SplitSup User Guide

E. S. Allman, L. S. Kubatko, and J. A. Rhodes

August 1, 2016

Version 1.0

Contents

Chapter 1

Introduction

This manual describes the use of `SplitSup`, C software that computes split scores as developed in [AKR16].

Under the general Markov model (GM) of base substitution, if DNA sequence data is generated on a tree T, then a matrix flattening of the pattern distribution for any split arising from an edge of T will be approximately of rank 4. Splits not arising from edges should have corresponding matrix flattenings of rank greater than 4. The software `SplitSup` returns split scores for checking if a matrix flattening is ‘close’ to rank 4 or not.

Possible split scores range from 0 (strongest support) to 1 (weakest support). The precise formula for the score, which depends on singular values of the matrix flattening, is given in [AKR16]. Theoretically, if a split occurs on a tree, the split score should approach zero as sequence length is increased.

For some purposes (e.g., mixture models, or a combined coalescent/substitution model as in [CK14]) researchers may wish to compute scores based on approximate rank other than 4. An option allows for this.

Current, only DNA sequence data is allowed. Modifications for amino acid sequences are possible, if a researcher has a compelling need for it.

Chapter 2

Using SplitSup

2.1 Input files

`SplitSup` requires two input files with the names `infile` and `splitsfile`:

File name	Contents
`infile`	DNA sequence data, in PHYLIP format
`splitsfile`	splits whose scores are to be computed

The data in `infile` must be sequential (not interleaved).

The file `splitsfile` is a text file, formatted as follows: The first line of the file is the number of splits for which scores will be computed. Each subsequent line specifies one of the splits, in the format

```
split-size   elements-in-split-separated-by-spaces
```

Here the `elements-in-split-separated-by-spaces` is a list of elements of *one* of the partition sets in the split (the one of size `split-size`). For example, if `splitsfile` were

```
2
5 1 2 3 6 7
3 5 6 7
```

then `SplitSup` will compute split scores for two splits: a 5-split for which one of the partition sets contains taxa 1, 2, 3, 6, and 7 in `infile` and the other partition set contains the remaining taxa, and a 3-split for taxa 5, 6, and 7.

Note that `splitsfile` does *not* use the taxa names, but rather the order of the sequences in `infile`. For instance, if `infile` has, in order, sequences for `human,` `gorilla, chimp, orangutan`, then `human` is taxon 1, `gorilla` is 2, `chimp` is 3, and `orangutan` is 4.

There is some ambiguity in naming splits. In analyzing a dataset with 10 taxa, the splits indicated by the lines

`3 1 2 3` and `7 4 5 6 7 8 9 10`

are identical.

2.2 Command line options

`SplitSup` may be used in two ways, though a variety of analyses fit into these schema. In the first way, `splitsfile` has a large number of splits for multiple scores computations. In the second way, `splitsfile` has only a single split, for a sliding window analysis.

The syntax for these analyses is

```
SplitSup [-rRANK] [-s -wWINDOW_SIZE -oOFFSET_SIZE -mMIN_NUMBER_SITES]
```

where square brackets indicate optional parameters. If a user calls the program with the basic command

```
SplitSup
```

then the two input files are read, and all split scores are written to the output file `scoresfile`, overwriting its previous contents. Details on the format of `scoresfile` are given below.

As an optional parameter, a user may choose the rank `RANK` underlying the score computation. For the GM model of DNA base substitution and its submodels such as GTR, the default value of `RANK`=4 is the appropriate one. For a 2-class mixture of the GM model, the rank of flattenings corresponding to true splits should be 8, so specifying `-r8` is appropriate (see [AR06] for this and similar results). Underlying the software `SVDquartets` of [CK14, CK15] and its model of a combined coalescent and GTR+Γ+I substitution model are matrix flattenings for quartets which are of rank 10, so an `-r10` option can be used to compute relevant scores with `SplitSup`.

The last set of optional parameters is used to select the sliding window analysis, in which split scores are computed for shorter segments of the alignments. For this, the `splitsfile` should contain only one split. Calling the program, for example, with

```
SplitSup -s -w1000 -o100 -m200
```

performs a sliding window analysis with a window size of 1000 bp and a slide step size (offset) of 100 bp, requiring a minimum of 200 non-gapped base pairs in any window for the split score to be computed. The `-s` switch indicates that the sliding window analysis has been selected, and it must be followed by the three additional parameters.

To produce a sliding window analysis comparing several different splits requires running `SplitSup` several times with the `-s` option. For instance, in [AKR16] three splits were considered for some primate data: $HC|GO$, $HG|CO$, and $HO|CG$. A simple script file, with three versions of `splitsfile` was used, with each `scoresfile` copied before the next was produced.

2.3 Output file

`SplitSup` stores scores to an output file named `scoresfile`. The format of this file depends on whether the `-s` option for a sliding window analysis was invoked.

If `-s` was not invoked, so `splitsfile` potentially listed many splits, then `scoresfile` contains a list of scores for these splits. For instance if `splitsfile` had contents

```
12000
3 1 2 3
5 1 2 4 6 7
   ⋮
```
(11998 more splits)

then `scoresfile` would be formatted as

```
12000
3 0.246620643974497
5 0.274404335917279
   ⋮
```
(11998 more scores)

The first line of `scoresfile` indicates the number of scores that were computed. Subsequent lines in the file correspond to the ordered list of splits in `splitsfile` and have two entries: the size of the split and the value of the score.

If `-s` was invoked for a sliding window analysis, say with command

```
SplitSup -s -w10000 -o1000 -m500
```

and `splitsfile`

```
1
2 1 4
```

then `scoresfile` will be formatted as

```
      ⋮
169001  638   623   0.000076033553543
170001  1631  1568  0.000123768383328
171001  2605  2501  0.000765813771734
172001  3600  3451  0.000744102188514
173001  4579  4367  0.001116200230904
174001  5542  5284  0.001562678240580
175001  6521  6225  0.001635966713922
      ⋮
```

Each line of `scoresfile` has the following four entries: *window-start-location*, *number-of-non-gapped-sites-in-window*, *number-of-constant-sites-in-window*, and *split-score-for-window*. To interpret the first line shown above, for example, one sees that the sliding window started at site 169001 in the alignment, had 638 non-gapped sites, 623 constant sites, and the score for the 2-split was 0.000076033553543.

If a window fails to have the minimum required non-gapped sites set by the `-m` switch, there will be no output to `scoresfile` for it.

2.4 Warning

The distributions of split scores of k-splits for differing k are *not* the same. Comparing splits of different sizes by their split scores requires great caution. See [AKR16] for details.

Chapter 3

Getting and citing `SplitSup`

3.1 Sample files

A selection of sample analysis files are included in the directory `sampleFiles` on GitHub for help in learning to use `SplitSup`.

3.2 Required C libraries

The source code and supporting C files needed to compile `SplitSup` are all available at GitHub. Future versions and updates will be uploaded there. To compile `SplitSup`, download the following files from the GitHub repository and place them in the directory where you will compile the software.

```
flatBinary.c
main.c
svdlib.h
main.h
svdutil.c
las2.c
svdlib.c
svdutil.h
```

The GSL (GNU Scientific Library) library is also required. It is freely available from `https://www.gnu.org/software/gsl//gsl.html`. Depending on the location of the GSL installation on the local machine, C compiler flags -I and -L are likely needed. In the example below, the -I and -L flags point the C compiler to subdirectories of `/opt/local/` where the GSL files reside. These paths should be changed appropriately.

Simple bare-bones commands for compiling `SplitSup` are

```
gcc -I/opt/local/include -c main.c
gcc -I/opt/local/include -c flatBinary.c
gcc main.o flatBinary.o -L/opt/local/lib/ -lgsl -lgslcblas -o SplitSup
```

3.3 Pronunciation and citation

The name `SplitSup` stands for *split support*, and should be pronounced “Split Soup”. (While the name `SplitPsup` was attractive, we did not want to encourage an erroneous interpretation of split scores as p-values.)

If you use `SplitSup` for your research, please cite [AKR16]. This citation will be updated once the paper is accepted.

Bibliography

[AKR16] E. S. Allman, L. S. Kubatko, and J. A. Rhodes. Split scores: a tool to quantify phylogenetic signal in genome scale data. `arXiv.math/*****`, pages 1–14, 2016.

[AR06] E. S. Allman and J. A. Rhodes. The identifiability of tree topology for phylogenetic models, including covarion and mixture models. *J. Comput. Biol.*, 13(5):1101–1113, 2006.

[CK14] J. Chiffman and L. S. Kubatko. Quartet inference from SNP data under the coalescent model. *Bioinformatics*, 30(23):3317–3324, 2014.

[CK15] J. Chifman and L. Kubatko. Identifiability of the unrooted species tree topology under the coalescent model with time-reversible substitution processes, site-specific rate variation, and invariable sites. *Journal of Theoretical Biology*, 374:35 – 47, 2015.