 20 haplotypes using 2 CUPs cores. **Caution: this approach may very memory and time consuming, if want to generate the theoretical file for a large number of haplotypes (eg. 300 haplotypes)**.

Note: If an *Out-of-memory* exception was fired, please use flag `-Xmx` to allow JVM using more memory. eg. `java -jar -Xmx20G TheoreticalRatioV1.1.jar 2 200`.

3.2 Generate scale ratio file by simulation

If you want to generate the scale ratio file for a large number of haplotypes or want to incorporate sophisticated demographic parameters in to the scale ratio. Please use this approach to generate the scale ratio file.

```
>java -jar ComputeRatioFromMS.Simu.AKKA.V2.0.jar --help
--------------------------------
    ComputeRatioFromMS.Simu.AKKA    version: V2.0     Author:wavefancy@gmail.com
--------------------------------
Usages:
Read ms output directly from std.
Parameter1(int): Number of threads used for computing.
[-h|--help]: Output this help
--------------------------------
```

3.2.1 An example to generate scale ratio file by simulation

```
>ms 10 100 -t 400 | java -jar -Xmx10G ComputeRatioFromMS.Simu.AKKA.V2.0.jar 2
deFre   Ratio   Count
0.3     0.0361459576    308
0.4     0.0817171207    178
0.5     0.1736725793    92
0.6     0.2357960826    77
0.7     0.3515625       50
0.8     0.4285816902    35
```

Use ms [1] simulation software to generate simulation data, we simulated 10 haplotypes by 100 time replicates. **Please simulate sufficient replicates to make sure each allele frequency category have sufficient count.** This approach may consume lots of memory, please make sure to allow JVM use enough memory.

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

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*