Inference of Episodic Changes in Natural Selection Acting on Protein Coding Sequences via CODEML

Joseph P. Bielawski,[1,2] Jennifer L. Baker,[3] and Joseph Mingrone[2]

[1]Department of Biology, Dalhousie University, Halifax, Nova Scotia, Canada
[2]Department of Mathematics & Statistics, Dalhousie University, Halifax, Nova Scotia, Canada
[3]Center for Research on Genomics and Global Health, National Human Genome Research Institute, National Institutes of Health, Bethesda, Maryland

This unit provides protocols for using the CODEML program from the PAML package to make inferences about episodic natural selection in protein-coding sequences. The protocols cover inference tasks such as maximum likelihood estimation of selection intensity, testing the hypothesis of episodic positive selection, and identifying sites with a history of episodic evolution. We provide protocols for using the rich set of models implemented in CODEML to assess robustness, and for using bootstrapping to assess if the requirements for reliable statistical inference have been met. An example dataset is used to illustrate how the protocols are used with real protein-coding sequences. The workflow of this design, through automation, is readily extendable to a larger-scale evolutionary survey.

Keywords: codon model • natural selection • episodic evolution • maximum likelihood • d_N/d_S ratio • experimental design

How to cite this article:
Bielawski, J.P., Baker, J.L. and Mingrone, J. 2016. Inference of episodic changes in natural selection acting on protein coding sequences via CODEML. *Curr. Protoc. Bioinform.* 54:6.15.1-6.15.32.
doi: 10.1002/cpbi.2

INTRODUCTION

Understanding the biological significance of genetic and genomic variation requires an understanding of the evolutionary processes that are responsible for its origin and persistence. For example, many changes are expected to have little or no effect on protein function, and a process of neutral drift best explains those changes. Alternatively, other DNA changes will have had a positive effect on a protein's function, and will have been subject to Darwinian selection. A sample of protein-coding DNA sequences permits an investigation of these evolutionary processes, thereby providing a means of discovering the functionally important substitutions that have occurred within the encoded protein. One of the most powerful approaches is to use rigorous statistical techniques to model evolution at the level of the codon. Codon models permit separate estimation of both the synonymous (no change in the encoded amino acid) and nonsynonymous (change the encoded amino acid) rates, with the former serving as an estimate of the neutral rate of evolution for the encoded protein. An episode of functional divergence can then be identified as a period when the nonsynonymous rate was significantly greater than the neutral rate.

Current Protocols in Bioinformatics 6.15.1-6.15.32, June 2016
Published online June 2016 in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpbi.2

Markov models for codon evolution specify a specific parameterization of the process of substitution from one codon state to another within a protein-coding gene. These models offer many advantages, but foremost among them is explicit parameterization of the nonsynonymous to synonymous (neutral) rate ratio ($\omega = d_N/d_S$). Estimates of this ratio for a protein-coding gene provide valuable measures of the intensity of natural selection acting at the level of its protein product (Yang and Bielawski, 2000; Anisimova and Liberles, 2012). When purifying (negative) selection dominates the evolution of amino acids, selection will prevent fixation of deleterious nonsynonymous mutations, leading to $\omega < 1$. If nonsynonymous substitutions were free from selection (i.e., neutrality at the level of the protein product) the rate of nonsynonymous substitution would be equal to the rate of synonymous substitution, with $\omega \approx 1$. In the rare case that positive selection promoted the fixation of a series of beneficial substitutions (e.g., movement of a population to a new fitness peak in response to a change in environment or emergence of a new enzyme activity), the nonsynonymous rate can exceed the synonymous rate, leading to $\omega > 1$. Because the latter scenario represents an episodic (short-lived) change in the intensity of selection pressure, codon models that average ω over long periods of evolutionary history have low power to detect such cases of adaptive evolution (Yang and Dos Reis, 2011). However, episodic changes can be detected using codon models that permit the intensity of selection pressure to vary both over sites and evolutionary history (e.g., Yang and Nielsen, 2002; Zhang et al., 2005).

Phylogenetic Analysis by Maximum Likelihood (PAML) is a package of individual programs for phylogeny-based analysis of the process of molecular evolution (Yang, 2007), with one program, CODEML, for detecting purifying ($\omega < 1$) or positive selection ($\omega > 1$) at sites within a set of related proteins (site selection) or along branches of an evolutionary tree (e.g., speciation events). The strength of the package is its rich set of evolutionary models. The models serve as the basis of parameter estimation via maximum likelihood (ML), a wide variety of hypothesis tests, ancestral state reconstruction, estimation of divergence times, detection of positive Darwinian selection at the molecular level, and many other analyses. Models are available for DNA-, codon- and protein-level data, and the codon models implemented in the program CODEML permit the intensity of selection to vary over macro-evolutionary time (branch-models; e.g., Yang, 1998), over sites within a gene (sites-models; e.g., Yang et al., 2000), and over both sites and time (branch-site and clade-site models; e.g., Bielawski and Yang, 2004; Zhang et al., 2005). Branch-site Model A (Zhang et al., 2005) will be employed exclusively within this unit, as it is a model for episodic changes in selection intensity and it serves as the basis of a formal likelihood ratio test (LRT) for an episode of positive Darwinian selection (Zhang et al., 2005; Yang and Dos Reis, 2011).

This unit describes the use of branch-site codon models, as implemented in CODEML, to infer when the intensity of positive (or negative) selection has differed from the average across the tree. Branch-site codon models formalize several categories of the evolutionary process (see site classes in Fig. 6.15.1A), but permit uncertainty about which sites have experienced a particular process by treating them as the categories of a statistical distribution (Fig. 6.15.1B). The site classes fall into two broad evolutionary regimes: constant and episodic selection (Fig. 6.15.1A). The constant regime comprises two site classes, where the same ω parameter is used within a site class for all branches of a phylogeny. The intensity of selection is permitted to differ among those classes by using a different ω for each one (ω_0 and ω_1). Within the episodic regime, sites will experience unique selection intensity along a particular branch, or branches, of a tree. Such branches are referred to as "foreground" (FG) branches, and they employ an independent parameter for the selection intensity unique to the FG (ω_{FG}). Note that selection intensity for all other branches is modeled using either ω_0 or ω_1; thus, those two parameters are shared among the constant and episodic regimes. Branch-site Model A is obtained by taking the

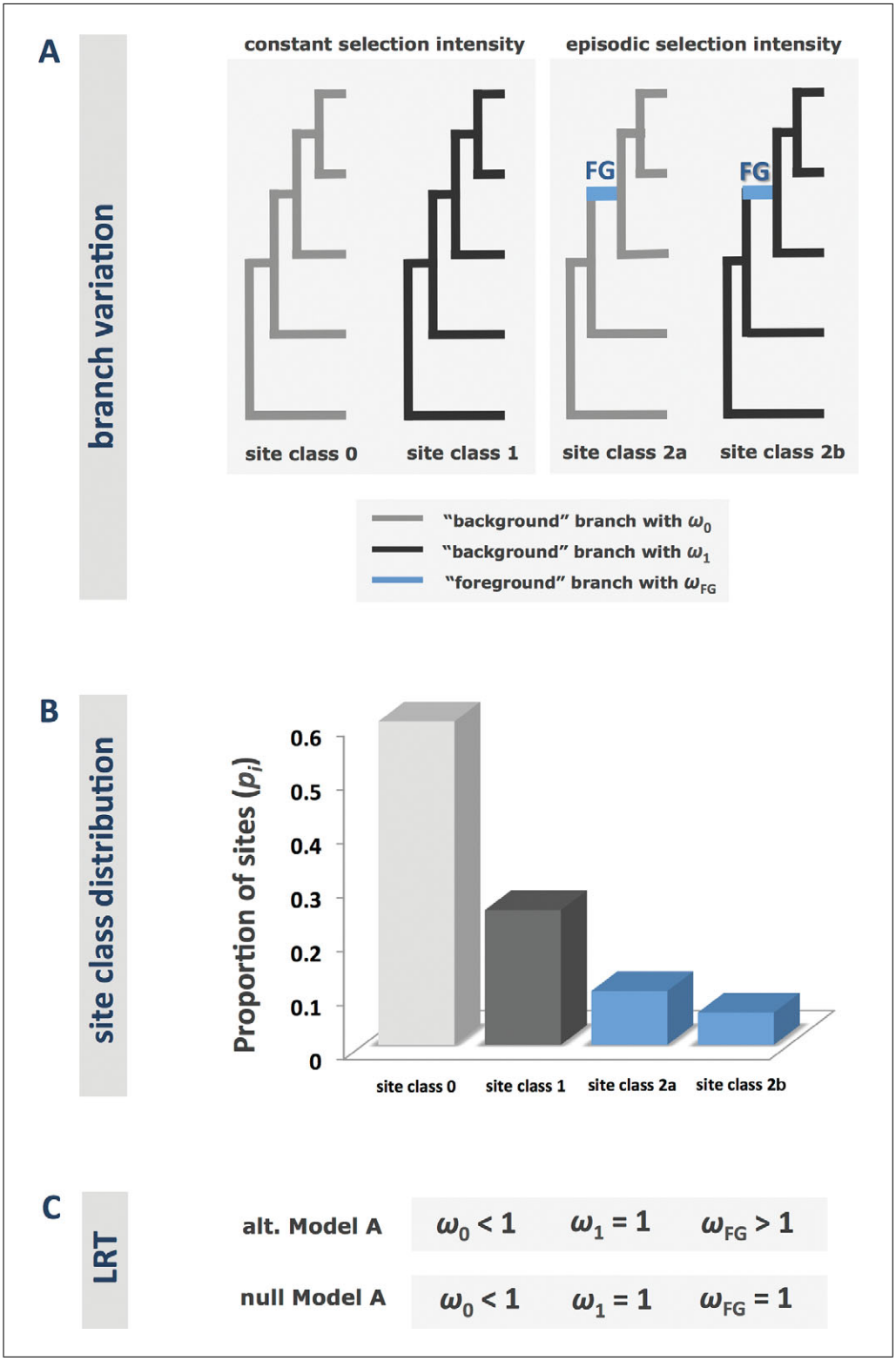


Figure 6.15.1 *(legend appears on next page)*

ω parameters defined in Figure 6.15.1A and imposing some restrictions on them (see two forms of Model A in Fig. 6.15.1C). For the site classes within the constant regime, one is restricted to purifying selection ($\omega_0 < 1$) and the other is set to neutral evolution ($\omega_1 = 1$). For the two site classes of the episodic regime, one specifies episodic evolution (ω_{FG}) under a background of purifying selection ($\omega_0 < 1$) and the other specifies episodic evolution (ω_{FG}) under a background of selectively neutral evolution ($\omega_1 = 1$).

Branch-site models require the user to specify in which branches the episodic changes in selection pressure have occurred; consequently, among-branch ω variation is a fixed effect within Model A. This is in contrast to among-site ω variability, which is treated as a random effect (Fig. 6.15.1B). The model is therefore ideally suited to testing hypotheses about altered selection during any biological event that can be specified as a unique branch, or set of branches (e.g., a gene duplication, lateral gene transfer, or niche colonization event). To carry out an LRT for episodic positive selection, the user specifies the biological event as the foreground (FG) branch in Model A, and fits the data to two forms of this model (Fig. 6.15.1C). The null hypothesis (no positive selection along the FG branch) is specified by setting $\omega_{FG} = 1$. The alternative hypothesis (positive selection along the FG branch) is specified by permitting ω_{FG} to have a value >1, with the specific value of ω_{FG} estimated from the data via maximum likelihood. The null hypothesis is rejected when the likelihood of the data is significantly higher under the alternative form of Model A (Zhang et al., 2005; Yang and Dos Reis, 2011).

In this unit we present three basic protocols and five support protocols, which are intended to fit within a broader experimental design (Fig. 6.15.2). Although this design is focused on inferring episodic changes in natural selection using codon Model A, it is generally applicable to any codon-model based investigation of natural selection. Substituting another branch-site (Yang and Nielsen, 2002; Zhang et al., 2005), or clade-site model (Bielawski and Yang, 2004; Zhang et al., 2005) within the design is straightforward. Moreover, it is also easy to add further quality control and reliability analyses depending on the particular needs of the research question or type of data. The workflow of this design, through automation, is readily extendable to a larger scale evolutionary survey. We employ the *Ceacam* gene from primates as an example dataset for each of the basic protocols. *Ceacam* is well known for its expression on activated T cells and also for its involvement in T cell inhibition (Gray-Owen and Blumberg, 2006; Chen et al., 2012). The primate dataset is well sampled relative to the requirements of branch-site codon models. As the purpose of Basic Protocol 2 is to identify problematic cases, we also provide a second dataset, primate *NR1D1*, in which the requirements for inference have not been met (Baker et al., 2016). *NR1D1* is a member of the nuclear receptor superfamily that modulates the expression of core clock proteins (Bugge et al., 2012), and has been shown to act as a regulator of metabolic genes (Zhang et al., 2015). Using Basic

Figure 6.15.1 *(image appears on previous page)* (**A**). Graphical representation of how the ω parameters of a branch-site codon model differ among branches of a phylogenetic tree and among different site classes. Two site classes (0 and 1) specify constant selection intensity. This is achieved by specifying the same ω parameters for all branches of the tree within a given site class. Sites are permitted to have different levels of selection intensity through site classes that employ different ω parameters (ω_0 and ω_1). Two site classes (2a and 2b) specify episodic changes in selection intensity. This is achieved by specifying a unique ω parameter for a branch, or branches, where the intensity of selection changed. This branch is referred to as the "foreground" branch (FG), and the selection intensity at such a branch is modeled via the ω_{FG} parameter. The FG branch is specified by the user, and is treated as a fixed effect within the model. Note that selection intensity for all other branches is modeled using ω_0 and ω_1; thus, these two parameters are shared among all four of the site classes of the model. (**B**). The unconstrained discrete distribution for the four site classes of this branch-site codon model. The model assumes that the evolutionary process varies among sites, but does not assume that the true process is known for each site. To accommodate this uncertainty, variability among sites is treated as a random effect and modeled according to this distribution. The p_i parameters of the distribution specify the probability that a site within the data evolved under each of the four site classes. The values of these parameters are estimated from a given dataset via maximum likelihood. (**C)** Model A is obtained by placing restriction on the values that the ω parameters can take. All versions of Model A restrict site class 0 to purifying selection ($\omega_0 < 1$) and set site class 1 to neutral evolution ($\omega_1 = 1$). Model A serves as the basis for a LRT, and therefore has two forms: null and alternative. In the null form, ω_{FG} is fixed to 1 (no positive section). In the alternative form, ω_{FG} is permitted to be >1. As these models are nested, they comprise a likelihood ratio test for an episode of positive selection along the foreground branch.

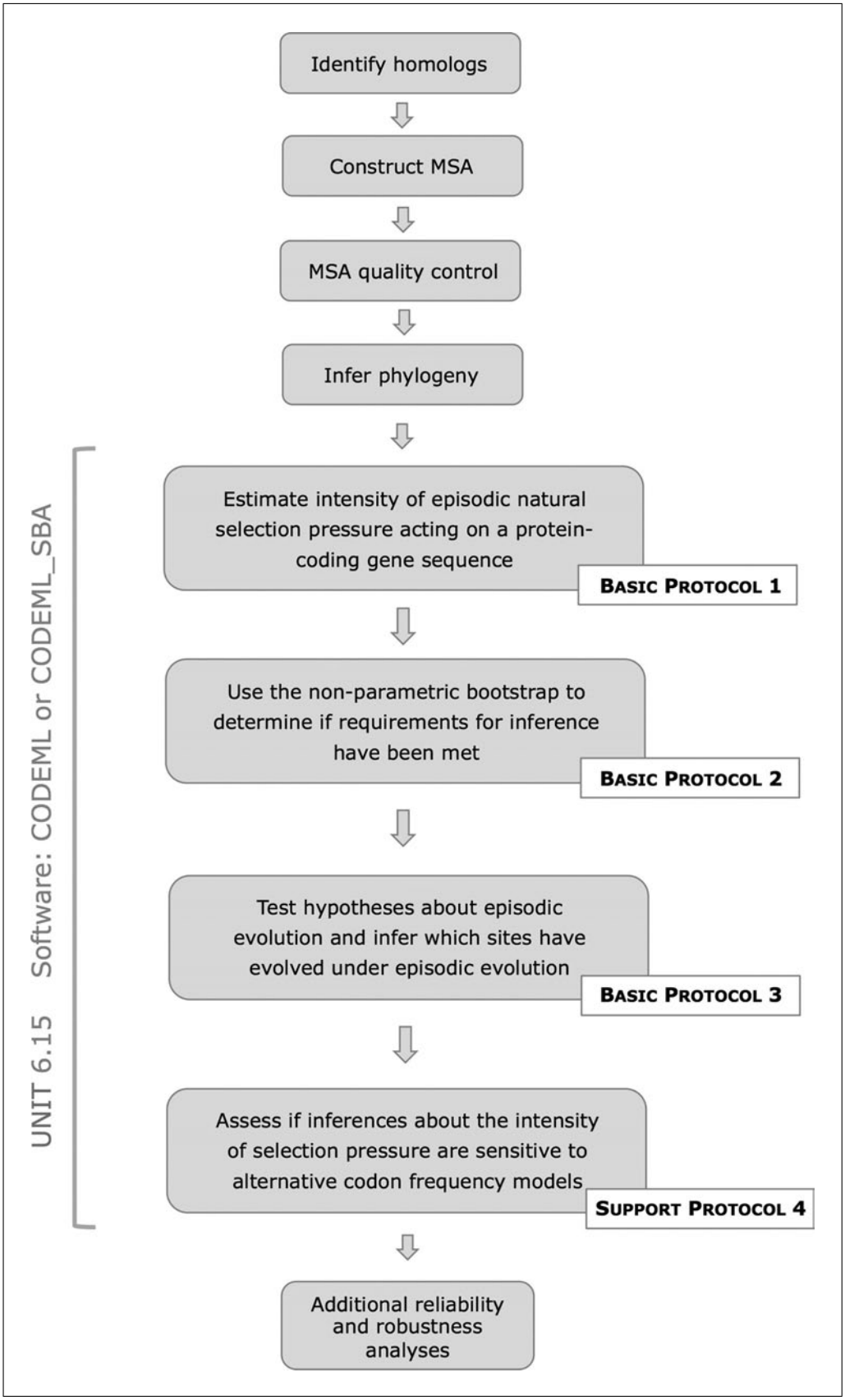


Figure 6.15.2 *(legend appears on next page)*

Protocol 2 to contrast these two genes will provide users of codon models with valuable experience in assessing the reliability of results obtained from any kind of complex codon models.

BASIC PROTOCOL 1

MAXIMUM LIKELIHOOD ESTIMATION OF EPISODIC SELECTION INTENSITY

The first step in the inference process is to fit a codon model to the dataset for the purpose of estimating its ω distribution. Here, because the target of inference is the presence of sites where an episode of altered selection occurred, this protocol is based on a branch-site codon model. Further, branch-site codon Model A (Fig. 6.15.1) is used because it will serve as the basis for an explicit likelihood ratio test for positive selection in Basic Protocol 3. The example dataset for this protocol is a set of 11 primate *Ceacam* sequences. *Ceacam* encodes a cell adhesion protein that plays a role in both tissue differentiation and modulation of immune responses. The sequence divergence represented by the *Ceacam* dataset is appropriate to codon models. The suitability of *Ceacam* is evaluated in Basic Protocol 2, and further discussed within the section that presents guidelines for interpreting the results.

Necessary Resources

Hardware

CODEML or CODEML_SBA can be compiled to run on computers running Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems.

Software

Users can use either the CODEML program of the PAML software package, or the variant that permits bootstrapping, called CODEML_SBA. The results will be equivalent. See Support Protocols 1 and 2 for further details about how to download and install these programs.

Files

Fitting a model to a dataset requires a minimum of three files: (i) a tree file, (ii) a sequence alignment file in standard PAML format, and (iii) a program configuration file (referred to as a control file). The tree file and sequence file for the example dataset are provided on-line at *http://www.currentprotocols.com/protocol/BI0615*.

EvoWorks: *https://bitbucket.org/EvoWorks/protocol-inference-of-episodic-selection/downloads*

1. Obtain and install CODEML or CODEML_SBA (see Support Protocols 1 and 2)

2. Create an analytical directory to hold both the input and result files for a single run.

   It is good practice to use directories to organize and document your results. Having a separate directory for each analysis (hereafter "analytical directory") avoids accidently overwriting your results files. Keeping all input and results together provides a convenient way to ensure reproducibility, as existing input files can be re-run to reproduce results.

Figure 6.15.2 *(image appears on previous page)* Overview of an experimental design for computational inference of episodic changes in the intensity of natural selection. The workflow illustrates that many stages of data processing, quality control, and analysis are required to obtain a reliable outcome. The first four stages of the workflow are concerned with rigorous assessment of sequence and alignment quality, and the inference of a phylogenetic tree for those sequences. Although those stages are not covered in this unit, other units are relevant to those tasks. Basic Protocols 1 to 3 and Support Protocol 4 of this unit cover the next four stages of analyses. The last stage is intended to represent a suite of robustness analyses, and suggestions are provided in the main text for how to use alternative software packages to assess the robustness of the results.

Keeping a copy of the control file is essential, as it serves as a direct record of how the software was configured for each analysis.

3. Obtain the sequence alignment file and tree file required for the desired analysis, and copy them to your analytical directory.

   Users will need to prepare their own sequence alignments (e.g., UNIT 3.8; Notredame, 2010) and infer their own phylogenetic trees [see UNIT 6.1 (Page, 2003), UNIT 6.3 (Desper and Gascuel, 2006), UNIT 6.4 (Wilgenbusch and Swofford, 2003), and UNIT 6.6 (Schmidt and von Haeseler, 2007)] for future analyses. The example dataset files for this protocol are:

   Alignment file: `Ceacam_seq.txt`

   Tree file: `Ceacam_tree.txt`

   These files are available at http://www.currentprotocols.com/protocol/BI0615. Open and familiarize yourself with the contents of both files. This is always good practice, as it allows you to visually confirm the contents of each file within an analytical directory. Users should consult the PAML manual (`pamlDOC.pdf`*) from the PAML distribution for additional information about acceptable formats.*

4. Label a branch, or branches, to be treated as foreground in branch-site Model A.

   Branch-site codon models require a user to specify exactly which branches should have a unique ω parameter in the model (i.e., foreground branches).

   A tree file is used to specify both the tree topology and foreground branches for the model. The tree is represented within that file using the "Newick format" (Felsenstein, 2004). Typically, the tree topology will be obtained from a phylogenetic analysis of the data in hand (e.g., see UNIT 6.6; Schmidt and von Haeseler, 2007). To label a foreground branch, the user must annotate the Newick-formatted tree. The annotation format is described in the PAML manual. A method of labeling the branches of a Newick tree is provided in Support Protocol 3.

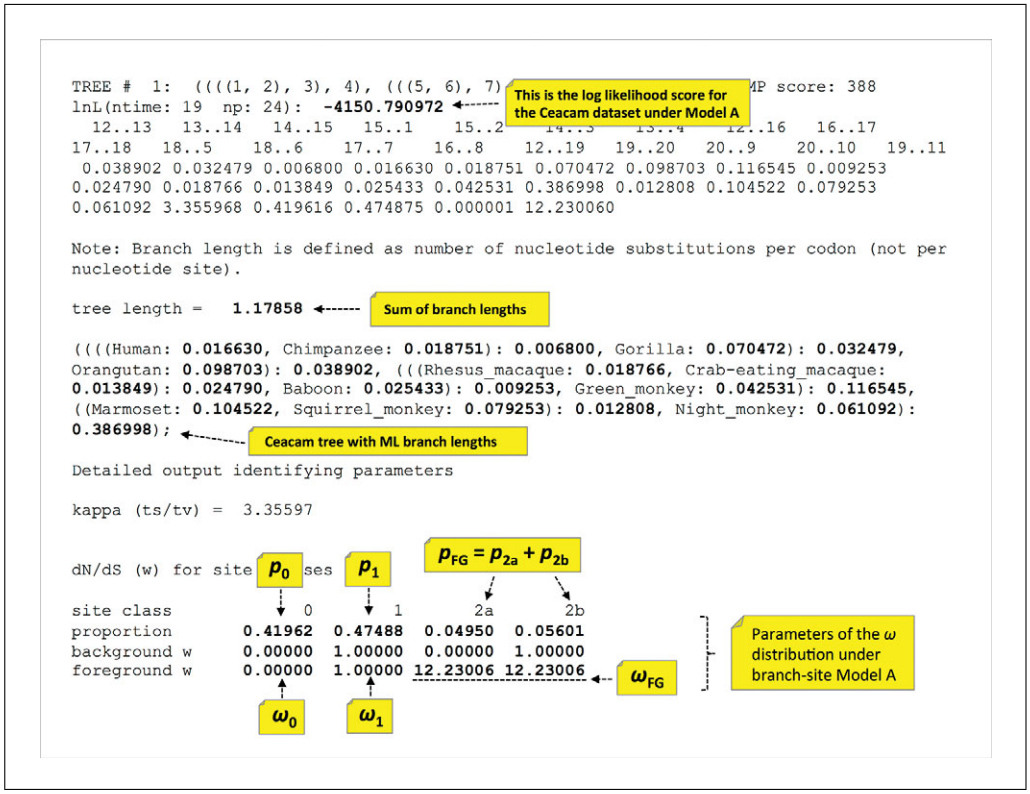
   The tree file provided for the example dataset (`Ceacam_tree.txt`*) is pre-labeled at nodes 1 and 16 (as numbered by the program DENDROCYPHER). As this is a rooted tree, these labels define the branch that separates the Old World and New World monkeys. With this labeled tree, Model A can be used to investigate the hypothesis that some sites in the Ceacam gene evolved by positive Darwinian selection during the divergence of the Old World and New World monkeys.*

5. Make a copy of the default configuration file (`codeml.ctl`) from the PAML distribution and place the copy within your analytical directory.

   Open this file and familiarize yourself with its contents. Users should consult the PAML manual for detailed descriptions of all program variables.

6. Open `codeml.ctl` in a plain text editor and edit the following lines within this file:

```
seqfile      = Ceacam_seq.txt        * sequence data filename
treefile     = Ceacam_tree.txt       * tree structure file name
outfile      = Ceacam_ModelA.out     * main result file name
seqtype      = 1                     * 1:codons; 2:AAs; 3:codons-->AAs
model        = 2                     * models for codons ...
NSsites      = 2                     * 0:one w;1:neutral;2:selection...
cleandata    = 1                     * remove sites with ambiguity ...
```




```
TREE #  1:  ((((1, 2), 3), 4), (((5, 6), 7)                    MP score: 388
lnL(ntime: 19  np: 24):  -4150.790972
  12..13   13..14   14..15   15..1    15..2    14..3    13..4    12..16   16..17
17..18   18..5    18..6    17..7    16..8    12..19   19..20   20..9    20..10   19..11
 0.038902 0.032479 0.006800 0.016630 0.018751 0.070472 0.098703 0.116545 0.009253
0.024790 0.018766 0.013849 0.025433 0.042531 0.386998 0.012808 0.104522 0.079253
0.061092 3.355968 0.419616 0.474875 0.000001 12.230060

Note: Branch length is defined as number of nucleotide substitutions per codon (not per
nucleotide site).

tree length =   1.17858

((((Human: 0.016630, Chimpanzee: 0.018751): 0.006800, Gorilla: 0.070472): 0.032479,
Orangutan: 0.098703): 0.038902, (((Rhesus_macaque: 0.018766, Crab-eating_macaque:
0.013849): 0.024790, Baboon: 0.025433): 0.009253, Green_monkey: 0.042531): 0.116545,
((Marmoset: 0.104522, Squirrel_monkey: 0.079253): 0.012808, Night_monkey: 0.061092):
0.386998);

Detailed output identifying parameters

kappa (ts/tv) =  3.35597

dN/dS (w) for site  ses

site class             0        1       2a       2b
proportion       0.41962  0.47488  0.04950  0.05601
background w     0.00000  1.00000  0.00000  1.00000
foreground w     0.00000  1.00000 12.23006 12.23006
```

Figure 6.15.3 Annotated portion of the main output of CODEML for Model A. The output is abridged, with a considerable amount removed to improve the presentation. The output that has been excluded can be valuable, and users of CODEML are encouraged to explore all the output. The abridged output has been annotated to help users to locate the likelihood score for the data, the maximum likelihood estimate of the branch lengths, and the parameters of the ω distribution for Model A.

Many other types of codon models can be specified via this file, and users should consult the PAML manual for those details.

An asterisk (*) is used to indicate a comment, with any subsequent text ignored. Placing an asterisk at the start of a line causes the program to completely skip the configuration variable on that line.

7. Configure other aspects of the CODEML run within the control file.

Some of the variables within the control file are usually safe to ignore. However, others that are routinely altered are described below.

The CODEML program is able to accommodate any of the alternative genetic codes represented by the code tables within GENBANK. For example, using `icode=0` *specifies the so-called universal genetic code.*

For model A, setting `fix_omega=0` *ensures that the parameters of the ω distribution will be estimated via maximum likelihood. Further, setting* `fix_kappa=0` *ensures that the transition/transversion ratio will be estimated via maximum likelihood.*

Ideally, branch lengths should be jointly optimized with the other model parameters (`method=0` *within the control file). However, for larger datasets, joint optimization can make program runtimes very long. Faster results can be obtained by setting* `method=1`*, which causes the program to use a faster optimization algorithm that updates branches one at a time. Yet another alternative is to obtain branch lengths from another source, such as ML under a simpler model, and then fix them for the branch-site analysis. To do this, the desired branch lengths must be added to the tree file, and* `fix_blength=2` *must be set in the control file. The latter options are less desirable, and are discouraged unless a joint optimization is prohibitive.*

IMPORTANT NOTE: *This unit is not a substitute for the PAML manual. Users must read the PAML manual to ensure that they have set their program variables correctly.*

IMPORTANT NOTE: *It is good practice to use tree file, sequence file, and result file names that are highly informative about their contents. These can be set within the control file using the* `seqfile=`, `outfile=`, *and* `treefile=` *variables. The program produces a variety of files with fixed names (e.g.,* `rates`, `rst`, `rub`*). You should not use these names for your own files, as they will be overwritten. Users should run the program with default settings once and familiarize themselves with the output of the program.*

8. Open a terminal prompt and navigate within your file system to your analytical directory. Check that the directory contains the required files, and that the control file is appropriate.

9. Run CODEML or CODEML_SBA and examine results.

   A variety of files will be produced. For this model, you will want to open and examine the contents of the main results file (`Ceacam_ModelA.out` *if you used the file name suggested for the example dataset) and the* `rst` *file. Both are plain text files. These files contain a large amount of information. Figure 6.15.3 provides a portion of results output by CODEML, with annotations, to help identify particularly useful output such as the likelihood score, the estimated values of the ω distribution, and a Newick tree with the estimated branch lengths.*

USING THE BOOTSTRAP TO ASSESS IF THE REQUIREMENTS FOR INFERENCE HAVE BEEN MET

BASIC PROTOCOL 2

ML estimates (MLEs) of model parameters can provide insight into the evolutionary process, including the influence of purifying or positive selection at sites, or along branches. However, the estimates often have very large errors and they can sometimes be unstable. Instability refers to parameter estimates that have considerable mass associated with divergent values. Rather than simply assuming that the ML estimates have converged to a representative value, we can use the non-parametric bootstrap to verify this assumption. When parameter estimates are well behaved (i.e., when regularity conditions hold), the parameter estimates should have unimodal and symmetric distributions. When regularity conditions are violated, parameter estimates will be non-Gaussian and over-dispersed. Without regularity conditions, the desirable priorities of consistency and efficiency cannot be assumed for ML.

Bootstrapping is a procedure of "sampling with replacement." In this context, columns within a multi-sequence alignment are sampled at random to create many new alignments with different distributions of site patterns than the original. The program CODEML_SBA is used to bootstrap an original sequence alignment (select alignment columns at random with replacement) and to infer an ω distribution from each new alignment. These bootstrap distributions should have good properties when regularity conditions have been met. For this protocol, we provide two example datasets (*Ceacam* and *NR1D1*) where the sequences were sampled exclusively from primates. Regularity conditions hold for one dataset, *Ceacam*, and bootstrapping reveals that the MLEs are well behaved. Regularity conditions are not met for the other dataset, *NR1D1*, leading to bimodal distributions for both p_0 and ω.

Necessary Resources

Hardware

CODEML_SBA can be compiled to run on computers running Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems

Software

See Support Protocol 2 for details about how to download and install CODEML_SBA

Files

Bootstrapping requires: (i) a tree file in standard Newick format, (ii) a sequence alignment file in standard PAML format, and (iii) a control file. The tree files and alignment files for two example datasets are provided online (see Basic Protocol 1). The required configuration file for CODEML_SBA is included in the software download.

1. Obtain and install CODEML_SBA (see Support Protocol 2).
2. Create a new analytical directory to hold the input and the result files for a single run.
3. Obtain the sequence alignment file and tree file required for the desired analysis and copy them to your analytical directory.

   The files for the first example dataset are:

   Alignment file: `Ceacam_seq.txt`

   Tree file: `Ceacam_tree.txt`

   These files are available at http://www.currentprotocols.com/protocol/BI0615.
4. Make a copy of the default configuration file (`codeml.ctl`) for CODMEL_SBA and place the copy within the same analytical directory.
5. Open codeml.ctl in a plain text editor and edit the lines within this file as follows:

```
seqfile       = Ceacam_seq.txt       * contains sequence alignment
* seqfile     = boot.txt             * filename for bootstrap data
treefile      = Ceacam_tree.txt      * tree structure file name
bootstrap     = 100                  * N bootstrap alignments
* ndata       = 100                  * number of datasets
cleandata     = 1                    * remove sites with ambiguity ...
* sba         = 1                    * smoothed bootstrap aggregation
```

   These settings are necessary for the first of a two-step procedure. The objective of the first step is to generate 100 bootstrap samples based on the original alignment (`Ceacam_seq.txt`*).*

   IMPORTANT NOTE: *An asterisk (*`*`*) is used to indicate a comment, with any subsequent text ignored. The asterisk at the start of three of the above lines (e.g.,* `* sba = 1`*) is critical to proper program configuration in this step. Note that there are no asterisks at the start of the other three lines (this is also critical).*
6. Run CODEML_SBA to obtain the bootstrap samples.

   This step of the procedure is very fast, and will produce a single file called `boot.txt`*. This file will contain 100 sequence alignments, each obtained by bootstrap sampling of the columns within the original sequence alignment.*
7. Open `codeml.ctl` in a plain text editor and edit the following lines within this file:

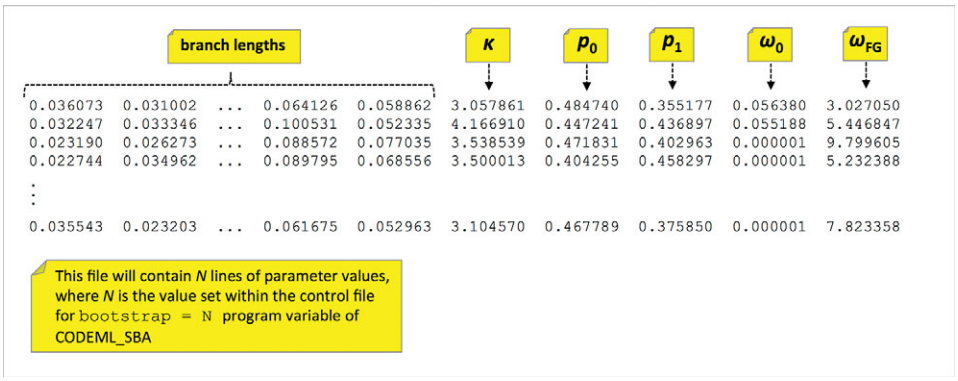


Figure 6.15.4 Annotated portion of results from the `sba.params` file. The file contains the maximum likelihood estimates of model parameters for each bootstrap dataset. The annotations help identify the parameters of the ω distribution under model A: p_0 , p_1 , ω_0 and ω_{FG}. Note that p_2 is not printed to the file because it is equal to 1- $p_0 + p_1$, and ω_1 is not printed to the file because it is fixed at 1.0 in this model.

```
* seqfile       = Ceacam_seq.txt     * contains sequence alignment
seqfile         = boot.txt           * filename for bootstrap data
treefile        = Ceacam_tree.txt    * tree structure file name
ndata           = 100                * number of datasets
* bootstrap     = 100                * N bootstrap alignments
cleandata       = 1                  * remove sites with ambiguity ...
sba             = 1                  * smoothed bootstrap aggregation
```

Do not forget that you must also set the program variables required for the codon model you wish to assess. As the focus of this protocol is Model A, the following program variables must be set as follows:

```
model         = 2       * models for codons ...
NSsites       = 2       * 0:one w;1:neutral;2:selection...
```

8. Run CODEML_SBA a second time to obtain MLEs for bootstrap datasets.

   This step of the procedure is much slower, and may take several hours (depending on the computer).

   All the estimated parameters (including branch lengths) for each bootstrap dataset are provided as a single line in a file called `sba.params`. *If* `bootstrap = 100` *is specified, then* `sba.params` *should contain 100 lines of parameter values.*

   The order of parameters in each line within `sba.params` *begins with the branch lengths and ends with the parameters of the ω distribution. For branch-site codon Model A, the ω distribution is given by the last 4 values in each row. Those parameters are: p_0, p_1 , ω_0, and ω_{FG}. Figure 6.15.4 provides a portion of results from an* `sba.params` *file, with annotations to help identify particularly useful output.*

9. Plot the distributions for the ω parameters and the mixture proportions (p). Inspect these distributions for signs that the required statistical regularity conditions have been met.

   These distributions should be approximately unimodal and bell-shaped. A statistical software package (e.g., R) can be used to plot the distributions.

   IMPORTANT NOTE: *Truncation of an otherwise bell-shaped distribution due to a restriction of the parameter values is acceptable (i.e., p parameters cannot be less than 0*

and ω_{FG} cannot be less than 1). A long right tail for the ω_{FG} distribution is also acceptable, as it is often difficult to estimate larger values, even when regularity conditions have been met.

IMPORTANT NOTE: *Instabilities in the parameter estimates are indicated by strongly bimodal distributions for the p parameters of the ω distribution. Such bootstrap distributions indicate that the conditions for reliable estimation of the ω distribution, as well as inference under the LRT, have not been met.*

10. Obtain the sequence alignment file and tree file required for the second example analysis and copy them to a different analytical directory

    Alignment file: `NR1D1_seqfile.txt`

    Tree file: `NR1D1_treefile.txt`

    These files are available at http://www.currentprotocols.com/protocol/BI0615. Open each file and familiarize yourself with its contents.

11. Repeat steps 3 through 9. Compare the bootstrap distribution for each of the four free parameters of the Model A ω distribution (p_0, p_1 , ω_0 *and* ω_{FG}) between the two example datasets.

BASIC PROTOCOL 3

TESTING THE HYPOTHESIS OF EPISODIC EVOLUTION AND MAKING SITE-SPECIFIC INFERENCES

Hypothesis testing via the LRT is reliable when regularity conditions are met (see Basic Protocol 2). In this protocol, branch-site codon Model A (Fig. 6.15.1) serves as the basis to formally test the hypothesis that sites evolved under positive selection along the foreground branch of the *Ceacam* gene tree. The LRT is based on comparing the likelihood under Model A restricted such that $\omega_{FG} = 1$ (Fig. 6.15.1; null model) to Model A permitted to have $\omega_{FG} > 1$ (Fig. 6.15.1; alternative model). When this LRT is significant, the MLEs from Model A with $\omega_{FG} > 1$ are input to Bayes' formula for the purpose of computing the posterior probability that each site could have evolved with $\omega_{FG} > 1$. CODEML employs a method called Bayes Empirical Bayes (BEB) to accommodate potentially large estimation errors in the parameters of Model A.

Necessary Resources

Hardware

CODEML or CODEML_SBA can be compiled to run on computers running Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems

Software

Users can choose either the CODEML or CODEML_SBA; the results will be equivalent. See Support Protocols 1 and 2 for further details about how to download and install these programs.

Files

The protocol requires: (i) a tree file in standard Newick format, (ii) a sequence alignment file in standard PAML format, and (iii) a control file. The tree file and alignment file for the sample data are provided online (see Basic Protocol 1). The required control file is included in the CODEML and CODEML_SBA software download.

Prepare files

1. Obtain and install CODEML or CODEML_SBA (see Support Protocols 1 and 2).
2. Create an analytical directory to hold both the input files and the result files.

3. Obtain the sequence alignment file and tree file required for the desired analysis, and copy them to your analytical directory.

   The example datasets files for this protocol are:

   Alignment file: `Ceacam_seq.txt`

   Tree file: `Ceacam_tree.txt`

   These files are available at http://www.currentprotocols.com/protocol/BI0615.

4. Make two copies of the default configuration file (`codeml.ctl`) and place both of them within the same analytical directory.

   Name the alternative copies of `codeml.ctl` *as follows:*

   ```
   null_codeml.ctl
   alt_codeml.ctl
   ```

   The LRT requires comparing results obtained from a constrained version of codon Model A ($\omega_{FG} = 1$) to those obtained from a less constrained version of that model ($\omega_{FG} > 1$). The differences in the analyses are achieved by changing settings in the control file. Keeping two versions of `codeml.ctl` *within this analytical directory (as opposed to changing and overwriting a single copy of this file) maintains a direct record of how the software was configured for each analysis.*

5. Open `null_codeml.ctl` in a plain text editor and edit the following lines within this file:

   ```
   seqfile     = Ceacam_seq.txt        * sequence data filename
   treefile    = Ceacam_tree.txt       * tree structure file name
   outfile     = Ceacam_MA_Null.out    * main result file name
   seqtype     = 1                     * 1:codons; 2:AAs; 3:codons-->AAs
   model       = 2                     * models for codons ...
   NSsites     = 2                     * 0:one w;1:neutral;2:selection ...
   fix_omega   = 1                     * 1: omega_1 fixed, 0: estimate
   omega       = 1                     * initial or fixed omega, for ...
   cleandata   = 1                     * remove sites with ambiguity ...
   ```

6. Open a terminal prompt and navigate within your file system to your analytical directory. Check that this directory contains the required files.

7. Run CODEML with the first control file (`null_codeml.ctl`).

   Because this control file does not use the default name, you must direct CODEML to use the alternatively named file by providing its name at the command line when the program is executed. For this example:

   ```
   codeml null_codeml.ctl
   ```

8. Open the main result file and inspect the contents.

   Figure 6.15.3 provides annotated program output for Model A. Note that Figure 6.15.3 shows results obtained from the alternative form of Model A; results under the null form of the model will be different.

9. Rename the `rst` file to `rst_MA_null.txt`.

   IMPORTANT NOTE: *CODEML writes results to files with names such as* `lnf`, `rates`, `rst`, *and* `rub`. *These files are overwritten each time the program is run. If you want to preserve any results within these files they must be re-named.*

10. Open `alt_codeml.ctl` in a plain text editor and edit the following lines within this file:

```
seqfile      = Ceacam_seq.txt       * sequence data filename
treefile     = Ceacam_tree.txt      * tree structure file name
outfile      = Ceacam_MA_alt.out    * main result file name
seqtype      = 1                    * 1:codons; 2:AAs; 3:codons-->AAs
model        = 2                    * models for codons ...
NSsites      = 2                    * 0:one w;1:neutral;2:selection...
fix_omega    = 0                    * 1: omega_1 fixed, 0: estimate
omega        = 1                    * initial or fixed omega, for ...
cleandata    = 1                    * remove sites with ambiguity ...
```

11. Open a terminal prompt and navigate within your file system to your analytical directory. Check that this directory contains the required files.

12. Run CODEML with the second control file (`alt_codeml.ctl`).

    Because this control file does not use the default name, you must direct CODEML to use the alternative control file by providing its name at the command line when the program is executed. For this example:

```
codeml alt_codeml.ctl
```

13. Open the main result file and inspect the contents.

14. Rename the `rst` file to `rst_MA_alt.txt`.

15. Inspect and compare the MLEs of the model parameters obtained under both the null and alternative versions of Model A.

    It is good practice to also review the estimates of other model parameters, and the empirical estimates of the codon frequencies.

    In this case it is important to inspect the branch lengths, looking for any implausibly large estimates, or for a very large proportion of branches with very small, or zero, length.

Formal testing of the hypothesis of sites evolving under positive selection along the foreground branch

16. Obtain the log likelihood scores for both Null Model A ($\omega_{FG} = 1$) and Alternative Model A ($\omega_{FG} > 1$). See annotations within Figure 6.15.3 to help identify the likelihood score within the main result file. Multiply the absolute difference in log likelihood scores by 2, and compare to a χ^2 distribution with 1 degree of freedom. This constitutes a LRT for an episode of positive selection along the FG branch at some sites within the data.

17. *Optional:* To check if the result of the LRT carried out in step 16 is sensitive to the assumed model, repeat this protocol under several alternate strategies for modeling codon frequencies (see Support Protocol 4).

Posterior identification of which sites evolved under positive selection along the foreground branch

18. Open the `rst` file for alternative Model A (`rst_MA_alt.txt`), identify the set of Bayes Empirical Bayes posterior probabilities (BEB-PPs) for each site, and copy those to a new plain text file, or into a spreadsheet.

    Figure 6.15.5 provides a portion of the `rst` *file, with annotations, to help identify the results relevant to BEB-PPs.*

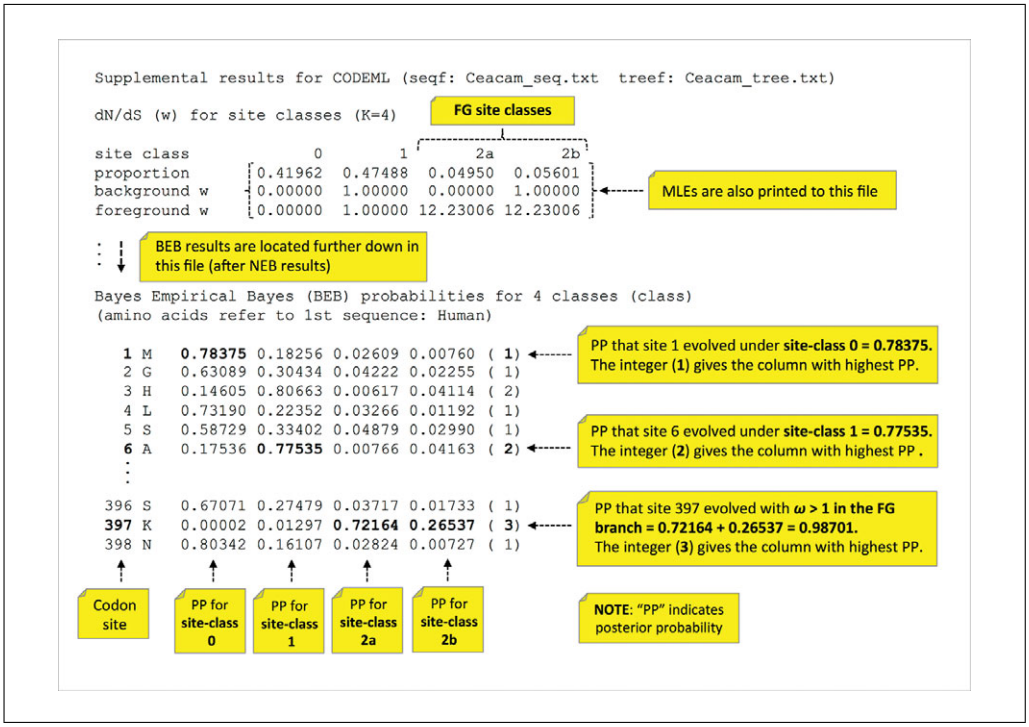


Figure 6.15.5 Annotated portion of the supplementary output of CODEML printed to the `rst` file. This file contains site-wise calculation of the NEB- and BEB-derived posterior probabilities of each evolutionary regime (site class) within Model A. As BEB is usually preferable to NEB (when both are available), only the BEB portion of the output is presented.

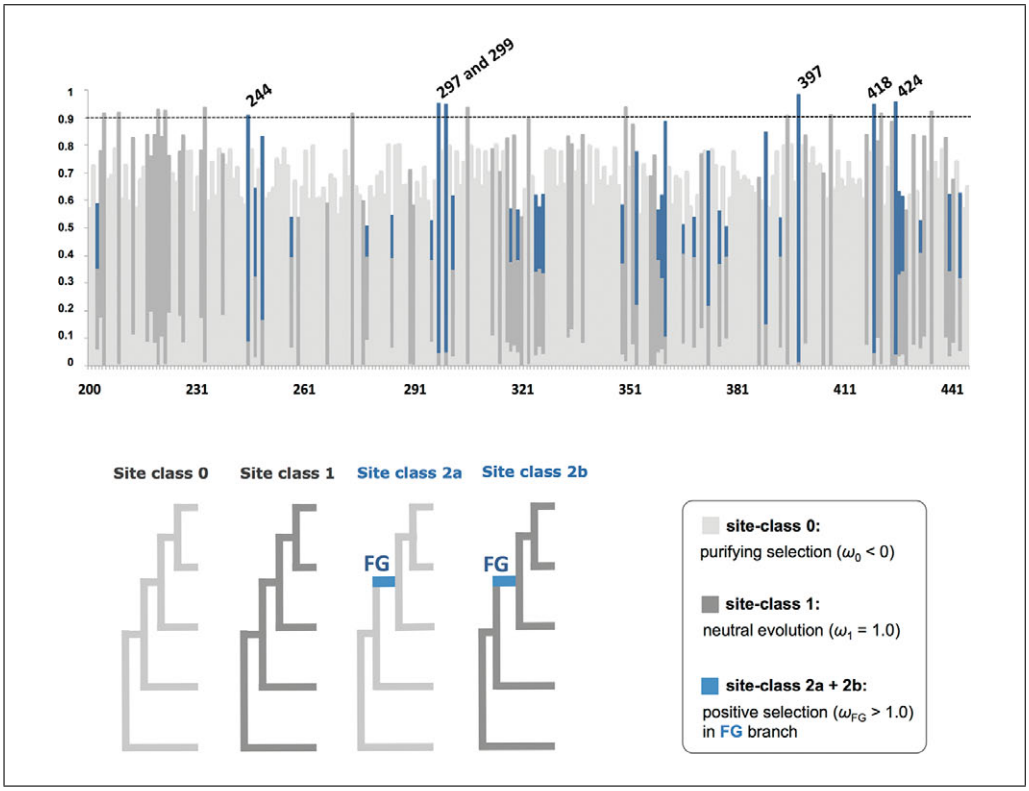


Figure 6.15.6 Plot of the posterior probability of purifying selection ($\omega_0 < 1$), neutral evolution ($\omega_1 = 1$), and positive selection in the foreground branch ($\omega_{FG} > 1$) for half of the sites in the primate *Ceacam* dataset. Results are plotted only for the second half of the data to improve the interpretability of the plot, and because those sites with high posterior probability of $\omega_{FG} > 1$ are located in this region of *Ceacam*. The posterior probabilities are computed using the BEB method under codon Model A, with the values at each site summing to 1.0.

The BEB-PPs for each site include the posterior probability that it evolved under positive selection along the FG branch (i.e., $\omega_{FG} > 1$). Inference of such sites should be based on a threshold chosen by the user. Typically, conservative thresholds are adopted (e.g., 0.90, 0.95, or 0.99).

For the Ceacam dataset, there are 6 sites with BEB-PP > 0.90, and 5 > 0.95. Similar results are obtained under the Naïve empirical Bayes (NEB) method; when both NEB and BEB are available, BEB is recommended (Yang et al., 2005).

In addition to identifying sites having the highest probability of positive selection, these data can be imported into plotting software to produce a graphical summary of the distribution of selection regimes among sites. Such a plot is shown for half of the Ceacam gene in Figure 6.15.6.

19. *Optional:* Use the results obtained under alternative codon frequency models (Support Protocol 4) to check if BEB-based site identification is sensitive to the assumed model.

SUPPORT PROTOCOL 1

OBTAIN AND INSTALL PAML

This protocol describes how to obtain and install the PAML package of programs for Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems. The PAML package includes the program CODEML.

Necessary Resources

Hardware

Any system with an Internet connection and a Web browser

NOTE: The protocols in this unit are for the command-line version of the PAML package. Do not download PAML-X, which employs a graphical user interface. Versions of PAML-X for OS X and Linux have not been comprehensively tested.

For Windows

1a. Download the latest version of the PAML software package for Windows from the PAML Web site: *http://abacus.gene.ucl.ac.uk/software/paml.html#download*.

The package is distributed as an archive. The archive will have the name `paml*.tgz`*, where the * represents the current version number.*

2a. Unpack the archive into a local folder.

3a. Pre-compiled Windows executable files are provided within `paml*\bin\`. Place the executables in a directory that is included in your search path.

For OS X

1b. Download the latest version of PAML for OS X from: *http://abacus.gene.ucl.ac.uk/software/paml.html#download*.

The archive named `paml*.macosx.tgz` *contains pre-compiled OS X executable files within* `paml*/bin/`.

2b. Unpack the archive into a local folder. Move the executables to a directory that is included in your search path, or add `paml*/bin/`to your search path.

For FreeBSD

1c. Packages can be installed on FreeBSD using the binary package management tool (pkg). Use the following command to install PAML:

```
pkg install paml
```

For GNU/Linux

1d. Some GNU/Linux systems maintain a PAML package that is easy to install. For example, the Advanced Package Tool (APT) can be used to install executables on Debian-based GNU/Linux systems

Use the following command to install PAML via `apt-get`:

```
apt-get install paml
```

The package can sometimes lag behind the latest version, and other GNU/Linux systems do not offer a PAML package. In those cases, PAML should be installed from source (see steps 1e to 4e, below)

Installation from source for OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems

1e. Download the latest version of the PAML software package from the PAML Web site: *http://abacus.gene.ucl.ac.uk/software/paml.html#download*

The package is distributed as an archive. The archive will have the name `paml*.tar.gz`*, where the* `*` *represents the current version number.*

2e. Delete the Windows executable files within `paml*/bin/`.

3e. Navigate within your file system to the `paml*/src/` folder (this is where the source files are located). Use the following command to compile the programs:

```
make -f Makefile
```

If the make command fails, you might have to edit the `Makefile` *located within the* `paml*/src/` *folder. Additional information about compiling the programs is provided in the PAML manual, and online at the PAML Web site. The GNU* `make` *command on FreeBSD is* `gmake`*. It must be installed with* `pkg install gmake`*.*

4e. If you have a folder for local programs inside your home account, then you can move the executable files there. Otherwise, move the executable files to the `paml*/bin/folder` and add this folder to your search path.

OBTAIN AND INSTALL CODEML_SBA FOR UNIX/UNIX-LIKE AND OS X SYSTEMS

SUPPORT PROTOCOL 2

This protocol describes how to obtain and install CODEML_SBA for OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems.

Necessary Resources

Hardware

Unix or Unix-like system with an Internet connection and a Web browser

1. Download the latest version of CODEML_SBA from *https://github.com/Jehops/codeml_sba* and unpack the archive into a local folder. If you prefer, you can clone the repository with `git clone` (*https://github.com/Jehops/codeml_sba*).

2. Navigate within your file system to the `codeml_sba/` folder, where the source files are located. Use the following command to compile the program:

For systems using BSD make, type: `make -f Makefile`

For systems using GNU make, type: `make -f Makefile.gnu`

3. If you have a folder for local programs inside your home account, then you can move the executable files there.

Unix-like systems use the `chmod` *command to set file permissions, including permitting the file to function as an executable. To make CODEML_SBA executable type* `chmod +x codeml_sba.`

SUPPORT PROTOCOL 3

LABELING THE FOREGROUND BRANCH OF A NEWICK TREE

This protocol describes how to download and install the program DENDROCYPHER and use it to label a branch within a Newick formatted tree file. Figure 6.15.7 uses a simple 4-taxon tree to illustrate the relationship between the Newick format and the branches of a phylogeny that will have a unique parameter in a model (e.g., the foreground branches having ω_{FG} in Model A).

Hardware

Any system with an Internet connection and a Web browser

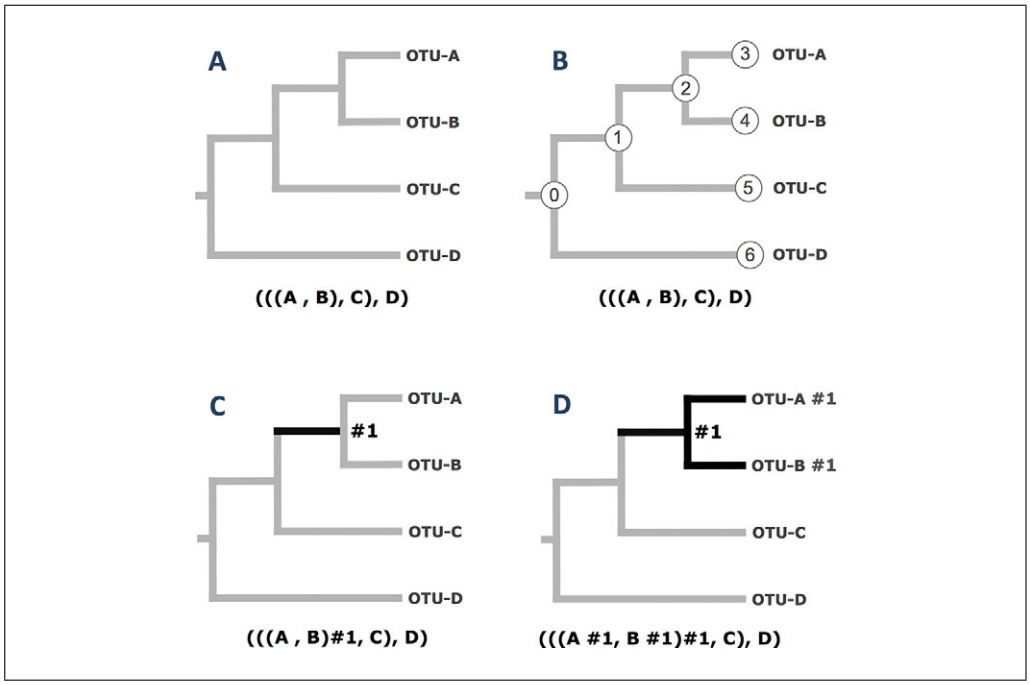


Figure 6.15.7 The relationship between a labeled Newick tree and the branches of a phylogeny that will have a unique parameter in a model. The operational taxonomic units (OTUs) at the tips of the tree are labeled A, B, C, and D. Those branches that a user wants to have unique model parameters are referred to a "foreground branches." The foreground branches of each phylogeny are shown in bold. The Newick tree (parenthetical tree notation) corresponding to each phylogeny is shown below each case. In this example, the foreground branch is indicated within a Newick tree using the #1 label. (**A**) An unlabeled phylogenetic tree. This tree will have no branch-specific parameters. (**B**) The same (unlabeled) tree as in A, but with node IDs shown for both tip and internal nodes. The Newick tree below the phylogeny is the same as in A; this is because Newick trees do not include information about node IDs. The node IDs shown are those assigned by the program DENDROCYPHER. Note that different programs can assign node IDs differently. The ID for a branch is equal to the ID of its descendant node. (**C**) A phylogenetic tree having a single foreground branch, shown in bold. The foreground branch will have ω_{FG} in Model A. Because different programs can label the same node differently, the user must supply the identifier for the FG branch in the Newick tree. In this case the identifier is "#1". Note that the FG branch in this tree is equivalent to branch 2 in tree B above. (**D**) A phylogenetic tree having a set of three foreground branches, shown in bold. All three foreground branches will have ω_{FG} in Model A because they all have the same identifier.

1. Download the latest version of DENDROCYPHER from *https://bitbucket.org/EvoWorks/dendrocypher/downloads*.
2. Unpack the archive into a local folder.
3. Use the following command to compile the program:

```
make -f Makefile
```

4. Create an analytical directory and move the executable file there.
5. Download FigTree from *http://tree.bio.ed.ac.uk/software/figtree/*.

   FigTree is a program for tree manipulation and visualization. The feature required for this protocol is its ability to read user-defined labels within a Newick formatted tree file and display both the tree and labels via its graphical interface.

   Pre-compiled Windows, OS X, and Java executables are provided at the above site. The Java version should run on any system with Java version 1.5 or higher installed.

6. Place a copy of the tree file you want to annotate within the same analytical directory that contains the DENDROCYPHER executable.

   DENDROCYPHER takes as input a plain text file containing a tree in Newick format. The tree must be rooted. The file must be named `tree.txt`.

7. Navigate within your file system to the directory that contains the `DendroCypher` executable and run the program by typing `DendroCypher` at the command prompt.

   DENDROCYPHER uses a simple menu system within the terminal window as its user interface. Under MENU *is a numbered list of options. Program options are selected by entering a menu option at the command prompt.*

8. Enter `1` at the prompt to show the tree on the screen.

   Take this opportunity to examine the tree on the screen to make sure it is the one that you want to annotate. Note that every node of this tree is annotated with an integer value. These are the node IDs. Branches are labeled in DENDROCYPHER by specifying a node ID.

9. If the tree is correct, enter `2` at the prompt to print it to a file called `numbered_tree.txt`
10. Open this file in FigTree to visualize the tree with each node numbered.

    Use the following FigTree menu options to open the file: File→ Open→ `numbered_tree.txt`.

    FigTree will recognize that the nodes of the tree are labeled and a dialog box will appear asking you to supply a name for the labels it read from the tree. You can supply your own name. We recommend "Node IDs."

11. To show the node IDs in the tree (i) check the Node Labels box on the left side of the program window, then (ii) expand the Node Labels option panel, and finally (iii) change the value for the Display variable from "node ages" to Node IDs.

    This will cause the Node IDs within `numbered_tree.txt` *to appear on the screen.*

12. Based on the graphical representation of the tree topology in FigTree, make note of the node ID(s) that corresponds to the branch, or branches, of the tree you wish to label.

    Within DENDROCYPHER, branch numbers are equal to the ID of the node that they connect to as an ancestral branch. So, a terminal branch number is equal to its tip node ID. Likewise, an interior branch number is equal to the ID of the descendant node of the branch. So, in FigTree, choose the number shown at a node to indicate the horizontal branch that connects directly to that node.

		AAA	AAA	AAA	
example substitution		$A^1 \rightarrow C^1$	$A^2 \rightarrow C^2$	$A^3 \rightarrow C^3$	
		CAA	ACA	AAC	
equilibrium codon frequency	Fequal	1/61	1/61	1/61	np = 0
	F1×4	$\pi_C\pi_A\pi_A$	$\pi_A\pi_C\pi_A$	$\pi_A\pi_A\pi_C$	np = 3
	MG	$\pi_C{}^1$	$\pi_C{}^2$	$\pi_C{}^3$	np = 9
	F3×4 (GY)	$\pi_C{}^1\pi_A{}^2\pi_A{}^3$	$\pi_A{}^1\pi_C{}^2\pi_A{}^3$	$\pi_A{}^1\pi_A{}^2\pi_C{}^3$	np = 9
	F61 (GY)	π_{CAA}	π_{ACA}	π_{AAC}	np = 60

Figure 6.15.8 Alternative models for the equilibrium value of each codon. The heading of the figure illustrates the effect on codon substitution when the same nucleotide change (A → C) occurs at the first, second, and third positions of an AAA codon. The equilibrium frequencies are given for the target codon under each of 5 alternative strategies for modeling its equilibrium value. The π_i parameter is the frequency of the i^{th} nucleotide. Superscripts denote the position of the codon (1, 2, or 3). When there is no superscript, the π_i parameter denotes a value for all the sites within the data. Fequal assumes that all sense codons are equally frequent; this is 1/61 under the standard genetic code. F1 × 4 models the equilibrium frequency of the target codon as a simple product of the empirical frequencies of the four nucleotides within the data. MG does not model empirical codon frequencies. Rather, it models codon transition probabilities as proportional to the frequency of the target nucleotide at the position where the substitution took place. F3 × 4 models the equilibrium frequency of the target codon as a product of the empirical frequencies of the four nucleotides at each position of the codon. F61 uses the empirical estimates of each of the 61 codons as their equilibrium frequencies. The number of free parameters in each model is denoted np. Because frequencies must sum to 1, some parameter values can be obtained by subtraction, and do not count as free parameters within np. MG denotes Muse and Gaut (1994). GY denotes Goldman and Yang (1994).

13. Re-launch DENDROCYPHER and re-load the tree.

14. Enter `5` to label the desired branch(es) of the tree.

    The program will prompt you to enter a node ID and then a label for that node ID. The label must be an integer.

    To label a branch, supply the ID of the descent node for that branch.

15. Enter `8` to view a Newick formatted tree with the labels you have entered.

    Branches in a tree with labels (if any) are denoted by a "# mark" in the tree file. This marks the branch to be labeled, and the value of the label is a user-defined integer value. See Figure 6.15.7 for simple examples of labeled and unlabeled trees.

16. If the tree looks correct, enter `9` to print the labeled tree to a file called `marked_tree.txt`.

    The tree within the file `marked_tree.txt` *now has a branch or branches marked with a user-defined integer. These branches will be treated as the "foreground" branches in the branch-site codon models. This tree file can be read directly by CODEML or CODEML_SBA.*

SUPPORT PROTOCOL 4

ASSESS ROBUSTNESS OF RESULTS TO ALTERNATIVE MODELS FOR CODON FREQUENCIES

CODEML and CODEML_SBA allow users to choose among several different strategies for modeling the equilibrium value of each codon (Fig. 6.15.8). For some datasets, these alternatives can lead to very different equilibrium values, which in turn can make the inferences sensitive to the chosen model. This protocol indicates how to check for robustness across three alternative models that are widely used in real data analyses.

Necessary Resources

Hardware

CODEML or CODEML_SBA can be compiled to run on computers running Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems

Software

Users can choose either the CODEML or CODEML_SBA; the results will be equivalent. See Support Protocols 1 and 2 for further details about how to download and install these programs.

Files

The protocol requires: (i) a tree file in standard Newick format, (ii) a sequence alignment file in standard PAML format, and (iii) a control file. The tree file and alignment file for the sample data are provided online (see Basic Protocol 1). The required control file is included in the CODEML and CODEML_SBA software download.

1. Open `codeml.ctl` in a plain text editor and edit the following line within this file:

   ```
   CodonFreq = 2 * 0:1/61 each, 1:F1X4, 2:F3X4 ...
   ```

   This specifies the Goldman and Yang (1994) F3×4 model, which assumes that transition probabilities are a function of equilibrium frequency of the ***target codon****. The codon equilibrium frequency is computed using the empirical frequencies of each nucleotide at the three positions of the codon (F3×4-GY in Fig. 6.15.8). The model has* ***9 free parameters****.*

   Ensure that all other program variables are set appropriately within the control file. Recall that setting `model = 2` *and* `NSsites = 2` *is required to specify branch-site codon Model A.*

2. Open a terminal prompt and navigate within your file system to your analytical directory. Check that the analytical directory contains the required files, and that the control file is appropriate.

3. Run CODEML or CODEML_SBA.

4. Rename the `rst` file.

5. Open `codeml.ctl` in a plain text editor and edit the following line within this file:

   ```
   CodonFreq = 5 * 0:1/61 each, 1:F1X4, 2:F3X4 ...
   ```

   This specifies the Muse and Gaut (1994) F3×4 model. It assumes that transition probabilities are a function of the equilibrium frequency of the ***target nucleotide*** *at the position of the codon where the state change occurred (MG in Fig. 6.15.8). This model also has* ***9 free parameters****.*

   IMPORTANT NOTE: *Change the name of the main result file via* `outfile =`*, or the results from the previous run will be overwritten.*

6. Repeat steps 2 to 4.

   Do not forget to rename the `rst` *file.*

7. Open `codeml.ctl` in a plain text editor and edit the following line within this file:

   ```
   CodonFreq = 3 * 0:1/61 each, 1:F1X4, 2:F3X4 ...
   ```

*This model assumes that transition probabilities are a function of equilibrium frequency of the **target codon**, and the frequency of each codon is empirically estimated from the data (F61-GY in Fig. 6.15.8). Under the universal genetic code, this model would have **60 free parameters**. This model is not recommended for small datasets.*

IMPORTANT NOTE: *Change the name of the main result file via* `outfile =`*, or the results from the previous run will be overwritten.*

8. Repeat steps 2 to 4.

   Do not forget to rename the `rst` *file.*

9. Obtain the (i) MLEs of model parameters, (ii) the log likelihood score, and (iii) the BEB posterior probability that each site evolved under positive selection.

   By comparing these results, including any likelihood ratio tests, users can investigate the robustness of their biological inferences to alternative strategies for modeling the equilibrium value for each codon.

   To assess the robustness of an LRT, steps 1 to 9 above must be repeated for both of the models (null and alternative) that comprise the LRT.

SUPPORT PROTOCOL 5

SMOOTHED BOOTSTRAP AGGREGATION FOR IDENTIFYING SITES WITH A HISTORY OF POSTIVE SELECTION

This protocol describes how to use smoothed bootstrap aggregation (SBA) to identify sites subject to positive selection in the FG branch under Model A. The method is implemented in the program CODEML_SBA.

Necessary Resources

Hardware

CODEML_SBA can be compiled to run on computers running Windows, OS X, FreeBSD, GNU/Linux, and other Unix-like operating systems

Software

See Support Protocol 2 for details about how to download and install CODEML_SBA

Files

Bootstrapping requires: (i) a tree file in standard Newick format, (ii) a sequence alignment file in standard PAML format, and (iii) a control file. The tree files and alignment files for two example datasets are provided online (see Basic Protocol 1). The required configuration file for CODEML_SBA is included in the software download.

1. Obtain and install CODEML_SBA (see Support Protocol 2)
2. Create a new analytical directory to hold the input and result files for a single run.
3. Obtain the sequence alignment file and tree file required for the desired analysis, and copy them to your analytical directory.
4. Make a copy of the default configuration file (`codeml.ctl`) for CODMEL_SBA and place the copy within the same analytical directory.
5. Open `codeml.ctl` in a plain text editor and edit the lines within this file for the **first step of a three-step procedure**:

```
seqfile    = seqfile.txt    * contains sequence alignment
* seqfile  = boot.txt       * filename for bootstrap data
treefile   = treefile.txt   * tree structure file name
bootstrap  = 100            * generate N bootstrap ...
* ndata    = 100            * number of datasets
* sba      = 1              * smoothed bootstrap aggregation
* h        = .4             * bandwidth parameter...
```

The objective of the first step is to generate 100 bootstrap samples from an alignment.

6. Run CODEML_SBA to obtain the bootstrap samples.

   This step of the procedure is very fast, and will produce a single file called boot.txt. This file will contain 100 sequence alignments, each obtained by bootstrap sampling the columns of the original sequence alignment.

7. Open `codeml.ctl` in a plain text editor and edit the following lines for the **second step of the three-step procedure**:

```
* seqfile   = seqfile.txt  * contains sequence alignment
seqfile     = boot.txt     * filename for bootstrap data
treefile    = treefile.txt * tree structure file name
ndata       = 100          * number of datasets
* bootstrap = 100          * generate N bootstrap alignments
sba         = 1            * smoothed bootstrap aggregation
* h         = .4           * bandwidth parameter...
```

   Do not forget that you must also set the program variables required for the codon model you wish to assess. As the focus of this protocol is Model A, the following program variables must be set as follows:

```
model      = 2    * models for codons ...
NSsites    = 2    * 0:one w;1:neutral;2:selection...
```

   At the time of publication, SBA was implemented for codon models M2a, M8, and Model A. SBA will be added for additional models.

8. Run CODEML_SBA a second time to obtain MLEs for bootstrap datasets

   The step of the procedure is much slower, and may take several hours (depending on the computer).

   All the estimated parameters for each bootstrap dataset are provided as a single line in a file called `sba.params`. *If* `bootstrap = 100` *is specified, then* `sba.params` *should contain 100 lines.*

   The order of parameters in each line within `sba.params` *is the same order as printed within the main result file of CODEML and CODEML_SBA (see Fig. 6.15.4 for an example based on Model A).*

9. Open `codeml.ctl` in a plain text editor and edit the following lines for the **third step of the three-step procedure**:

```
seqfile    = seqfile.txt  * contains sequence alignment
* seqfile  = boot.txt     * filename for bootstrap data
treefile   = treefile.txt * tree structure file name
* ndata    = 100          * number of datasets
* bootstrap = 100         * generate N bootstrap alignments
sba        = 2            * smoothed bootstrap aggregation
h          =.4            * bandwidth parameter...
```

Do not forget to also set the program variables required for the codon model you wish to assess. For Model A, the following program variables must be set as follows:

```
model        = 2      * models for codons ...
NSsites      = 2      * 0:one w;1:neutral;2:selection...
```

The `h` *variable sets the bandwidth for kernel smoothing of the p parameters of the ω distribution. The value of* `h` *must be set between 0 and 1; a bandwidth of 0 means that the bootstrap distribution is unaffected by the kernel smoothing, whereas a value of 1 converts the bootstrap distribution to a uniform distribution. The default value,* `0.4`*, yields a moderately "smoothed" version of the bootstrap distributions for p, and has worked well for a variety of datasets.*

Note that parameter values derived from the bootstrap distribution are applied to the original dataset; thus, the sequence file must be re-set to the original data.

10. Run CODEML_SBA a third time to obtain a set of posterior probabilities for positive selection in the FG branch at each site. These values should then be aggregated in the form of the mean posterior probability, with inference at each site being based on the mean value. This form of inference is referred to as smoothed bootstrap aggregation.

    This step produces a comma separated values file (CSV file) that can be directly read by a spreadsheet application. Each value in the file is the posterior probability that a site evolved with $\omega_{FG} > 1$. The data in each row of the file is obtained from one set of bootstrap-derived MLEs (with some parameter values smoothed via a kernel-smoothing technique). Inference is based on the distribution of posterior probability scores within a column. The number of columns in this file is equal to the number of sites in the alignment, and the values of the i^{th} column are the posterior probabilities for the i^{th} site.

    Taking the average over the column of values for the i^{th} site yields a posterior probability that marginalizes over the uncertainty of all the model parameters, including the branch lengths. The values in each column can be further employed to compute a confidence interval for the posterior probability of positive selection at each site in the original alignment.

11. Compare the mean posterior probability for each to a threshold value (e.g., 0.95), and report those sites with values greater than the threshold.

GUIDELINES FOR UNDERTANDING RESULTS

Interpretation of the Parameter Estimates (See Basic Protocol 1)

The first step with any model-based analysis is to fit the chosen model to the data and carefully inspect the fitted parameter values. In the case of codon models, the MLEs of the branch lengths provide an important signal about the suitability of the sequence divergence represented by the data. Large numbers of very short (or zero) branch lengths could indicate that there is insufficient signal within the sample. When there is too little signal, the MLEs will have very high estimation errors, and can even be unstable. Instabilities suggest that the approximation to large sample theory required for the ML inferential framework cannot be assumed for the sample of data in hand (see Basic Protocol 2 for a method of assessing this requirement for inference). A well-known case where MLE instabilities appear to make the LRT for positive selection unreliable is the *Tax* gene (Suzuki and Nei, 2004), and in that case there are many branches with very small branch lengths. Although there are some small branches for the dataset employed in Basic Protocol 1 (*Ceacam*), there is enough sequence divergence within the tree to warrant an attempt at fitting Model A (see branch length estimates in Fig. 6.15.3). Basic Protocol 2 provides a means of assessing the reliability of inferences derived from Model A for the *Ceacam* dataset.

Unreasonably large branch lengths also can signal a problematic dataset. Note that branch lengths for codon models are defined as the number of nucleotide substitutions per codon site (not per nucleotide site). Thus, substitution saturation is indicated by branch length estimates >3 (not >1). Branch length MLEs >3 should be considered a potential cause for concern. When many branches are $>>3$, the data are likely too divergent for analysis under a codon model, and analysis under an amino acid model should be considered. The presence of a few, or just one, unreasonably long branch ($>>3$) can arise when there is an error in one or more of the foundational assumptions of the model. For example, non-stationarity in the codon equilibrium frequencies can lead to highly inflated branch lengths, which can negatively influence MLEs, LRTs, and site identification via empirical Bayes methods (Bay and Bielawski, 2011). None of the *Ceacam* branch lengths are unusually long, and all are <3, indicating that the sequences are not too divergent for a codon model (Fig. 6.15.3).

The probability of a change from any one codon state to another within the model can be modeled as depending on the equilibrium value of either the target codon (Goldman and Yang, 1994) or the target nucleotide at one of the three positions of the codon (Muse and Gaut, 1994). Figure 6.15.8 illustrates the difference between these two strategies, as well as some variant models. These are two very different strategies for modeling codon evolution, and when a gene has very biased codon usage, the choice of strategy can impact the ω MLEs (e.g., Aris-Brosou and Bielawski, 2006). It is good practice to examine the codon usage patterns within a dataset, and the *Ceacam* dataset has somewhat uneven codon usage. Taking the four-fold degenerate codon family for serine as an example clearly illustrates that there are both preferred and un-preferred synonymous codons for this gene. Codon TTC is highly preferred, being observed 158 out of the 263 times (60%) that a serine is encoded within this dataset. Alternatively, codon TCG is used just six times (2%) to encode a serine. These and other codon usage patterns suggest that results could be sensitive to the assumed model for codon frequencies. Such codon usage patterns indicate that inferences about episodic changes in selection intensity (Basic Protocol 3) should be assessed for robustness to the assumed model for equilibrium codon frequencies (Support Protocol 4).

The MLEs of the *Ceacam* ω distribution obtained for Basic Protocol 1 (Fig. 6.15.3) were obtained by using the F3 $\times$ 4 model (Fig. 6.15.8), which permits codon usage bias. Those results suggest that a large fraction of sites (42%) were subject to very strong purifying selection pressure ($\omega_0 = 0$) throughout the entire *Ceacam* phylogenetic tree, and another fraction (48%) evolved under relaxed purifying selection ($\omega_1 = 1.0$; fixed within the model). In addition, a small fraction (10%) were estimated to have evolved by episodic positive selection along the FG branch ($\omega_{FG} = 12.2$). Note that very large estimates of ω will often occur for an FG branch when there is strong signal in the data for a higher than neutral nonsynonymous substitution rate. However, at the same branch, the signal for synonymous change can be sparse, making it difficult to accurately estimate ω. In such cases (which are not uncommon), the model can detect the signal for $\omega > 1$, but it cannot precisely estimate how much greater than 1. As long as the conditions for ML inference have been met, the LRT can be used to formally test the hypothesis that $\omega_{FG} > 1$ (Basic Protocol 3) even when the value of ω_{FG} cannot be precisely estimated. Moreover, non-parametric bootstrapping (Basic Protocol 2) can be used to assess the uncertainty of the estimated values of parameters such as ω_{FG}.

In the case of the *Ceacam* phylogenetic tree, the foreground was specified as the branch separating the Old World and New World monkeys. Thus, fitting the model to these data has revealed a signal for an episode of adaptive divergence in *Ceacam* function between these groups of primates. Basic Protocol 3 provides the means to formally test the statistical significance of this signal. In addition, Basic Protocol 3 also includes empirical

Bayes inference of which sites evolved by positive selection along the FG branch. This is required because the MLE for p_2 ($p_{2a} + p_{2b}$) indicates only that there is a 10% chance that a randomly chosen site had positive selection along the FG branch—i.e., the MLEs do not tell us which sites they are. Readers are referred to Yang et al. (2005) for further details about empirical Bayesian methods for identification of positively selected sites.

Assessing if the Signal within a Dataset is Sufficient for Reliable Inferences (See Basic Protocol 2)

The target of inference in this case was a small fraction of sites (p_2) having a unique level of selection pressure along just a single branch (ω_{FG}). This is a challenging inference task, and users should expect that estimation error will be associated with the parameters of Model A, even in a well sampled dataset. However, the desirable properties of ML inference are achievable with such data if the required statistical regularity conditions have been met. In Basic Protocol 2, the non-parametric bootstrap was used as a means of corroborating statistical regularity for the *Ceacam* dataset. CODEML_SBA was used to obtain bootstrap distributions for all model parameters, but the p and ω parameters represent the biggest estimation challenge, and these should be checked. We found that these parameters had the unimodal distributions predicted by ML theory (e.g., p_0 and ω_{FG} are shown in Fig. 6.15.9). Note that values for some model parameters are restricted (e.g., $\omega_{FG} \geq 1$), and truncated bootstrap distributions are possible in such cases. Also, the MLE for ω_{FG} could be at the stop points for the optimization algorithm for some datasets. As these are not true components of the MLE distribution, users should take such limits into consideration when assessing the properties of the distribution. This is different from the boundary values on the p_i parameters (0 and 1), which are determined by the model (and, not the optimization algorithm). All the values obtained by bootstrapping *Ceacam* were included in distributions shown in Figure 6.15.9; those distributions appear unimodal, and symmetric when not too close to a boundary (p_o). Based on these results, the LRT and BEB techniques employed in Basic Protocol 3 are expected to be reliable for the *Ceacam* dataset.

A second dataset, *NR1D1,* was provided for Basic Protocol 2 for the purpose of illustrating potential instabilities in distributions. This is also a primate dataset, but it differs in that the human terminal linage is specified as the FG branch. This represents a very common scenario among studies of primate adaptive evolution, as they often seek to discriminate the human-specific evolutionary transitions from other events within the primate evolutionary tree. Although the *NR1D1* alignment is larger than *Ceacam* (12 versus 11 sequences, and 1842 vs. 1362 nucleotides), it appears to represent a sparser sample of evolutionary signal. The hominid clade branch lengths are generally much smaller for *NR1D1* than *Ceacam*, which is the region where the FG branch was specified. Further, by specifying the human linage as the FG branch, Model A is being focused on a branch that is well known to represent very low sequence divergence. For these reasons, it is reasonable to expect that the required regularity conditions might not be met for *NR1D1*. Parametric bootstrapping reveals that this was indeed the case (Fig. 6.15.9). The distributions of both p_0 and ω_{FG} were clearly bimodal, which contradicts expectations derived from ML theory. These results suggest that LRT and BEB techniques employed in Basic Protocol 3 might not be reliable for the *NR1D1* dataset. In general, studies of closely related sequences with complex codon models (such as episodic evolution in humans) will likely be much more prone to these problems.

Making Inferences about Episodic Changes in the Intensity of Natural Selection (See Basic Protocol 3)

Comparison of branch-site Model A with $\omega_{FG} > 1$ to Model A with $\omega_{FG} = 1$ is known as the "branch-site test for positive selection." Because there are several LRTs based on branch-site models, this LRT is sometimes simply referred to as "Test 2." The test

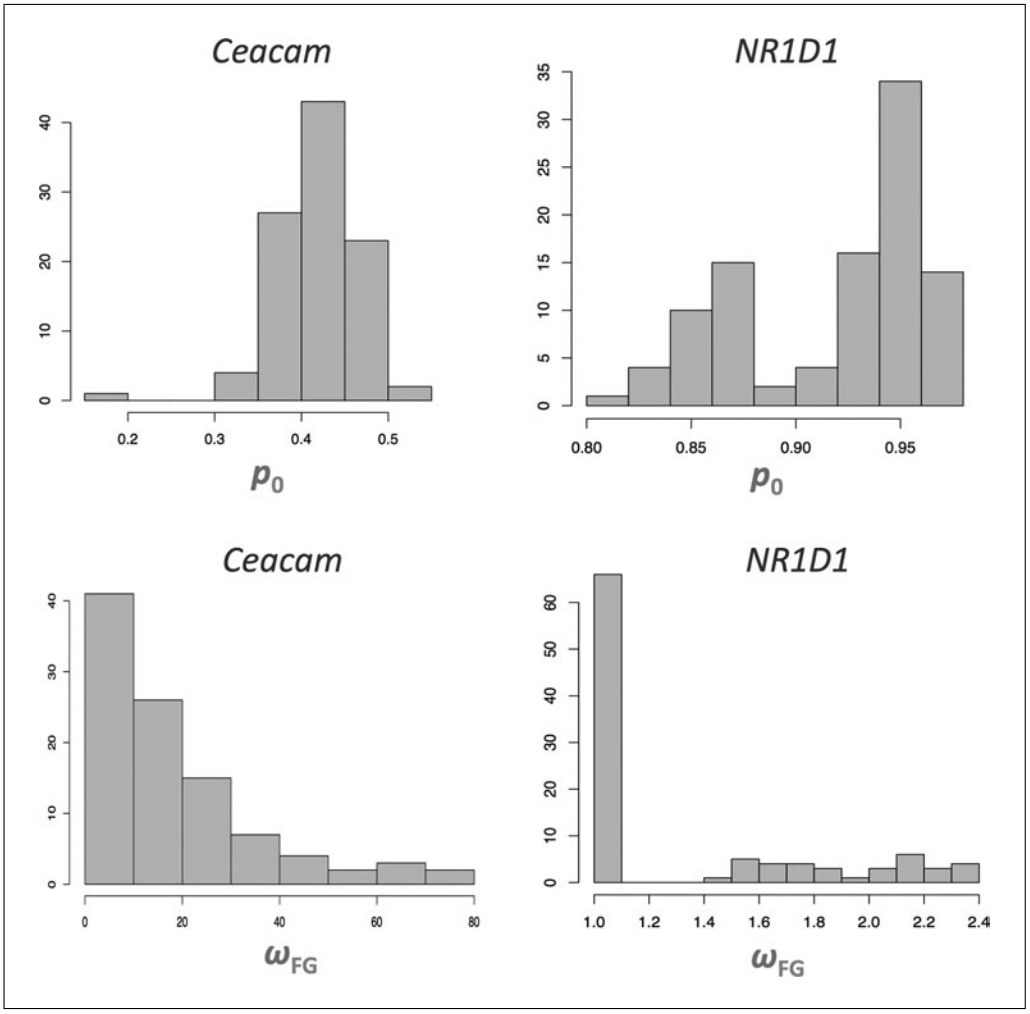


Figure 6.15.9 Bootstrap-based estimates of MLE distributions for *Ceacam* and *NR1D1*. Distributions are shown for parameters p_0 and ω_{FG}. The distributions for *Ceacam* are unimodal, and symmetric up to the boundary on the parameter space. The distributions for *NR1D1* are bimodal. As asymptotic normality has not been attained for the *NR1D1* estimates, the required regularity conditions have not been met for this gene.

statistic for this LRT ($2\Delta \ln L = 16.999$) is highly significant, with $p = 0.000037$ under χ^2_1. Because the null form of Model A is obtained by fixing $\omega_{FG} = 1$, the null distribution is actually a 50:50 mixture of χ^2_0 and χ^2_1 (at $\alpha = 0.05$, the critical value is actually 2.71 instead of 3.48). Thus, using χ^2_1 for this test is recommended (see PAML manual), as it makes the test a little more conservative. The *Ceacam* test statistic is highly significant, so it does not make a difference in this case.

The LRT indicates that there is significant evidence for $\omega_{FG} > 1$ within the *Ceacam* tree. This result permits two inferences about *Ceacam* evolution. First, this sample of data contains significant evidence for functional divergence between Old World and New World primates in a genetic system related to both tissue differentiation and modulation of immune response. Second, the fraction of sites with $\omega_{FG} > 1$ (p_2 in the model) is significantly larger than zero. The MLE of p_2 is 10%, suggesting that a randomly selected site has a 10% chance of having $\omega_{FG} > 1$ along the FG branch. However, further interpretation of this value is limited. First, the MLE of p_2 has estimation errors (95% bootstrap interval: [5%, 23%]). Second, these results provide no indication of which sites might had evolved with $\omega_{FG} > 1$. This is why a second set of empirical Bayes analyses was carried out as part of Basic Protocol 3.

Table 6.15.1 Sites within *Ceacam* with a Posterior Probability > 0.9 for Positive Selection Along the Foreground Branch ($\omega_{FG} > 1$) under Three Different Methods[a]

		SBA	
NEB	BEB	Mean PP	CI
397: 0.99	397: 0.99	397: 0.98	[0.915 , 1.000]
424: 0.97	424: 0.96	424: 0.97	[0.836 , 1.000]
418: 0.96	418: 0.95	418: 0.96	[0.803 , 1.000]
299: 0.96	299: 0.95	299: 0.96	[0.806 , 1.000]
297: 0.96	297: 0.95	297: 0.96	[0.796 , 1.000]
360: 0.91	244: 0.91	360: 0.93	[0.633 , 0.999]
244: 0.90		244: 0.92	[0.576 , 1.000]
		388: 0.90	[0.472 , 0.999]

[a]NEB indicates naïve empirical Bayes. BEB indicates Bayes empirical Bayes. SBA indicates smoothed bootstrap aggregation. The kernel bandwidth for smoothing under SBA was 0.4.

Empirical Bayes (not to be confused with Full Bayes) is a method that passes the MLEs of the ω distribution to Bayes' Rule for the purpose of computing the posterior probability that the data at a given alignment site could have evolved under each of the categories of the model (site classes in Fig. 6.15.1). A site is identified as having evolved with $\omega > 1$ along the FG branch (site classes 2a and 2b in Fig. 6.15.1) if the posterior probability is large according to Bayes' Rule. NEB is a technique for computing these posterior probabilities by passing the MLEs to Bayes Rule without any accommodation of the estimation errors (a technique that is acceptable when the ω distribution is well estimated). Bayes empirical Bayes (BEB) is a technique that is used to obtain posterior probabilities that are corrected for the estimation errors. BEB is not available for all models, but when it is (as with Model A), those results should be preferred over NEB (Yang et al., 2005). Table 6.15.1 shows sites with high a posterior probability for $\omega > 1$ along the FG branch under both NEB and BEB. Note that both NEB and BEB are sensitive to instabilities in the MLEs. A new method, called Smoothed Bootstrap Aggregating (SBA), is now available that is as least as powerful as BEB but is robust to instabilities in the MLE. Unlike BEB, SBA derives its correction for parameter uncertainty from the data in hand. A detailed description of SBA is beyond the scope of this unit; however, Support Protocol 5 describes how to use CODEML_SBA to obtain SBA posterior probability classifiers for each site. SBA has the added benefit of providing output for computing a confidence interval for the posterior probability of a particular evolutionary regime, such as $\omega > 1$ along the FG branch (Table 6.15.1). In the case of *Ceacam*, there is no evidence of instabilities, and the results are largely consistent among all three methods (Table 6.15.1).

COMMENTARY

Background Information

Inference of episodic natural selection requires many stages of data processing and analysis (Fig. 6.15.2), and results should be rigorously evaluated at each of those stages to ensure a reliable outcome. Although rarely done, such assessments should be incorporated into an explicit experimental design. Careful experimental design will be essential to the success of large-scale surveys of gene sequence evolution, which have become commonplace with the advent of low-cost, high-throughput DNA sequencing technologies (e.g., Baker et al., 2016). We strongly suggest that the experimental design of any computational investigation include formal assessment of (i) sequence and alignment quality, (ii) reliability of the statistical framework, and (iii) robustness to the underlying model assumptions.

For the present analysis, both sequence and alignment quality should be rigorously assessed prior to analyses with codon models (the first three stages in Fig. 6.15.2). In

addition to exercising quality control over the primary sequence data, the homology relationships among the sampled sequences must be determined. Unintentional inclusion of paralogs within a dataset can lead to discrepancies between gene trees and species trees (e.g., Mindell and Meyer, 2001), and inappropriate use of a species tree can negatively impact the results, even leading to false conclusions about the history of natural selection pressure (Bielawski, 2013). In addition, the results obtained from codon models can be very sensitive to alignment errors (Fletcher and Yang, 2010; Schneider et al., 2009). We believe that an experimental design should include a mechanism for formal cross checking of alignments. When the number of datasets in a study is not too large, we recommend that every alignment be independently assessed by at least two coauthors, who subsequently compare results to identify and mask regions of alignment uncertainty. For large-scale surveys, where such assessments can be impractical for every gene, we recommend a two-phase screening process (Baker et al., submitted). The idea is to make an initial pass over a large set of alignments obtained by automated methods and identify a subset of candidate genes. In the second phase of analysis the alignments of only those genes that had a significant LRT are visually inspected and masked, and then re-analyzed to verify the significant results. We are encouraged by a recent, fully Bayesian, method for joint assessment of natural selection and alignment uncertainty (Redelings, 2014), which provides yet another tool for controlling the negative impact of alignment uncertainty in such studies.

Critical Parameters

Estimation methods such as maximum likelihood are widely used because they have attractive, and generally reliable, limiting properties. Under the appropriate conditions, maximum likelihood may be assumed to be consistent and efficient, with asymptotic normality of its estimates (Bickel and Kjell, 2015). However, the statistical regularity conditions required for these properties are rarely checked, and failure to meet those conditions can severely impact inferences under the more complex codon models. The task of assessing signal at a small subset of sites for evolutionary change along few, or even one, branch of a tree is very challenging. Moreover, the requirement that the true parameter value is at the interior of the parameter space is often violated under some codon models. This leads to the very real possibility that regularity conditions might not be met with some datasets, and, further, that the resulting instabilities in the MLEs can invalidate the LRTs and negatively bias BEB-based inferences. Thus, we recommend that the properties of the MLEs under complex codon models should be investigated in all datasets via the bootstrap procedure formalized in Basic Protocol 2 of this unit. Indeed, the *NR1D1* example provided for Basic Protocol 2 illustrates that the required regularity conditions will not necessarily be met in all datasets, and that this condition can be diagnosed by assessing the bootstrap estimates of the parameters of the ω distribution.

Robustness refers to the degree to which results obtained under a given method are resistant to errors produced by deviations from the underlying assumptions of the method. Application of a codon model to a real dataset requires a very large number of assumptions, ranging from the fundamental assumption of positional homology (i.e., that the alignment is correct) to the assumption that a specific parametric distribution for ω is appropriate for the data in hand. The CODEML program provides the capability to assess robustness of (i) MLEs, (ii) hypothesis tests, and (iii) empirical Bayes site identification to alternative models for both equilibrium frequencies and the ω distribution. The former type of robustness investigation is the basis of Support Protocol 4, which is directly referenced in Basic Protocol 3. In this setting, Model A is fixed, and alternative models for codon frequencies (Fig. 6.15.8) are employed to re-test a hypothesis and re-identify sites within the alignment. However, CODEML also provides alternative branch-site modes (Model B, Yang and Nielsen, 2002; Zhang et al., 2005), and clade-site models (Models C and D; Bielawski and Yang, 2004), and these permit alternative LRTs for sites subject to a change in the intensity of selection pressure. Thus, a second strategy for assessing robustness via CODEML is to fix the model for equilibrium frequencies and employ alternative ω distributions to re-test the same hypotheses, and to re-identify sites within the same alignment. We recommend that all investigations of selection pressure via CODEML employ both strategies. Further, empirical studies of large groups of genes have revealed that changes in selection intensity can go undetected when only a single family of codon models is used (Schott et al., 2014, Baker et al., submitted).

Suggestions for Further Analysis

The chosen model for codon equilibrium frequency is just one of many fundamental assumptions upon which codon models are built. Rigorous assessment of robustness to the other assumptions is rarely pursued, although relevant computational tools are available via alternative software packages. Among the most critical are assumptions about the absence of (i) gene recombination, (ii) among-site variability in the synonymous rate, (iii) among-site variability in one or more other aspects of the substitution process, and (iv) uncertainty about the most appropriate genealogy for the data in hand. Frequent within-gene recombination can negatively impact some LRTs (Anisimova et al., 2003; Shriner et al., 2003). We suggest screening genes for this using either the GARD-MBP method (Kosakovsky Pond et al., 2006) or GENECOV (Sawyer, 1989), both of which are reasonably powerful and yet robust to a history of positive selection (Bay and Bielawski, 2011). Scheffler et al. (2006) offer a computationally tractable solution for genes having detectable levels of recombination. Among-site variation in any aspect of the baseline rate of DNA/RNA substitution, including d_S, can negatively impact estimates of the ω distribution of some models (Koskovsky Pond and Muse, 2005; Rubinstein et al., 2011). We recommend using a multilayer codon model to formally test for such variation (Rubinstein et al., 2011) as well as assess its potential impact on the posterior distribution for ω (Baker et al., 2016). The multilayer models are only available for codon models where ω varies among sites, but detecting such sensitivity is critical as the branch-site models assume a homogenous baseline rate of DNA/RNA substitution. Variation in other aspects of the substitution process, such as the transition/transversion rate-ratio or codon frequencies, also can negatively impact the inference of natural selection pressure (Bao et al., 2008). For genes where a priori biological knowledge suggests such a possibility (e.g., transmembrane proteins) the program LIBAC can be used to assess its impact on the inference of natural selection pressure (Bao et al., 2008). Lastly, all analyses carried out in CODEML are required to assume that the gene tree topology is known a priori. It is often the case that the true tree is unknown, and there is substantial uncertainty in its estimate. It is always good practice to obtain a set of reasonably good estimates of the tree topology using a program such as RAxML (Stamatakis, 2014) and reanalyze the data under each of those topologies. In a large-scale survey of primate nuclear receptor (NR) genes, Baker et al. (2016) uncovered cases of sensitivity to a wide variety of fundamental assumptions even though the dataset was comprised of relatively closely related, and homogeneously evolving, lineages. We predict that robustness analyses such as these (last stage in Fig. 6.15.2) will have an even greater impact on studies of more divergent lineages.

Within this unit we have presented a set of protocols that are intended to fit within a larger and more comprehensive analytical design. Through automation, the analytical structure depicted in Figure 6.15.2 can be extended to large-scale evolutionary surveys. Such a design implies larger numbers of hypothesis tests, and the single test significance level will no longer provide adequate control over the probability of making one or more type I errors. We recommend adding false discovery rate control (Storey et al., 2002) as a means of inferring which subgroups of genes share a common evolutionary scenario such as episodic evolution. By employing a series of quality-control, statistical-reliability, and robustness analyses within a design, users of codon models can identify problematic cases. We believe that careful experimental design is just as critical to computational biology as it is to traditional experimental biology. However, we recognize that each dataset poses unique challenges, and the increase in understanding that comes from the process of data analysis outlined in Figure 6.15.2 should be used to update the design as required.

Literature Cited

Anisimova, M. and Liberles, D. 2012. Detecting and understanding natural selection. Codon Evolution: Mechanisms and Models, pp. 73-96. Oxford University Press, Oxford.

Anisimova, M., Nielsen, R., and Yang, Z. 2003. Effect of recombination on the accuracy of the likelihood method for detecting positive selection at amino acid sites. *Genetics* 164:1229-1236.[*CE: Please provide DOI Number.]

Aris-Brosou, S. and Bielawski, J.P. 2006. Large-scale analyses of synonymous substitution rates can be sensitive to assumptions about the process of mutation. *Gene* 378:58-64. doi: 10.1016/j.gene.2006.04.024.

Baker, J.L., Dunn, K., Mingrone, J., Wood, B.A., Karpinski, B.A., Sherwood, C.C., Wildman, D.E., Maynard, T.M., Bielawski, J.P. 2016. Functional divergence of the nuclear receptor *NR2C1* as a modulator of pluripotentiality during hominid evolution. *Genetics* (available at http://www.genetics.org/content/early/2016/04/11/genetics.115.183889).

Bao, L., Gu, H., Dunn, K.A., and Bielawski, J.P. 2008. Likelihood-based clustering (LiBaC) for codon models, a method for grouping sites according to similarities in the underlying process of evolution. *Mol. Biol. Evol.* 25:1995-2007. doi: 10.1093/molbev/msn145.

Bay, R.A. and Bielawski, J.P. 2011. Recombination detection under evolutionary scenarios relevant to functional divergence. *J. Mol. Evol.* 73:273-286. doi: 10.1007/s00239-011-9473-0.

Bickel, P.J. and Kjell, A. 2015. Doksum. Mathematical Statistics: Basic Ideas and Selected Topics, volume I. Vol. 117. CRC Press. Boca Raton, Fla.

Bielawski, J.P. 2013. Detecting the signatures of adaptive evolution in protein-coding genes. *Curr. Protoc. Mol. Biol.* 2013:19-1. doi: 10.1002/0471142727.mb1901s101.

Bielawski, J.P. and Yang, Z. 2004. A maximum likelihood method for detecting functional divergence at individual codon sites, with application to gene family evolution." *J. Mol. Evol.* 59:121-132. doi: 10.1007/s00239-004-2597-8.

Bugge A., Feng D., Everett L.J., Briggs E.R., Mullican S.E., Wang F., Jager J., and Lazar M.A. 2012. Rev-erbalpha and Rev-erbbeta coordinately protect the circadian clock and normal metabolic function. *Genes Dev*. 26:657-667. doi: 10.1101/gad.186858.112.

Chen, L., Chen, Z., Baker, K., Halvorsen, E.M., da Cunha, A.P., Flak, M.B., Gerber, G., Huang, Y.H., Hosomi, S., Arthur, J.C., and Dery, K.J., 2012. The short isoform of the CEACAM1 receptor in intestinal t cells regulates mucosal immunity and homeostasis via Tfh cell induction. *Immunity* 37:930-946. doi: 10.1016/j.immuni.2012.07.016.

Desper, R. and Gascuel, O. 2006. Getting a tree fast: Neighbor joining, FastME, and distance-based methods. *Curr. Protoc. Bioinform.* 15:6.3.1-6.3.28.

Felsenstein, J. 2004. Inferring Phylogenies. Sinauer Associates, Sunderland, Massachusetts.

Fletcher, W. and Yang, Z. 2010. The effect of insertions, deletions, and alignment errors on the branch-site test of positive selection. *Mol. Biol. Evol.* 27:2257-2267. doi: 10.1093/molbev/msq115.

Goldman, N. and Yang, Z. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.* 11:725-736.

Gray-Owen, S.D. and Blumberg, R.S. 2006. CEACAM1: Contact-dependent control of immunity. *Nature Rev. Immunol.* 6:433-446. doi: 10.1038/nri1864.

Kosakovsky Pond, S. and Muse, S.V. 2005. Site-to-site variation of synonymous substitution rates. *Mol. Biol. Evol.* 22:2375-2385. doi: 10.1093/molbev/msi232.

Kosakovsky Pond, S.L., Posada, D., Gravenor, M.B., Woelk, C.H., and Frost, S.D.W. 2006. GARD: A genetic algorithm for recombination detection. *Bioinformatics* 22:3096-3098. doi: 10.1093/bioinformatics/btl474.

Mindell, D.P. and Meyer, A. 2001. Homology evolving. *TREE* 16:434-440.

Muse, S.V. and Gaut, B.S. 1994. A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. *Mol. Biol. Evol.* 11:715-724.

Notredame, C. 2010. Computing multiple sequence/structure alignments with the T-Coffee package. *Curr. Protoc. Bioinform.* 29:3.8.1-3.8.25.

Page, R.D. 2003. Introduction to inferring evolutionary relationships. *Curr. Protoc. Bioinform.* 00:6.1.1-6.1.13.

Redelings, B. 2014. Erasing errors due to alignment ambiguity when estimating positive selection. *Mol. Biol. Evol.* 31:1979-1993. doi: 10.1093/molbev/msu174.

Rubinstein, N.D., Doron-Faigenboim, A., Mayrose, I., and Pupko, T. 2011. Evolutionary models accounting for layers of selection in protein-coding genes and their impact on the inference of positive selection. *Mol. Biol. Evol.* 28:3297-3308. doi: 10.1093/molbev/msr162.

Sawyer, S. 1989. Statistical tests for detecting gene conversion. *Mol. Biol. Evol.* 6:526-538.[*CE: Please provide doi number.]

Scheffler, K., Darren P.M., and Seoighe, C. 2006. Robust inference of positive selection from recombining coding sequences. *Bioinformatics* 22:2493-2499. doi: 10.1093/bioinformatics/btl427.

Schmidt, H.A. and von Haeseler, A. 2007. Maximum-likelihood analysis using TREE-PUZZLE. *Curr. Protoc. Bioinform.* 17:6.6.1-6.6.23.

Schneider, A., Souvorov, A., Sabath, N., Landan, G., Gonnet, G.H., and Graur, D. 2009. Estimates of positive Darwinian selection are inflated by errors in sequencing, annotation, and alignment. *Genome Biol. Evol.* 1:114-118. doi: 10.1093/gbe/evp012.

Schott, R.K., Refvik, S.P., Hauser, F.E., López-Fernández, H., and Chang, B.S.W. 2014. Divergent positive selection in rhodopsin from lake and riverine cichlid fishes. *Mol. Biol. Evol.* 31:1149-1165. doi: 10.1093/molbev/msu064.

Shriner, D., Nickle, D.C., Jensen, M.A., and Mullins, J.I. 2003. Potential impact of recombination on sitewise approaches for detecting positive natural selection. *Genet. Res.* 81:115-121. doi: 10.1017/S0016672303006128.

Stamatakis, A. 2014. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* 30:1312-1313. doi: 10.1093/bioinformatics/btu033.

Storey, J.D. 2002. A direct approach to false discovery rates. *J. Roy. Sta.t Soc. B.* 64:479-498. doi: 10.1111/1467-9868.00346.

Suzuki, Y. and Nei, M. 2004. False-positive selection identified by ML-based methods: Examples from the Sig1 gene of the diatom *Thalassiosira weissflogii* and the tax gene of a human T-cell

lymphotropic virus. *Mol. Biol. Evol.* 21:914-921. doi: 10.1093/molbev/msh098.

Wilgenbusch, J. C. and Swofford, D. 2003. Inferring evolutionary trees with PAUP. *Curr. Protoc. Bioinform.* 00:6.4.1-6.4.28.

Yang, Z. 1998. Likelihood ratio tests for detecting positive selection and application to primate lysozyme evolution. *Mol. Biol. Evol.* 15:568-573. doi: 10.1093/oxfordjournals.molbev.a025957.

Yang, Z. 2007. PAML 4: Phylogenetic analysis by maximum likelihood. *Mol. Biol. Evol.* 24:1586-1591. doi: 10.1093/molbev/msm088.

Yang, Z. and Bielawski, J.P. 2000. Statistical methods for detecting molecular adaptation. *TREE* 15:496-503.

Yang, Z. and Dos Reis, M. 2011. Statistical properties of the branch-site test of positive selection. *Mol. Biol. Evol.* 28:1217-1228. doi: 10.1093/molbev/msq303.

Yang, Z. and Nielsen, R. 2002. Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. *Mol. Biol. Evol.* 19:908-917. doi: 10.1093/oxfordjournals.molbev.a004148.

Yang, Z., Wong, W.S.W., and Nielsen, R. 2005. Bayes empirical Bayes inference of amino acid sites under positive selection. *Mol. Biol. Evol.* 22:1107-1118. doi: 10.1093/molbev/msi097.

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

Zhang, J., Nielsen, R., and Yang, Z. 2005. Evaluation of an improved branch-site likelihood method for detecting positive selection at the molecular level. *Mol. Biol. Evol.* 22:2472-2479. doi: 10.1093/molbev/msi237.

Zhang, Y., Fang, B., Emmett, M.J., Damle, M., Sun, Z., Feng, D., Armour, S.M., Remsberg, J.R., Jager, J., Soccio, R.E., Steger, D.J., and Lazar, M.A. 2015. Discrete functions of nuclear receptor Rev-erb-alpha couple metabolism to the clock. *Science* 348:1488-1492. doi: 10.1126/science.aab3021.