Sliding Windows Analysis Procedure to detect Selective Constraints (SWAPSC)

Version 1.0, April 2004

Mario A. Fares

Suggested citation

(Fares, M. A. (2004) Bioinformatics In press)

The author can be reached at:

Mario A. Fares

Molecular Evolution and BioInformatics Laboratory

Department of Biology

National University of Ireland

Maynooth, Co. Kildare

Ireland

Email: mario.fares@may.ie

Phone: 353 1 7086368.

Fax: 353 1 7083845.

Table of Contents

1. History

SWAPSCv1.0 is the first version of the Sliding Window Analysis Procedure to detect Selective Constraints in protein-coding genes. The software was previously written in PERL (2002). The large amount of information data generated and the exhaustive screening algorithm of selective constraints required the performance of a more friendly and easy-to-run software. The first useful version of the software is then SWAPSCv1.0 for Windows and UNIX. This software include an automatic windowing screening for selective constraints and the Kimura´s method of Li (1993) to compare synonymous and non-synonymous nucleotide estimated substitution rates with those expected under neutrality.

2. INTRODUCTION

SWAPSC is a program to analyse selective constraints in protein-coding genes. This method was applied before to protein-coding sequences of RNA viruses (Fares et al. 2002). At the moment, the program only includes an executable file (is not a package), although the different mathematical operations were divided into functions to ease future manipulation of the code. The originality of the program resides in its mathematical flexibility and stability and the automatic performance of the different steps of the method of Fares et al. (2002). One of the most important features of SWAPSC is the capacity to analyse enormous sequence-alignment files in a reasonable time. The test is highly conservative for detecting significant selective constraints at specific codon regions of a protein alignment in single branches of the tree. Given the fact that branches of the tree and codon regions are examined simultaneously the amount of information generated is enormous.

For accurate and reliable estimation of the parameters, the program requires alignment of sequences no shorter than 200 to 300 amino acid or codon sites. In caseof shorter alignments is recommended the use of a greater number of sequences and a greater number of simulated alignments to estimate synonymous and non-synonymous rates. The performance of the method requires the use of sets of simulated alignments and hence depends on the use of other programs like the Phylogenetic Analysis by Maximum Likelihood (PAML) package v 3.14 (Yang 2000 a).

3. General assumptions and requirements of the program

The program can be run under different conditions, using different sequence alignment sizes or lengths and there is no limitation for the number of sequences in the alignment. Several assumptions however have to be taken into account to have optimised and robust statistical results:

a) nucleotide-sequence alignment of at least 200 to 300 codon sites.
b) A completely resolved phylogenetic tree (completely bifurcated or rooted tree).
c) Binomial distribution of synonymous (Ks) and non-synonymous (Ka) nucleotide substitutions is assumed.
d) At the moment, only the Kimura-based method of Li (1993) is used but further methods will be implemented soon in the next version.

e) The process of nucleotide and amino acid substitutions is stationary. This means that amino acid and nucleotide frequencies have remained constant over time. No molecular clock is assumed or imposed. Authors using the program should put special effort in obtaining completely resolved trees since the final results of selective constraints is dependent on the topology of the tree. Users are hence encouraged to conduct as many tree inference analyses as possible in order to obtain the most resolved tree. If there is any unresolved cluster in the tree, the user should bifurcate the tree manually. The program will infer a distance 0 for the artificial branches introduced in the tree and thus no biased effects will affect the resulted estimates.

4. Model

4.1. Theory

The first step of the method is the estimation of the expected probability of non-synonymous, $P(d_N)$, and synonymous, $P(d_S)$, nucleotide substitution per codon site in each window-sliding step. Since we are going to refer to the Li´s model for the estimate of nucleotide substitution parameters, we are going to use K_S and K_a here on to refer to synonymous and non-synonymous substitutions, respectively. To estimate $P(K_S)$ and $P(K_a)$ let first suppose that the X-th non-synonymous nucleotide substitution and the Y-th synonymous nucleotide substitution are variables with probability $1/L$ to occur in the sequence under study, where L is the total number of codon sites in the sequence. If we assume that non-synonymous and synonymous substitutions are discrete and take values x_i and y_i, respectively, then the expectation of X^r and Y^r are the r^{th} moment about zero of the non-synonymous substitution variable X_N and synonymous change variable Y_S. These moments are defined as

$$\theta_N^r = \mathrm{E}(X_N) = \frac{1}{n}\sum_i (x_i)^r,\ \theta_S^r = \mathrm{E}(Y_S) = \frac{1}{s}\sum_i (y_i)^r \quad (1)$$

Where *n* and *s* are the total number of non-synonymous and synonymous nucleotide sites, respectively.

The first moment, *r* = 1, is the expectation of X_N and Y_S and are the mean of the variable that describes non-synonymous and synonymous nucleotide substitutions on an alignment of sequences, respectively.

θ_S^1 and θ_N^1 are estimated using *K* random sequence alignments simulated by maximum-likelihood under a known phylogenetic tree using the EVOLVER program in the PAML package, v3.14 (Yang 2000 a). This program generates a codon sequence for the root of the tree (inferred from the real data set) and evolves the sequence along the branches of the phylogeny using specified branch lengths and substitution parameters (Yang et al. 2000 b). Simulations are performed using as parameters the branch lengths estimated by the modified Nei and Gojobori´s method (Zhang et al. 1998) as well as codon frequencies estimated from the real data set. Once sequences are simulated, we estimate non-synonymous substitutions per non-synonymous site (K_{aij}) and synonymous substitutions per synonymous site (K_{Sij}), between sequences *i* and its ancestral inferred sequence-*j*, by the unbiased method of Li (1993). The simulations allow us to avoid the effect of the nucleotide compositional bias in the third codon positions on the codon usage, the estimation of K_a and K_S under neutrality and the buffering of the regional codon-composition effect on the estimates of K_a and K_S.

The second step consists on the estimation of the probabilities of nucleotide substitutions in the *K* random sequence alignments following a binomial distribution of each variable compared to the sum of both:

$$P(\theta_N) = \frac{\theta_N}{\theta_N + \theta_S}, \; P(\theta_S) = \frac{\theta_S}{\theta_S + \theta_N} \tag{2}$$

where θ_N and θ_S are the mean number of synonymous and non-synonymous substitutions estimated from the *K* random sequence alignments as:

$$\theta_N = \frac{1}{NK}\sum_{l=1}^{K}\sum_{f=1}^{\mathrm{N}}\left(K_{a_{ij}}\right)_{fl}, \; \theta_S = \frac{1}{NK}\sum_{l=1}^{K}\sum_{f=1}^{\mathrm{N}}\left(K_{S_{ij}}\right)_{fl} \tag{3}$$

Here, *N* stands for the total number of pairwise comparisons between random simulated sequence *i* and its ancestral inferred sequence *j*. K makes reference to the total number of simulated data sets used.

If the sequence alignment is large enough (let say 300 amino acids) we can assume that non-synonymous (K_a) and synonymous (K_S) nucleotide substitutions follow a Poisson distribution with parameters:

$$\lambda_N = nP(\theta_N), \; \lambda_S = sP(\theta_S) \tag{4}$$

Therefore, the probability of observing $K_a = \alpha$ and $K_S = \beta$ nucleotide changes in a specific window *Z* of the real sequence alignment are respectively:

$$P_Z(X_{N_{ij}^Z}, \lambda_N) = e^{-\lambda_{N_{ij}^Z}} \frac{\lambda_{N_{ij}^Z}^{X_{N_{ij}^Z}}}{X_{N_{ij}^Z}!} \text{ and } P_Z(Y_{S_{ij}^Z}, \lambda_S) = e^{-\lambda_{S_{ij}^Z}} \frac{\lambda_{S_{ij}^Z}^{Y_{S_{ij}^Z}}}{Y_{S_{ij}^Z}!} \tag{5}$$

Where $X_{N_{ij}^Z}$ and $Y_{S_{ij}^Z}$ are the observed number of non-synonymous and synonymous nucleotide substitutions between sequences *i* and its ancestral inferred sequence-*j* in the window *Z*, respectively, and are calculated as:

$$X_{N_{ij}^{Z}} = n\alpha \text{ , } Y_{S_{ij}^{Z}} = s\beta \tag{6}$$

This procedure of estimating $P_Z(\alpha,\lambda)$ and $P_Z(\beta,\lambda)$ is repeated in every sliding-window step and the possible action of selection in each region tested.

Once detected those window regions with significantly different number of substitutions than expected, we are interested in the study of the selection intensity, being the estimation of the non-synonymous-to-synonymous rate ratio and its variance good estimators of the intensity of selection acting on a specific codon region.

4.2. Optimising the window size

The sliding window-based method requires the finding of the most appropriate window size to analyse the different regions of the alignment. I have to sought a note of caution when the window size is chosen randomly because there is a strong effect of the codon number and composition of the window-region size used on the results obtained. Furthermore, the use of a specific optimised window size strongly depends on the data under analysis. Thus, assuming the absence of saturation of synonymous sites, there is a threshold of variability upper which highly conserved sequences require smaller window sizes to detect selective constraints than more variable sequences. I conduct an analysis of the appropriate window size to avoid false significant results but getting as much biological information as possible. To do so, I generate random window sizes (δ_i) in each alignment of the *K* random data sets simulated and slid each *i-th* window with size δ_i along the random sequence alignment, estimating for every window sliding step

(l) the average number of non-synonymous ($\overline{K}_{a_l}$) and synonymous ($\overline{K}_{S_l}$) nucleotide substitutions for all pairwise sequence comparisons, given the phylogenetic tree. Thereafter, the probability of having K_{a_l} for each l-th step is calculated using equation 5, obtaining by this way a distribution of probabilities along the sequence alignment with a mean $\overline{K}_a$. Before starting the window sliding procedure I randomise two sequence alignment pieces of size (δ - 1) that are joined later to the beginning and end of the sequence alignment to avoid undercounting the first and last δ - 1 codons in the first and last windows, respectively. Following this procedure, every codon site is counted δ times in all the sliding steps. The generation of random pieces of the sequence alignment is repeated during the (δ - 2) first and last sliding steps, avoiding hence the nucleotide composition effects of the random pieces on the calculations performed in each window step. I then obtain the distribution of probabilities of $\overline{K}_{a_l}$ in the K random alignments for the different window sizes randomly chosen. Finally, I plot the mean $P(K_a)$, and the 5% lowest $P(K_a)$ and 5% highest $P(K_a)$ for each window size against the window size used and choose the largest window having a 5% lower probability > 0.05 as the appropriate window size.

Once the appropriate window size is determined, SWAPSC slides windows of this size along the real data set in the same way as done for the random data sets and calculates the probabilities of the estimated $K_{a_{ijZ}}$ and $K_{S_{ijZ}}$ in each *Z-th* window step to test against chance using equation 5, as described in the previous section.

This method allows to discriminate, by the direct comparison of the expected and observed nucleotide substitutions, between different hypotheses that, in addition to neutral evolution, positive selection or purifying selection, can explain different mutational dynamics, as summarised in Table 1.

As a final complementary step to the method, in those windows with different $K_{a_{ij}^Z}$ or $K_{S_{ij}^Z}$ than expected by chance I measured the type and intensity of selection by estimating the non-synonymous-to-synonymous rate ratio $(\omega_Z = \frac{K_{a_{ij}^Z}}{K_{S_{ij}^Z}})$ for the comparison of sequences *i* and its inferred ancestral-sequence *j*. values of $\omega = 1$, $\omega < 1$ and $\omega > 1$ indicate neutral evolution, purifying selection and positive selection, respectively. However, to avoid obtaining biased values due to saturation of synonymous sites, specially in regions with multiple hits, those values of ω_Z higher than 1 have to be tested for significance. Consequently, the variance of ω [$V(\omega)$] need to be estimated from the data and used to test the significance of ω against neutrality. An estimator of $V(\omega)$ was obtained by means of the Fisher´s delta method (Weir 1996):

$$\hat{V}(\omega) = \frac{1}{d_S^2}\left\{\hat{V}(A_0) + \frac{L_2^2\hat{V}(B_2) + L_0^2\hat{V}(B_0)}{(L_0+L_2)^2} + \omega^2\left[\hat{V}(B_4) + \frac{L_4^2\hat{V}(A_4) + L_2^2\hat{V}(A_2)}{(L_2+L_4)^2}\right] + \frac{L_0}{L_0+L_2}Cov(A_0,B_0) + \frac{\omega L_2^2}{(L_0+L_2)(L_2+L_4)}Cov(A_2,B_2) + \frac{\omega^2 L_4}{L_2+L_4}Cov(A_4,B_4)\right\} \quad \textbf{(7)}$$

Here A_i and B_i are the transition and transvertion rates in the *i*-th degenerated site and their variances [$\hat{V}(A_i), \hat{V}(B_i)$] are given in the equations 3 and 4 in Li (1993), L_i is the number of the *i*-th degenerated sites in the region analyzed and $Cov(A_i, B_i)$ is the covariance of the transition and transvertion rates in the *i*-th degenerated site and is given in equation A8 from Ina (1998). It should be noted that this variance is calculated for ω values along the sequence alignment comparing sequences *i* and its immediate simulated ancestral-sequence *j*, thus there is no need to include covariances in the comparisons.

In some cases, there is no window size fitting the data. In those cases I have realised that window sizes of 2 codons is the minimum size to get enough evolutionary signal (synonymous and non-synonymous nucleotide substitutions). In these cases the program will fix the window size in 2 codons.

Table 1. Representation of the Different hypotheses to explain different values of non-synonymous to synonymous rate ratios ω, K_a and K_S different than expected. Hypotheses 0 to 6 indicate neutrality, positive selection, purifying selection, saturation of synonymous sites, saturation of non-synonymous sites, translational selection, and hot spots, respectively.

	K_a	K_S	**Hypothesis accepted**
$\omega > 1$	>	>	**1**
	>	=	**1**
	>	<	**1, 3, 5**
	=	>	**1, 3**
	=	=	**1**
	=	<	**3, 5**
	<	<	**3, 4**
$\omega = 1$	>	>	**6**
	>	=	**6**
	=	=	**0**
	=	<	**0, 3**
	<	>	**4, 6**
	<	=	**0, 4**
	<	<	**3, 4**
$\omega < 1$	>	>	**6**
	>	=	**2, 6**
	>	<	**3, 6**
	=	>	**2, 6**
	=	=	**2**
	=	<	**2, 4**
	<	>	**4, 6**
	<	=	**2, 4**
	<	<	**3, 4**

5. Running SWAPSC

To run SWAPSC in Windows the user has to double click on the executable icon SWAPSC. If the files provided have the correct information, the console window should show the information stored in the control file, the names for the sequences stored in the output file, the phylogenetic tree, the optimisation process of the window size and the reading process of the different branches of the tree. Note that if the window size is fixed by the user, no optimisation process will be shown.

To run the UNIX version, save SWAPSC files into a folder within your UNIX machine. Uncompress and unpack the SWAPSC version by typing:

- uncompress SWAPSC1.0.tar.Z
- tar xvf SWAPSC1.0.tar

make your file SWAPSC.cpp executable by typing:

- Chmod +x SWAPSC1.0.cpp

The user then has to compile the executable SWAPSC file by typing:

- g++ -fno-for-scope SWAPSC1.0.cpp

This instruction will generate a file called a.out. The generated a.out file has to be renamed by typing:

mv a.out SWAPSC1.0.exe

Then, the program runs by simply typing:

SWAPSCv1.0.exe

In some Linux machines a lot of errors are generated when trying to compile this version of SWAPSC. This errors however can be avoided by adding the following line to the beginning of the code source file:

using namespace std;

preceding the main body of the program in the source code.

6. Control file

The program has no interface. Consequently, the design of the code was done in as much friendly way as possible to avoid introducing much user information. Most of the estimations and operations are therefore conducted within the program and little information has to be introduced by the user. The text file containing control commands is named “SWAPSC.ctl” and has to be in the same folder as the executable file.

The different lines of the control file looks like the following:

```
data_file: My_data.txt          *File with the sequence alignment
Tree file: My_tree.txt          *File with the phylogenetic tree in Newick format
Output file: My_Out.txt         *Name of the file with the output results
Simulations: My_simul.txt       *Name of the file with the simulated alignments
Model: 0                        *0 = Li (1993), 1 = Nei&Gojobori, 2 = Pamilo&Bianchi
Window: 0                       *0 = inferred as in Fares et al. (2002), 1 = fixed
Window_size = 2                 *Codon size if fixed, minimum length 1 codon and
                    maximum length 20 codons
```

The first line of the control file asks for the file name containing the sequence alignment. The sequence alignment has to be in **Phylip sequential** format, and **coding** sense, otherwise an error will be reported in the console screen informing about bad alignment or length. Indels (gaps) are automatically excluded from the alignment by the program in all the subsequent analyses. Thus, it is not required to remove gaps manually.

The sequence alignment must be in Phylip format as shown in Figure 1.

```
4 900
seq_1
ATGGGCGTA........
Seq_2
ATGGCCGGA........
Seq_3
ATGGGAGAG........
Seq_4
ATGGGGGGG.......
```

Figure 1. Sequence alignment of protein-coding sequences in Phylip format.

The first line of the input file stands for the number of species or sequences used (4 in this case) and the nucleotide length of the sequence alignment (900 in this case). Please note that the alignment has to be a protein-coding alignment and therefore the result of dividing the alignment length by 3 has to be an integer number.

The line containing the number of sequences in the alignment and the length of the alignment in nucleotides is followed by the name of the first sequence and the sequence in nucleotides (either in lower or uppercase) in sequential format and in one line. Names for the sequences can be as long as 20 characters and allow for spaces and all possible non-text symbols.

The second line of the control file asks for the file containing the phylogenetic tree in **Newick** format. As an example, imagine that sequences 1 and 4 are more closely related, whereas 3 and 2 form a cluster apart (Figure 2). Also assume that seq 1 is the first sequence in your alignment, seq2 the second and seq4 the last.

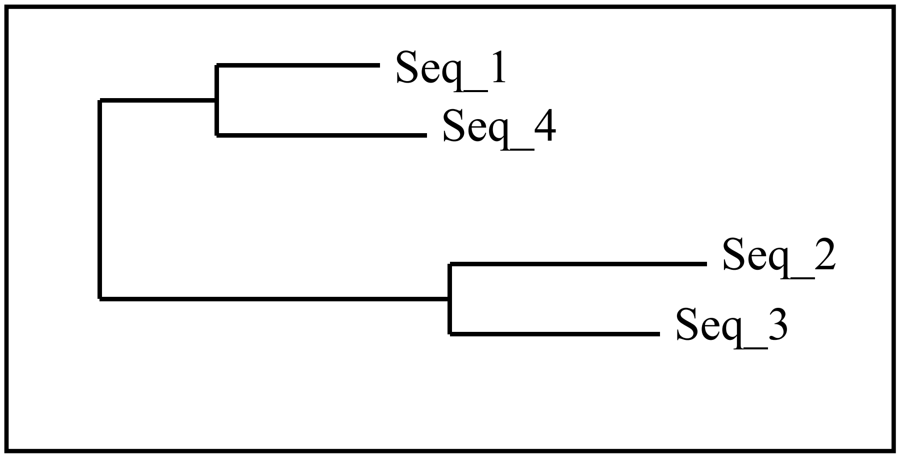


Figure 2. Phylogenetic relationships among the sequences in the alignment of Figure 1.

The file with the tree in newick format looks like the following:

((1,4),(2, 3))

Note that no spaces are allowed between names and colons has to be added to separate sequence numbers.

The third line of the control file asks for the name of the output file. Even though the output file use to be very big, the information is well organised so the user can follow easily each line of the file.

The line in the control file accounting for simulations will ask the user to introduce the name of the file containing the simulated data sets. The user can use as many simulated sets as desired to estimate the main parameters used to obtain the probability of non-synonymous (K_a) and synonymous (K_s) nucleotide substitutions. I would suggest however to use a minimum number of 100 simulated data sets.

The model of nucleotide substitutions can also be specified by the user so that different models can be used. Only the model of Li (1993) is implemented at the moment but other Kimura-based models such as Pamilo and Bianchi (1993) and the Nei and Gojobori´s (Zhang et al. 1998) models are going to be implemented soon.

Finally, the user is asked to either infer the window size following the statistical method of Fares et al.(2002), in which case the option 0 has to be selected, or to fix the window at a specific codon size (option 1). In the latter case, the user has to specify the window size in codons in the line containing the word Window_size. It is worth noting that the optimal window size is recommended (see details in Fares et al. 2002) and that fixing the window size by the user makes sense only in the case of having biological information about specific regions of the protein.

7. Simulated data sets

One of the critical steps to conduct a reliable window-based analysis is to obtain a pseudo-random distribution of synonymous and non-synonymous nucleotide substitutions along the sequence alignment and the phylogenetic tree. This simulated data sets of sequence alignments have to keep the same nucleotide substitution parameters values and the same phylogenetic relationships among the different sequences used as in the input file with the real sequence alignment. Therefore, a phylogenetic tree has to be obtained before starting with the simulation of sequences. I am not going to discuss the different methods of tree reconstruction, but imagine you have inferred a tree by Neighbour-Joining (Saitou and Nei 1987) using gamma-corrected amino acid distances. This tree can be written in newick format and used as an initial tree in the CODEML program from the PAML package (Yang 1997) together with the nucleotide sequence alignment. There are many different models implemented in CODEML program to estimate codon substitution parameters. These models can be divided into those that estimate the intensity of selection acting at single codon sites and those that estimate the intensity of selection at specific lineages of the phylogenetic tree but averaging the value of ω along the alignment. Within the former models, models can be subdivided into discrete models (M0, M1, M2, and M3) and continuous distribution models (M7 and M8). There are two different ways to obtain good estimates of codon substitutions. One way is to consider models that detect single codons under adaptive evolution. In this case, the user might determine which model is better fit to the data (for more details see Yang et al. 1998, 2000 b). Alternatively, the free ratio model (that assumes independent ω values for the different branches of the phylogenetic trees) can be used to estimate the codon-substitution parameters. None of both models can be reliably associated with the SWAPSC, since SWAPSC tries to detect selective constraints at single amino acid sites or group of codons and single lineages of the tree at the same time. The reliability of the models in PAML are, however, good enough as to infer simulated data sets to be used in SWAPSC. I would advise users to use the free-ratio model whenever possible to estimate the parameters to conduct the simulations since selective constraints are determined for each branch of the tree independently. In the case of big alignments (100 sequences or 1000 sequences), running the free-ratio model is prohibitively slow and discrete models would be good enough to obtain the simulated data sets.

Once simulated data set obtained, the format of this file to be used by SWAPSC has to be as shown in Figure 3.

```
2
4 900
seq_1
ATGGGCGGC......
Seq_2
ATGGGAGGG......
seq_3
ATGGGCGGC......
Seq_4
ATGGGAGGG......
4 900
seq_1
CCCAACGTC......
Seq_2
CCCAACGTC......
seq_3
CCCAACGTC......
Seq_4
CCCAACGTC......
```

Figure 3. An example of the simulated sequence file format to be read by SWAPSC program.

Where the first line specifies how many simulated data sets are being used, being in this case 2 data sets. Then we have the alignment of each set of sequences in Phylip format. The user can use as many simulated data sets as needed to get statistically optimum window sizes. When the variability between sequences is large and significant results would be obtained even using 1 codon size, the program sets the window size at 2 codons to get as much information as possible for the statistical analyses.

There is a perl script in our webpage available to use in UNIX to put the EVOLVER simulation output file into the format suitable for SWAPSC. This script has been written by Mr. Valentin Ruano in our lab and is called swapsc-in. To run the script the user should make it first executable by typing:

- chmod +x swapsc-in

then, to run the script it is enough with typing:

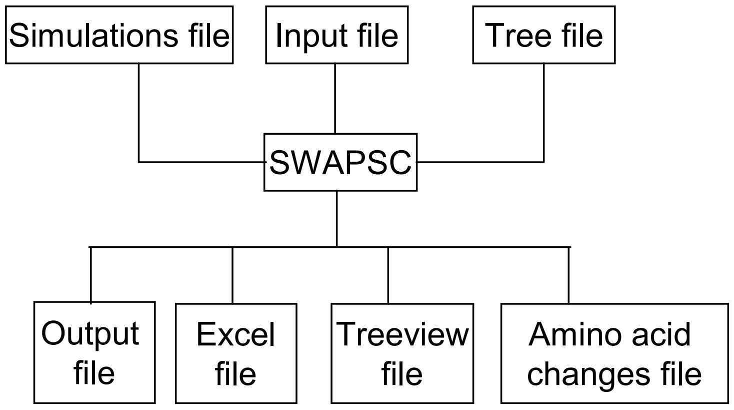
- swapsc-in <evolver_out>swapsc-in_out

here evolver_out is the name of the file resulted from evolver simulations whereas swapsc-in_out is the output of swapsc-in that contains the simulated sequence alignments in the appropriate format for SWAPSC, both file names are freely specified by the user.

8. Output file

Once the files required (Figure 4A) are provided, the program runs by simply double-clicking on the executable icon in Windows or by typing SWAPSC1.0 in LINUX or UNIX. There will be a main output file generated with the significant information and several additional output files meant to help the user to manage the amount of generated data (Figure 4B).

A



B

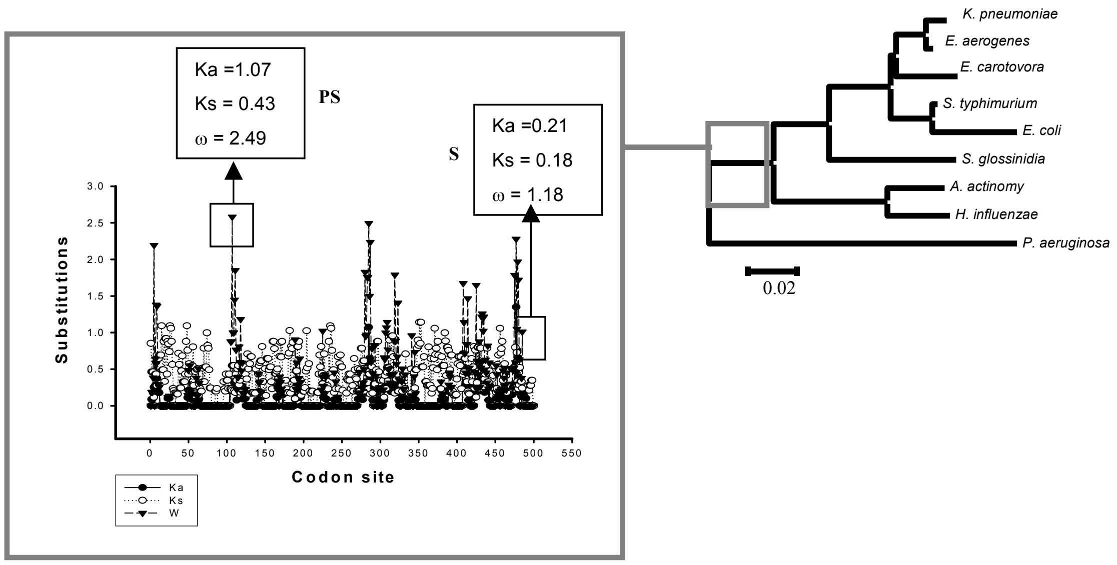


Figure 4. Flow of SWAPSC. A) files required and files generated by SWAPSC. B) an example of the output information that can be obtained after running

SWAPSC. The graph shows an example of the selective constraints acting on a specific branch of the phylogenetic tree.

The output file shows the following information:

- The number of sequences in the alignment as well as the length of the alignment in nucleotides.

e) The input sequence alignment.

f) The user phylogenetic tree with specification of the numbers for the internal nodes of the tree.

g) Ancestral sequences inferred by maximum parsimony.

h) Parameter estimates from the simulated data sets, which includes:

h.1) The mean number of non-synonymous nucleotide substitutions per non-synonymous sites (Ka) and synonymous nucleotide substitutions per synonymous sites (Ks). These numbers are estimated by the specified model in the control file.

h.2) The probabilities of Ka and Ks calculated from the simulated data sets under a binomial distribution of changes.

h.3) The average ω ($\omega = \frac{K_a}{K_s}$) and its variance calculated analytically following Fares et al. (2002).

h.4) The transition-to-transversion rates ratio ($\kappa = \frac{T_S}{T_V}$).

h.5) The optimum window size also estimated from the simulated data sets.

h.6) The 5% lower probability associated to the optimum window size, which confirms that no significant results should be detected by chance in the real sequence alignment data set.

i) A table with the essential results to detect selective constraints in the user sequence alignment. This table contains the reliable information for further analyses if required by the user. Only nucleotide regions with significant results (showing selective constraints different from the expectation) are shown in the table to avoid unmanageable output file. This information details:

i.1) The branch connecting the nodes.

i.2) Nucleotide region in that branch where selective constraints are detected.

i.3) The estimated number of non-synonymous substitutions for the codons in that nucleotide region (K_a).

i.4) The probability of K_a under a Poisson distribution in that window region.

i.5) The estimated number of synonymous substitutions K_s in that window region.

i.6) The probability of K_s calculated under a Poisson distribution.

i.7) The non-synonymous-to-synonymous rates ratio (ω).

i.8) The probability that ω in that region of the alignment is different from the mean ω value for the alignment. This probability is estimated assuming a normal distribution of ω values along the alignment.

i.9) The type of selective constraints acting on that nucleotide region. These selective constraints are classified as:

PS: indicating adaptive evolution or positive selection. Only regions where the estimated number of non-synonymous nucleotide substitutions is greater than expected by chance and where ω is significantly greater than 1.

NS: indicating negative selection or purifying selection. These are regions where the number of non-synonymous nucleotide substitutions are smaller than expected or where ω is significantly smaller than the mean ω estimated for the alignment.

AdN: indicating accelerated rates of non-synonymous nucleotide substitutions. These are regions where the estimated number of non-synonymous substitutions are greater than expected but where either ω is smaller than 1 or ω is greater than 1 but the regions presented statistical evidence of saturation of synonymous sites or alternatively the number of synonymous nucleotide substitutions is 0. Consequently, no conclusions can be reached regarding the existence or not of adaptive evolution.

S: indicating saturation of synonymous sites. These regions are those where the number of synonymous nucleotide substitutions is significantly smaller than expected.

HS: indicating regions where the number of synonymous and non-synonymous nucleotide substitutions are greater than expected under neutrality and hence hot spots.

j) A summary table with the information about the percentage of codon sites under the different selective constraints. The mean K_a, K_s and ω values is provided.

An example of the information obtained running SWAPSC is depicted in Figure 4B.

9. Additional generated output files

The program SWAPSC generates three more files that might be very useful for phylogenetic and evolutionary analyses. A file containing all the amino acid changes and their position in the different branches of the tree is generated (aa_Changes.txt). This file shows all the different types of amino acid substitutions in regions where selective constraints have been detected. It also shows if amino acid changes occur in overlapping regions, being important to collapse these regions to avoid over-estimation of regions under selective constraints. I used the one code-based letter to codify amino acid residues, and the arrow shows the sense of the substitution from the internal node to the more recent node or sequence. Numbers between brackets account for the amino acid position in the sequence alignment, including gaps.

On the other hand, the program also generates an Excel (.csv) table showing the complete information for each region of the sequence alignment and branch of the tree. This table includes the detailed information of the main Table of the output file. This file will easy the filtering of the information or the use of the complete information to test the distribution of selective constraints in the sequence alignment.

Finally, a Treeview-based file (Tree_nodes) is generated in the Window version of SWAPSC to enable the user to localise quickly the branches under specific selective constraints. These file does not contain a phylogenetic tree but rather the topology of the tree with the numbers of sequences of internal nodes labelled. Therefore, no distances or bootstrap values are shown in this tree but rather sequences in the tip of the tree (the name of the sequence followed by the number of that sequence in the input file) and the ancestral inferred sequences in the internal nodes are shown (Figure 5).

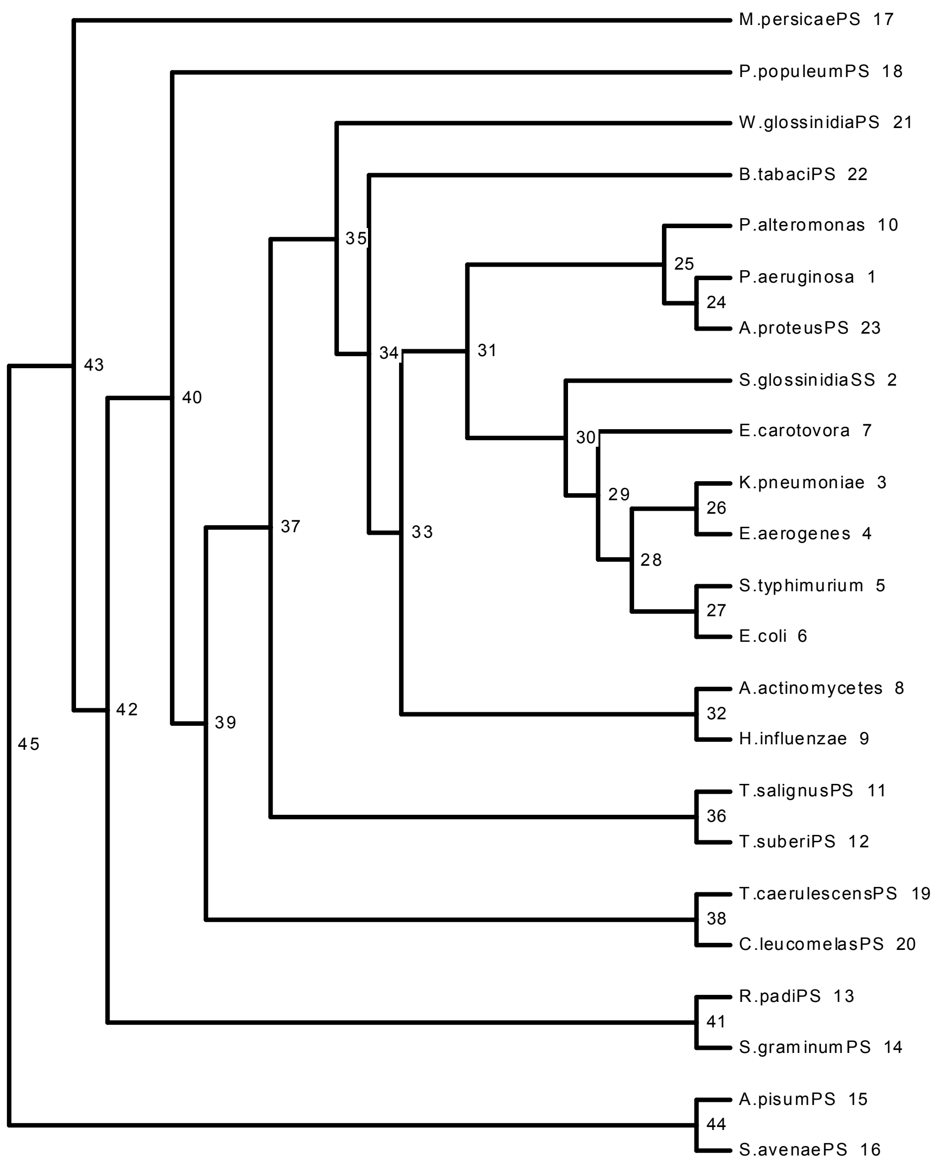


Figure 5. Tree generated by SWAPSC1.0 for a set of 23 sequences coding for the heat-shock protein GroEL. In the tips of the tree the sequence names followed by their order number in the input file are shown. Numbers in the internal nodes are the sequence numbers of ancestral inferred sequences used in the estimation of selective constraints.

10. Computer time consumption running SWAPSC

SWAPSC is an optimised software that uses Kimura-based models to detect selective constraints. The time of running the program is acceptable and increases depending on the input files sizes. The computation time of the main parameters of synonymous and non-synonymous nucleotide substitutions increases linearly with the number of simulated data sets. However, the computation time is acceptable even using 1000 simulated data sets, specially when the number of sequences in the alignment is below 200. The time however will not be as large as to desperate the user even using 1000 sequences with 1000 simulated data sets.

The optimisation procedure time of the window sizes seems to increase exponentially with the number of sequences in the sequence alignment. The program has been tried with 300 sequences using 100 simulated data sets and it took no more than 5 minutes to perform all the analyses required. If the window size is selected in SWAPSC.ctl, the computation time does not take more than few seconds. The highest computation time amount is therefore required by the optimisation procedure of the window size.

11. References

Fares, M. A. Elena, S. F., Ortiz, J., Moya, A., and Barrio, E. (2002). A sliding window-based method to detect selective constraints in protein-coding genes and its application to RNA viruses. J. Mol. Evol. **55**: 509-521.

Ina, Y. (1998). Estimation of the transition/transversion ratio. J. Mol. Evol. **46**: 521-533.

Li, W. H. (1993). Unbiased estimation of the rates of synonymous and nonsynonymous substitutions. J. Mol. Evol. **36**: 96-99.

Saitou, N. and Nei, M. (1987). The Neighbor-Joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**: 406-425.

Yang, Z. (1998). Likelihood ratios test for detecting positive selection and application to primate lysozime evolution. Mol. Biol. Evol. **18**: 7584-7589.

Yang, Z. (2000). Phylogenetic análisis by maximum likelihood (PAML). Version 3. University College London. London.

Yang, Z., Nielsen, R., Goldman, N., and Pedersen, A. M. (2000). Codon-substitution models for heterogenous selection pressures at amino acid sites. Genetics **155**: 431-449.

Zhang, J., Rosemberg, H. F. and Nei, M. (1998). Positive Darwinian selection after gene duplication in primate ribonuclease genes. Proc. Natl. Acad. Sci. USA **95**: 3708-3713.