Chapter 4

A Beginners Guide to Estimating the Non-synonymous to Synonymous Rate Ratio of all Protein-Coding Genes in a Genome

Daniel C. Jeffares, Bartłomiej Tomiczek, Victor Sojo, and Mario dos Reis

Abstract

The ratio of non-synonymous to synonymous substitutions (dN/dS) is a useful measure of the strength and mode of natural selection acting on protein-coding genes. It is widely used to study patterns of selection on protein genes on a genomic scale—from the small genomes of viruses, bacteria, and parasitic eukaryotes to the largest eukaryotic genomes. In this chapter we describe all the steps necessary to calculate the dN/dS of all the genes using at least two genomes. We include a brief discussion on assigning orthologs, and of codon-aware alignment of orthologs. We then describe how to use the CODEML program of the PAML package for phylogenetic analysis to calculate the dN/dS and how to perform some statistical tests for positive selection. We then outline some methods for interpreting output and describe how one may use this data to make discoveries about the biology of your species. Finally, as a worked example we show all the steps we used to calculate dN/dS for 3,261 orthologs from six *Plasmodium* species, including tests for adaptive evolution (see worked_example.pdf).

Key words dN/dS, CODEML, PAML, Synonymous/non-synonymous rate ratio, Evolutionary rate, Adaptive evolution, *Plasmodium*, Malaria

1 Introduction

1.1 The dN/dS Ratio

With the production of a complete genome sequence a relatively routine task, the bottleneck is now the annotation, analysis, and understanding of this genome data. A particularly useful statistic for protein-coding genes is the ratio of non-synonymous to synonymous substitutions $\omega = dN/dS$ (non-synonymous substitutions are nucleotide changes that alter the protein sequence, synonymous substitutions do not). This ratio measures the strength and mode of natural selection acting on the protein genes, with $\omega > 1$

Electronic supplementary material: The online version of this chapter (doi:10.1007/978-1-4939-1438-8_4) contains supplementary material, which is available to authorized users.

indicating positive (adaptive or diversifying) selection, $\omega = 1$ indicating neutral evolution, and $\omega < 1$ indicating negative (purifying) selection. The ω ratio summarizes the evolutionary rates of genes, and can be an informative feature, because it can identify which genes are the most (or least) conserved and also identify genes that may have undergone periods of adaptive evolution [1]. For parasite genomes, this can help to uncover genes that may be changing rapidly in the "evolutionary arms race" against the host's immune system [2, 3]. There is an extensive literature on the use of ω to study adaptive evolution (see for examples [1–5]).

1.2 Principles of Evolution in Protein-Coding Genes

To understand why ω measures the strength and mode of action of natural selection of genes, let's first consider a new mutation that appears in the genome of a single organism in a population. Over long evolutionary time scales, two outcomes are possible: the mutation may spread throughout the population, until all individuals carry the mutation, that is, the mutation becomes fixed in the population; or the mutation may be lost. The ultimate fate of the mutation (that is, whether it becomes fixed or lost) depends on the interplay between natural selection and random genetic drift. Population genetics classifies mutations as either neutral (having little effect on the organism), deleterious (bad for the organism), or advantageous (good for the organism). Neutral mutations will accumulate in the population at the same rate as the genomic mutation rate μ [6, 7]. On the other hand, deleterious mutations may still reach fixation due to drift, but will accumulate in the population at a slower rate $\mu_-(<\mu)$, while those that are advantageous will accumulate at a faster rate $\mu_+(>\mu)$.

Let's now consider only those mutations that occur at codon positions in protein-coding genes. Synonymous mutations are (mostly) neutral because they do not change the amino acid sequence of the protein encoded, and therefore the synonymous substitution rate will be the neutral rate $\mu_S = \mu$; on the other hand non-synonymous substitutions may be affected by selection and the non-synonymous substitution rate will be in general different to the neutral rate $\mu_N \neq \mu$. Therefore the ratio $\omega = \mu_N/\mu_S$ indicates the mode of selection acting at non-synonymous sites. In practice the rates μ_N and μ_S are not easy to estimate directly. However, the non-synonymous and synonymous distances, $dN = t\,\mu_N$ and $dS = t\,\mu_S$, among orthologous genes in a phylogeny can be estimated from a sequence alignment (with t being the time of divergence or branch length in the phylogeny) leading to the estimation of ω. Sometimes selection may also act at synonymous sites (since some codons may be suboptimal), but this is of main concern for highly expressed genes of fast-growing organisms, since selection on codon usage is in general very weak for most genes in most organism [8, 9]. Methods that explicitly model codon usage selection in the estimation of ω have been developed [10].

For an excellent account of the mathematical theory of ω, the reader can consult Bustamante [7].

Most non-synonymous changes in coding regions negatively alter the structure and function of the protein and are therefore deleterious, whereas most synonymous changes are nearly neutral. This will result in $\omega < 1$ for most genes. When there are strong structural constraints on a protein, purifying selection is strong and there is little or no accumulation of non-synonymous changes, such that the ω approaches zero. In this way, the ω estimate can be used to describe the degree of "selective constraint" (strength of purifying selection) in a gene. This can be a very informative value for describing sets of genes, which can aid in the interpretation of the functioning of the genome [11].

Of course positive selection does occur, if rarely. If positive selection has acted along many of the codons of a gene and throughout the entire phylogeny, then $\omega > 1$. In practice this seldom happens, because positive selection is usually only observed within a specific region of the protein (e.g.: a specific domain) and/or within one branch of the phylogeny (some but not all species). In this case the ω for the entire gene will be shifted (perhaps imperceptibly) towards 1. Models able to detect all these scenarios have been developed [12–16].

In this chapter we limit ourselves to describing how to estimate ω using the CODEML program from the Phylogenetic Analysis by Maximum Likelihood (PAML) package [17]. The CODEML program calculates dN and dS using the observed changes present in a multiple alignment of protein-coding gene sequences from several species in a phylogeny (i.e., given the phylogenetic tree). Statistical estimation of ω with CODEML uses maximum likelihood, employing sophisticated mathematical models to correct for multiple changes, accounting for the different numbers of non-synonymous and synonymous sites, among other complexities, as briefly described in later sections. We describe a few common tests of positive selection. We also describe how to prepare the necessary data for CODEML, that is, how to identify orthologs correctly and build an appropriate sequence alignment, and how to estimate the phylogeny (i.e., the tree topology and branch lengths). We show a real-life example of these methods by examining selection in a set of 3,269 one-to-one orthologs of six *Plasmodium* species.

2 Materials

2.1 Computer Resources

To implement the processes and run the examples described in this chapter you will need access to a computer running a UNIX-like operating system (such as Linux or Mac OS X). Although it is possible to run our examples (Subheading 5) on a typical desktop computer, many CPU hours will be required to process genome-scale data.

In particular, running CODEML for all genes of a genome could take considerable computational time, depending on the number of species and the number of genes in each species. For example, running CODEML on 3,261 *Plasmodium* orthologs took 10 h (average gene length 2.1 kb).

2.2 Software

We provide a list of recommended software in Table 1. We indicate which of these packages will require administration privileges and/or moderate knowledge of Unix to install. The software you will need depends on your data, so we strongly recommend that you read this chapter to the end, including the notes, before installing any necessary packages. All the software we recommend is free of charge for academic use. The essential software will include:

1. BioPerl (to process genome-scale data).
2. An alignment tool (depending on the proximity of your sequences, we recommend Clustal Omega [18] or PRANK$_C$ [19]).

Table 1
Software recommendations

Software	Function	URL	Refs.
<u>BioPerl</u>	Wrappers for automating running code and file I/O	http://www.bioperl.org/	
<u>BLAST</u>+	Ortholog assignment using RBB	Download ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/ Manual http://www.ncbi.nlm.nih.gov/books/NBK1763/	[31]
Clustal omega	Protein alignment	http://www.clustal.org/omega/	[18]
PRANK	Protein alignment	http://www.ebi.ac.uk/goldman-srv/prank/src/prank/	[19, 33]
GUIDANCE	Alignment filtering	http://guidance.tau.ac.il/source.html	[38]
PAL2NAL	Protein-to-nucleotide alignment	http://www.bork.embl.de/pal2nal/	[40]
PAML	Calculating evolutionary rates (ω, etc.)	http://abacus.gene.ucl.ac.uk/software/paml.html	[17]
RAxML	Calculating phylogenetic trees	http://sco.h-its.org/exelixis/software.html	[20]
MACSE	De novo codon-based alignment	http://mbb.univ-montp2.fr/MBB/subsection/softExec.php?soft=macse	[41]
Custom perl scripts developed for this chapter	Various	http://www.danieljeffares.com/data	

Software that may require administrator privileges to install are in **<u>bold underlined</u>** text

3. A package for building phylogenetic trees (we recommend RAxML [20]).
4. The PAML package [17].

2.3 Input Files (Genomes and Annotations)

To calculate ω for all genes in your chosen genomes you will need the following:

1. An annotated genome for the species you are most interested in: This must include accurate protein-coding gene predictions.
2. An annotated genome for *at least* one related species (preferably more): To obtain reasonable sensitivity the additional species must be sufficiently closely related to be accurately aligned (*see* Subheading 3.3 and **Note 1**).

3 Methods

This section describes how to create the necessary files (with various options) for running CODEML and parsing results. The workflow for all these methods is shown in Fig. 1, and an example is provided in the file worked_example.pdf. Some guidance on interpreting results is also included. Throughout this chapter commands will be shown in monotype font, e.g.,

```
perl runscript.pl\
--input myinputfile\
[--parameter 100]\
> myoutputfile
```

Parameters (file names, etc.) that need to be defined by the user are italicized, and optional parameters are placed in square brackets. To display usage information and show what inputs the scripts require, all scripts described here can be run either with no options or using the -h flag (or its longer equivalent `--help`), e.g.,

```
perl runscript.pl -h
```

3.1 Generating (or Collecting) Input Files

For each species to be analyzed, obtain FASTA format sequences of all protein-coding genes, and their corresponding translations. Often, these can simply be downloaded from a variety of websites and servers; we explore this first.

3.1.1 Gathering Gene Sequences from a Database

If it is possible to download the annotation files for some/all of the genes in the genomes of the phylogeny you're analyzing, gathering the list of genes is trivial. We provide a script to extract coding sequences from a Genbank or Embl format file. This script is run like this:

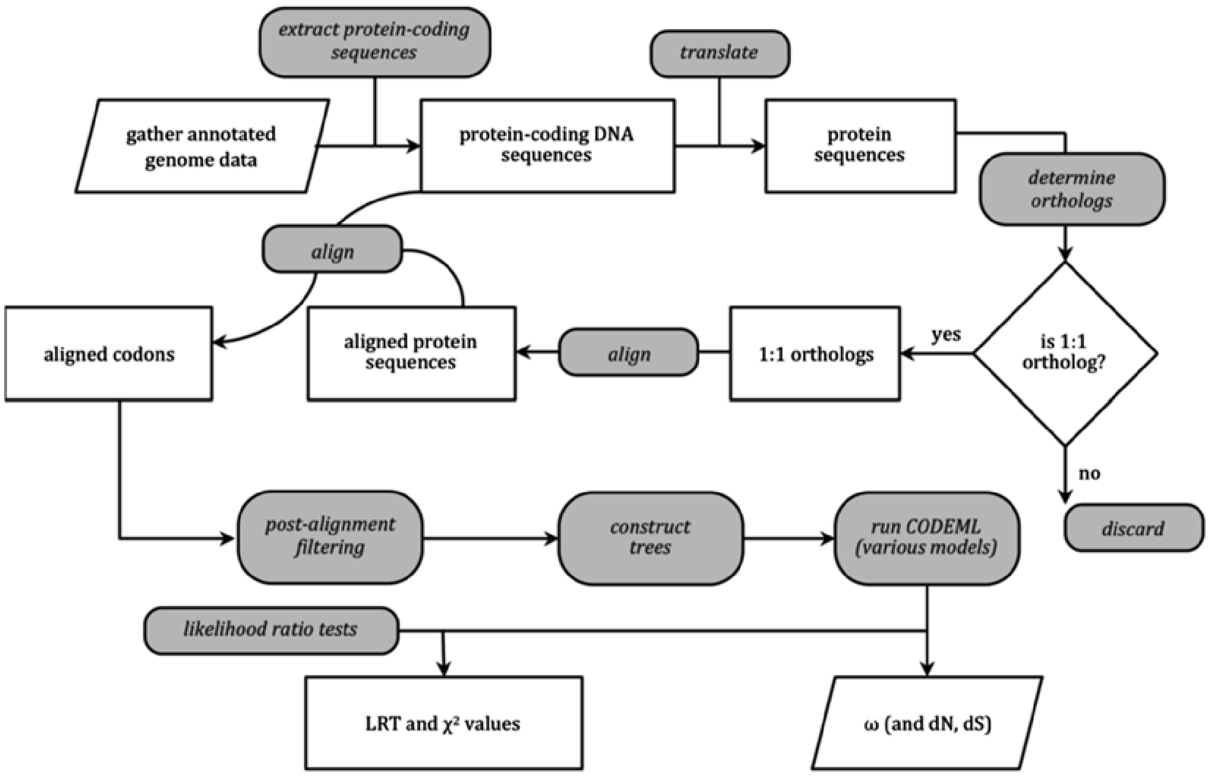

Fig. 1 A flow chart for calculating ω on a genome scale. Once you have obtained protein-coding sequences (DNA and protein), the steps to prepare data for analysis with CODEML will include ortholog assignment, alignment, possible post-alignment filtering, and tree construction. Finally running CODEML will produce ω values, as well as *dN* and *dS*. Running CODEML again with different models of evolution and then conducting likelihood ratio tests will give likelihood ratio test (LRT) values, which can then be tested for significance against χ^2 critical values. See main text for details

```
extract_genes_from_genome.pl\
-I "input_file1,input_file2, input_file3, etc"\
-s "species1,species2,species3"\
-t tag\
[-f genbank/embl (embl 175 default)]
```

The input_files will be Genbank or Embl format files, which contain both sequences and the start and end positions of all protein-coding exons. Species_name is used merely to name the output file, and the tag is the delimiter for gene names in the input file. This tag will differ depending on the species/input files. "systematic_id" is one example. We advise that you look into the Genbank or Embl file to determine this.

3.1.2 Generating Input Files from Contigs/ Chromosomes and Corresponding Genome Annotations

Alternatively, it's possible to generate the necessary DNA and protein files from chromosome sequences or contigs (in FASTA format) and corresponding annotation files (in standard GFF format). We provide a script to gather all coding sequence (CDS) fragments from the genome and join together those corresponding to the same gene in the correct order, using gene coordinates from the GFF file. Protein sequences are then produced by translating these CDS sequences using BioPerl. If provided, the script will also compare its own translations to a set of corresponding protein sequences from an initial genome annotation and ignore those that do not match, or print a warning and take the translation. This script is run as follows:

```
perl get_cds_prot_from_gff_cont.pl -i input_folder -o output_folder -g annotations.gff -d contigs.fasta [-l list_of_desired_ids.csv]
```

The optional `-i` flag takes the address of a folder where the input files are to be found, and analogously for `-o` and the output files. `-g` is required; it takes an annotation file in GFF format (version 3 by default, although this can be changed). `-d` is also required, and it takes a DNA sequence file in which the sections that contain the genes specified in the annotations can be found. By using `-l` you can specify a list of desired IDs from the GFF file to process; if you don't provide such list, the script will simply process all the genes in the GFF file. You can additionally specify a file containing all protein sequences by using `-p`. As with all our scripts, a full list of options can be obtained by running it with no options specified, or using the `-h` (or `--help`) flag. The final output of this script consists of a pair of files for each gene, one for the combined CDS DNA sequences, and the other for the protein sequence, both identified by the same gene ID (`<gene_id>.dna.fasta` and `<gene_id>.prot.fasta`, respectively).

3.2 Identifying Orthologs

Once you have obtained the protein translations of each gene for each genome, the next step is grouping genes into sets of orthologs. These orthologous groups of genes will then be aligned and used as input for CODEML. "Orthologs" are homologous genes that were separated by speciation while "paralogs" are homologous genes that were separated by gene duplication [21]. For a review of the principles and complexities of orthology *see* ref. 22.

Many tools have been created to assign orthologs (for reviews, *see* refs. 23–26). The three main approaches use sequence similarity with graph clustering, phylogenetic trees, synteny, or a combination of several methods. The differences between the performance of the widely used methods on the same data appear to be fairly small—the major factor is the complexity of the proteomes involved (number of proteins, number and complexity of gene duplications, extent of multi-domain proteins, and domain-shuffling) [25–27]. It is important to appreciate that all orthology prediction methods will contain errors. These should be identified and removed where possible. We provide guidelines at later stages to account for these. The most common approach is to remove all orthology groups that have more than one ortholog per species (retaining only 1:1

Table 2
Useful websites

Contents/topic	URL
Guide to using BioPerl modules to run PAML and parse output	http://BioPerl.org/wiki/HOWTO:PAML
PAML discussion group	http://www.ucl.ac.uk/discussions/viewforum.php?f=54
Database of ortholog groups of eukaryotic proteins using InParanoid	http://inparanoid.sbc.su.se/
Database of ortholog groups of proteins using OrthoMCL	http://www.orthomcl.org
Clusters of orthologous groups of proteins from whole genomes	http://www.ncbi.nlm.nih.gov/COG/
OMA (Orthologous MAtrix) database of orthologs for complete genomes	http://omabrowser.org/
Treefam (tree families database)	http://www.treefam.org/
Gene ontology	http://www.geneontology.org/
A list of orthology databases	http://questfororthologs.org/orthology_databases

orthologs). This will remove the complication of paralogs that are due to gene duplications.

3.2.1 Obtaining Predefined Orthologs for the Species in Your Phylogeny

If a manually curated set of orthologs has been produced as part of a genome project, particularly any that use synteny, we encourage you to use these. Another alternative is that for published genomes, ortholog assignments may have been produced in one or more orthology prediction databases, such as OMA or OrthoMCL (*see* Table 2).

3.2.2 Reciprocal Best BLAST Hit Method to Assign Orthologs

If ortholog lists are not available we recommend the Reciprocal Best BLAST hit (RBB) method (with simple clustering for multiple species). This method is simple, fast, scalable, arguably as accurate as tree-based methods, and does not need extensive parameter optimization [27]. However, *see* ref. 28 for advice about BLAST options.

The (RBB) method assigns two proteins (genes) as orthologs if protein A from species 1 identifies protein A’ from species 2 as its best hit, and vice versa [29, 30]. This requires that you run a BLAST search for each protein against each other genome in turn. The recommended BLAST parameters for RBB are a minimum BLASTP Evalue $\leq 1e^{-5}$ or $\leq 1e^{-6}$, and the combination of soft filtering with a Smith–Waterman final alignment (the -F “m S” -sT options in NCBI’s BLASTP) [27, 28].

The pairwise RBB method may be extended into a clustering algorithm (cRBB) so that it can be applied to more than two genomes as follows [27]. For the genes A,B,C (from species *a, b, c*) to be clustered into an orthologous group gene B must be the reciprocal best hit (RBH) to gene A, and gene C must be the RBH to *either* gene A or gene B. If not, then gene C is not included in the ortholog group.
To determine cRBB you will require:

(a) FASTA format files of all protein sequences of all species considered (one file per species).
(b) An installation of BLAST (we advise BLAST+) [31].
(c) Wrapper script(s) to run BLAST searches and cluster best hits.

We provide a script to produce a list of orthologs from a full analysis of a set of genomes using this RBB approach. Note that this script is likely to take a few hours to run, due to the many BLAST searches performed:

```
perl prthologs_from_RBBH.pl -o orthologs.csv -i
"all_proteins_species_1.fasta,all_proteins_
species_2.fasta,all_proteins_species_3.fasta"
```

Multiple files can be specified in the usual UNIX way (e.g., `"*spec*.fasta"`). This script runs a Reciprocal Best BLAST hits method and produces a set of orthologous gene IDs separated by a comma and indexed by the first species. The output file consists of a list of such comma-separated sets. In the example above only lists with orthologs in all three species will be returned, but adding `-n 2` would return all sets with two or more orthologs.

3.3 Alignment of Orthologs

Alignments can be an important source of false-positive cases (incorrectly inferred adaptive evolution) [32–34], so the choice of an alignment tool is an important consideration. A particular concern in the context of this chapter is that alignment quality decreases with decreasing protein similarity [18]. So this places a limit on the degree of divergence that can be used in calculations of evolutionary rates, particularly when false positives are a concern (such as when the aim is to detect the few genes that are subject to positive selection). In general, reliable alignments can be produced with Clustal Omega (and several other tools) with proteins that have at least 70 % identity [18]. In this section we give an overview of the typical steps you will need to follow if you're writing your own scripts, followed by instructions for running a script we provide that integrates all the relevant tasks. Finally, we provide an alternative script when reliable translations of your coding sequences (CDS) are not available.

3.3.1 Considerations for Choosing a Multiple Alignment Tool

There are many alternative tools for aligning multiple protein sequences. At the present time, two of the most reliable programs are Clustal Omega [18] and PRANK [19]. Clustal Omega appears to be the most powerful (fast, accurate) for divergent proteins, whereas PRANK performs well on more closely related sequences. PRANK has been explicitly designed for evolutionary analysis and performs well under simulation [33, 35]. We do not advise using the older ClustalW which is an entirely different program to the newer Clustal Omega.

With any level of divergence that provides sufficient power to detect adaptive evolution, alignments will contain errors [18] that cause false positives and false negatives [33]. In particular insertions and deletions are a major source of false positives in the detection of adaptive evolution using CODEML [35]. Using the *cleandata* option in CODEML may reduce the false positives, but at the possible loss of interesting sites. There are various filtering methods available to remove potentially unreliable alignment columns or codons [36–38]. The use of these is equivocal; they appear to improve evolutionary analysis in some cases [39], but have negligible effects in others [33]. In practice, alignment filtering will produce a more conservative analysis—lowering both false positives and true positives. It is difficult to generalize about the cost/benefit of such approaches. For a detailed analysis, we recommend comparing results produced from filtered and unfiltered alignments. Based on two simulation studies [33, 39], we recommend the GUIDANCE tool for alignment filtering [38].

Programs that calculate dN/dS will require codon-based alignments of the DNA sequences of all genes in each ortholog group; therefore gaps should be positioned so as not to change the reading frame. If you aligned the CD sequences with PRANK using the translate option or using the empirical codon models you will already have codon-based DNA alignments. On the other hand, if your alignment program generated amino acid alignments, you will need to perform "reverse translation" to construct a codon alignment from the unaligned DNA sequence files and the aligned amino acid sequence files. This is best achieved by aligning the corresponding protein sequences, and then converting the protein alignment to a nucleotide alignment using the corresponding gene sequences. We recommend the conducting analysis tool for this task [40]. This software produces a codon alignment with options for removing gaps and in frame stop codons, as well as mismatched codons. The "native" format for the CODEML program is the PHYLIP format, with some small modifications. We suggest that you refer to the PAML manual before constructing DNA alignments (*see* Table 1).

In summary, care must be taken to obtain the best alignment, and we recommend particular care and skepticism for this stage in the analysis. This is particularly important when conducting analysis on a genome scale, because a few false positives could dominate any signal for adaptive evolution, or skew the ω estimate with a systematic bias to particular types of genes.

3.3.2 Producing Codon-Specific Alignments from an Ortholog List

We provide a script to automate the tasks described above for a genome-scale list of sets of orthologs. To run it, you will need:

1. A list of sets of orthologs in a comma-separated value (CSV) file, where each line has a set of related orthologous gene IDs separated by a comma (*see* **Note 2** for an example of the CSV file, and comments).
2. DNA and protein sequences for each desired gene, in FASTA format, identified by the same gene ID indicated in the orthologs CSV file, and named `<gene_id>.dna.fasta` and `<gene_id>.prot.fasta`.
3. Run the script:

```
perl align_orthologs.pl -l orthologs.csv -i
input_folder -o output_folder -c -a
```

The `-l` option receives a list of orthologs in a CSV file, as described above. `-i` receives the location of the folder where your DNA and protein sequences reside, and you can also specify the folder where you want to put your output files via the `-o` option. `-c` tells the script to do the protein alignment using Clustal Omega, and -a indicates that you want to calculate PAL2NAL codon alignments. The script will print out warnings for any input files it cannot find, and it will only produce alignment files if it can find two or more of the orthologs in each set (i.e., each line of the CSV). You can send the list of any missing information to a file by adding the `-e` flag.

It is also possible in theory to align protein-coding sequences when reading frames are not known, using software such as MACSE [41] (*see* **Note 3**). If the goal of the analysis is to identify genes that are subject to positive selection, we advise caution when using such methods, because alignment inaccuracies increase the rate of false-positive results, as well increase (the already abundant) false negatives [35].

3.4 Estimating Phylogenetic Trees

Model testing with CODEML requires a phylogenetic tree of either the group of species or the gene concerned. When using CODEML "site tests" for positive selection are robust to tree topology, so in general the species tree should be used. This is the most common case (*see* below Subheading 3.5). If a species tree is not available, we recommend that you estimate one tree for the species you will analyze using a concatenation of all the aligned orthologs (see below). This should increase the power to detect cases where particular branches of the phylogeny have an increased evolutionary rate in a few orthologous groups. An exception is that in the unlikely scenario there is evidence for recombination between your species (i.e., if they are strains within a species or very recently separated species), then you may wish to estimate topologies for each gene. A concatenation of all aligned CDS sequences can be achieved with the concatenate_alignments.pl script that we provide (*see* Table 5).

There are many tools to estimate phylogenetic trees based on sequence data. We recommend the RAxML tool, which is fast and sufficiently accurate [20], run using the GTR gamma model. To obtain a phylogeny for ω estimation the command line required to run RAxML is

```
raxmlHPC -f a -x 12345 -p 12345 -# 100 -m GTRGAMMA
-s your_alignment_file.phy -n your_alignment_
prefix
```

3.5 Using CODEML to Calculate ω and Identify Positive Selection

3.5.1 Concepts in CODEML

PAML is a package of programs designed to analyze molecular sequences and estimate a variety of parameters of molecular evolution [17]. We concern ourselves here only with estimates of ω, and attempts to detect positive selection using the CODEML application in PAML. For more advice the PAML FAQ and PAML manual will be helpful, as will the PAML discussion group (*see* Table 2). The statistical theory of adaptive evolution is reviewed in [42]. We also recommend these more technical articles for further reading [15, 17, 32, 43]. There are of course other tools for calculating non-synonymous and synonymous evolutionary rates apart from CODEML. We recommend HyPhy, a particularly versatile tool for testing models of evolution [44]. While HyPhy allows the user to specify virtually any model for evolution, some expertise is needed to do this because this tool has its own batch language. The sitewise likelihood-ratio (SLR) software package is another alternative [45]. The SLR method makes less assumptions about how the strength of selection is distributed across sites and is considered complementary to PAML. As with PAML, the HyPhy and SLR tools are all in active development (as of 2012). We do not describe how to use HyPhy or SLR in this chapter.

Adaptive evolution seldom occurs in all species of a phylogeny, or over all sites in a gene, which makes it more difficult to locate the genes/sites concerned. The more likely scenario is that positive selection has occurred in some branches of the phylogeny, or in some specific sites in the gene or only in specific sites in some branches of the phylogeny. Each of these possibilities is formalized into a “model,” so that possible processes of evolution can be tested for explicitly. The main classes of models used in CODEML are “branch models” (where ω can vary over different branches in the phylogeny), “site models” (where ω can vary at different sites in the gene), and “branch-site” models (where ω can vary in particular sites, in particular branches). In tests for adaptive evolution that use branch models, positive selection is detected along

Table 3
Models of adaptive evolution implemented in CODEML[a]

Model	Description
Site models	
M0 (one ratio)	One average ω for the gene Null model for testing if selected branches evolve with different rate than the background branches Specify using NSsites = 0, model = 0
M1a (nearly neutral)	One ω across all lineages, models only two classes of sites ($0 \leq \omega < 1$ and $\omega = 1$) Specify using NSsites = 1, model = 0
M2a (positive selection)	One ω across all lineages, models three classes of sites ($0 <= \omega < 1$, $\omega = 1$, and $\omega > 1$) Specify using NSsites = 2, model = 0
M7 (beta)	One ω across all lineages, ten classes of sites with $\omega <= 1$ Specify using NSsites = 7, model = 0
M8 (beta and ω)	One ω across all lineages, 11 classes of sites on all lineages, 10 with $\omega \leq 1$, 1 with $\omega > 1$ Specify using NSsites = 8, model = 0
Branch models	
Free-ratio model	Allows different ω for each branch of the tree. Specify using NSsites = 0, model = 1
Two-ratio model	Allows several ω values for a specified branch (the "foreground" branch, usually your species of interest). The user must specify which this "foreground" branch, and the other "background" branches Specify using NSsites = 0, model = 2
Branch-site models[b]	
Model A	Like site M1a, M2 site model, but marked branches are treated as foreground allowing three classes of sites ($0 < \omega < 1$, $\omega = 1$, $\omega = >1$), and others, as background with only two classes of sites ($\omega = 0$, $\omega = 1$) Specify using NSsites = 2, model = 2, fixomega = 0
Model A1	Null model, foreground branches allowing two classes of sites ($0 < \omega < 1$, $\omega = 1$), and others, as background with only two classes of sites ($0 < \omega < 1$, $\omega = 1$) Specify using NSsites = 2, model = 2, fixomega = 1

[a]In all these models $\omega < 1$ indicates purifying selection, $\omega \leq 1$ indicates selection in a purifying to nearly neutral range, $\omega = 1$ indicates neutral evolution, and $\omega > 1$ indicates adaptive evolution. The ω values can refer either to the entire gene or to some sites (codons) within a gene. In some cases the models allow for adaptive evolution ($\omega > 1$) in some sites within one branch of the tremanue ("site-branch" models)
[b]See the PAML manual (Version 4.6, March 2012) for how to direct CODEML to use these models. Note that Model B and Site model 3 are no longer recommended

the branches only if the average ω over all codons in the gene is larger than one. This is unlikely to occur, because even if a few sites in the protein are evolving fast along the branch the average ω may not be >1, because most of the sites in the protein will remain under purifying selection. However some authors managed to get positive results using this approach to detect adaptive evolution [4, 46]. A more realistic model is site models, which allow ω to vary only within specific sites of the gene, but for all species. Finally, branch-site models allow ω to vary both among sites and across the branches of the phylogeny. These are probably the most realistic models. The models that are currently recommended to test these alternatives are described below (*see* also Table 3 for summary of the models used in CODEML, and how to direct CODEML to use these models).

1. The *one-ratio model* (M0 in CODEML) calculates the average ω for the whole gene, over all branches in the phylogeny. This is useful to obtain the average ω value for the gene, but is not thought to be a sufficiently realistic model to detect adaptive evolution.
2. The *Nearly Neutral model* (M1a) classifies codon sites in a gene into two groups: one group has codons subjected to purifying selection ($\omega < 1$), and the other group has codons under neutral evolution ($\omega = 1$). There are no codons under positive selection ($\omega > 1$).
3. The *Positive Selection model* (M2a) as the NearlyNeutral model, but an extra class of codon sites subjected to positive selection ($\omega > 1$) is allowed.
4. The *beta model* (M7) uses the flexible beta distribution to describe ω variation among sites. The distribution of ω values can take a variety of shapes in the range from 0 to 1, so codons under positive selection are not allowed.
5. The *beta and ω model* (M8) is the same as the beta model, except that it allows for some sites to be subjected to positive selection ($\omega > 1$).

The application of maximum likelihood in CODEML allows these models of evolution to be described as mathematical summaries of the stochastic process of molecular evolution. CODEML uses a maximum likelihood approach to attempt to fit the observed data (the sequence alignment) to the model of evolution that you specify. This involves estimating parameters such as the branch lengths, the transition/transversion ratio, and the ω ratio (see the PAML manual for details). Once this is done CODEML provides the parameters that are its best fit to the data and a *likelihood value*. This *likelihood value* (L) is the probability of observing the data with parameters generated by the model. Likelihood values are provided in the natural log, $\ln L$ (*see* **Note 4** for further details).

To determine if positive selection has occurred in a gene, you will need to show that a model that includes positive selection (where $\omega > 1$ in some sites) fits the data better than one that does not include positive selection (i.e., no $\omega > 1$ sites). This is achieved as follows:

1. Run CODEML with a simple model that does not allow positive selection. CODEML will estimate ω and determine the $\ln L$ for each gene with this model (l_0).
2. Run CODEML with a more general model that allows positive selection. CODEML will estimate ω and determine the $\ln L$ for each gene with this model (l_1).
3. Determine which model is more likely for each gene using a likelihood ratio test (LRT, *see* **Note 4**). The LRT statistic = $2 \times (l_1 - l_0)$
4. You may reject the simpler model for any particular gene if the LRT statistic is greater than the critical χ^2 value with k degrees of freedom (*see* **Note 4**, and example in supplementary file worked_example.pdf).

We describe in the next sections a general schema for how to do this in practice. We also provide a detailed step-by-step example of this process in the supplementary file worked_example.pdf. This example shows how we calculated ω for all the 1:1 orthologs in six *Plasmodium* species, and detected some statistically supported cases of adaptive evolution.

3.5.2 Estimating ω for All Genes Using the Simple One Ratio Model

Once you have a codon-specific alignment and a phylogenetic tree, the next step is to run CODEML with a null model (usually model M0). The models are specified in the CODEML control file (usually with a .ctl extension). The control file also specifies which sequence file, the tree file, and other parameters that CODEML should use. A detailed explanation of this file and all the options available is given in the PAML manual, and we provide an example CODEML-M0.ctl in supplementary material. The most important parameters to note are:

```
seqfile = myfile.paml
```

The sequence alignment file (containing all gene alignments).

```
treefile = tree.txt
```

The plain text file containing the phylogenetic tree of the species, in Newick format.

```
outfile = M0-output.txt
```

The name of the output file.

```
ndata = N
```

Where *N* is the number of alignments to be analyzed.

```
CodonFreq = 2
```

Which specifies which positions to use to calculate the nucleotide frequencies.

```
model = 0
```

This specifies whether to allow ω to vary among lineages in the phylogeny.

```
NSsites = 0
```

Specifies whether the model CODEML uses ω to vary among sites of a gene.

Once you have edited your control file, you should run CODEML:

```
codeml codeml-M0.ctl
```

Expect CODEML to run for many hours (our example of 3,261 *Plasmodium* orthologs took 10 h). The output will be contained in the file you specified (M0-output.txt above). It is simple to extract the ω values from CODEML's output file with grep:

```
grep omega M0-output.txt > M0-omega.txt
```

3.5.3 Estimating ω and the lnL for All Genes Using Alternative Models

To evaluate whether the data for a particular gene fits an alternative evolutionary model you will need to run CODEML again, specifying another model. This is done by modifying the control file (saving it with a new name), sometimes modifying the tree file, and running CODEML again. The most common use for this is to examine whether each gene better fits a model that includes *some sites* that have adaptive evolution (where $\omega > 1$). Remember that it is unlikely that the average ω for the *entire gene* will be >1. Once this is done CODEML will produce a log likelihood estimate (lnL), which you can use for likelihood ratio tests. To determine

Table 4
Recommended tests of selection in CODEML

Models		***k***	**Hypothesis tested**
Site models:	M2a vs. M1a	2[a]	Does adding a third class of sites with $\omega > 1$ (adaptive evolution) fit the data better than a model with two classes $\omega < 1$, $\omega = 1$?
	M8 vs. M7	2[a]	Does adding an extra class of sites with $\omega > 1$ (adaptive evolution) fit the data better than a model with ten classes with flexible normalized non-synonymous ratio distribution?
Branch models:	*Free-ratio* vs. *one-ratio* model	$2s$–4	For a tree of *s* species, is ω different among lineages?
	Two-ratio vs. *one-ratio* model	1	Are the foreground branches that you specify more likely to have different ω from background branches?
Branch-site models:	MA($\omega > 1$) vs. MA($\omega = 1$)	1[b]	Is the defined "foreground branch" more likely to contain sites with $\omega > 1$

[a]In these models the regularity conditions are not met and the asymptotic distribution of the LRT statistic is not known. Using χ^2 with the given *k* degrees of freedom possible makes the test conservative (Yang and dos Reis [32])

[b]In the branch-site test, the asymptotic distribution of the LRT statistic is a 1:1 mixture of point mass zero and χ^2 with $k = 1$ (Yang and dos Reis [32])

which model to use as the null and alternatives, consult Table 4. For example, comparing the model M2a (which allows some sites to have $\omega > 1$) against the null model M1a (which doesn't allow this) examines whether the data better fits a model with some adaptive evolution. See below for more detail about the likelihood ratio tests.

1. *To specify a site model*: Modify your control file to include these lines (deleting the previous settings). The "NSsites = 0 1 2 7 8" text instructs CODEML to determine the lnL with several models. Note that site models allow you to predict which sites have been subject to selection.

   ```
   model = 0
   NSsites = 0 1 2 7 8
   outfile = site-models-output.txt
   ```

2. *To specify a branch model*: Modify your tree file to mark the branch that you wish to test. This is done by adding a hash tag (e.g., "#1") to the branch: e.g.:

   ```
   (((2,(3, 1)),6 #1),5,4)
   ```

   For clarity, save your tree file with a new name. Then modify your control file to include these lines:

   ```
   treefile = marked-tree.txt
   model = 2
   NSsites = 0
   outfile = branch-model-output.txt
   ```

3. *To specify a "branch-site" model*: Modify your control file to include these lines:

   ```
   treefile = marked-tree.txt
   model = 2
   NSsites = 2
   outfile = branch-site-model-output.txt
   fix_omega = 1
   omega = 1
   ```

3.5.4 Likelihood Ratio Tests (LRT) of Positive Selection

Once you have run CODEML with a null model and an alternative model, you will then use a likelihood ratio test to see if the data are a significantly better fit to the alternative model (*see* Table 3 or models and Table 4 for which null and alternative models to test). Note that adaptive evolution is usually rare in genomes, so the no-selection model is usually the null. The steps to take to perform a likelihood ratio test are the following:

1. A log likelihood ($\ln L$) value for each gene has been calculated by CODEML for a null and alternative model.
2. Calculate the value of the LRT statistic (twice the difference of the log-likelihood between the null model and alternative model).
3. Determine the degrees of freedom (k) for your test. This is calculated as $k = p_1 - p_0$, where p_1 is the number of parameters estimated in the alternative model, and p_0 is the number of parameters in the null model. For simplicity, we list the degrees of freedom in Table 4.
4. Compare the LRT statistic with the critical value from χ^2 distribution, with the appropriate degrees of freedom (k) and the significance level that you want (α), which is usually 0.05 or 0.01.
5. (a) If the value of the LRT statistic is greater than the critical value, you reject the null hypothesis, which means that there are sites (or branches, depending on the test) that have undergone adaptive evolution.

   (b) Alternatively, the p-value can be calculated from the cumulative distribution function of the χ^2 statistic where appropriate (*see* **Note 5**). These models are described in the following references [14, 47, 48].

Table 5
Scripts we provide with this chapter

Script	Function
`genes_from_genome.pl`	Extracts CDS sequences and proteins from a Genbank or Embl file
`gff_cds_proteins_processor.pl`	Extracts CDS sequences from a FASTA nucleotide file according to GFF coordinates, translates
`orthologs_from_RBBH.pl`	Assigns orthologs using the clustered Reciprocal Best Blast (cRBB) approach
`align_orthologs.pl`	Aligns proteins, generates codon-aware nucleotide alignment
`multiple_sequence_splitter.pl`	Splits a FASTA file with many sequences into one file per sequence
`concatenate_alignments.pl`	Takes a list of alignment files and outputs a single concatenated alignment file
`codeml_simple.pl`	Runs CODEML for all genes in a list
`codeml_site_models.pl`	Runs CODEML for all genes in a list. Performs the likelihood ratio tests for site models
`codeml_branch_models.pl`	Runs CODEML for all genes in a list. Performs the likelihood ratio tests for branch models

When testing for adaptive evolution on thousands of genes, a method to correct for multiple testing is desirable. We recommend using the false discovery rate approach described by Benjamini et al. [49].

For large sets of data you can perform all CODEML calculations and the tests using our perl wrapper scripts (*see* Table 5).

3.5.5 Detecting Particular Sites That Have Been Subject to Selection

Genes that were identified as containing sites under selection can be investigated further to determine the probability that each codon has been subject to adaptive evolution ($\omega > 1$). CODEML will already have performed a Bayesian identification of these sites (as described in [50, 51], which is presented in the main output file (e.g., branch-site-model-output.txt above). We provide more detail about how to examine this output in the supplementary data file worked_example.pdf.

4 Interpreting Results on a Genome Scale

A genome-scale evolutionary analysis of protein-coding genes can be very useful for describing features of the genome. Two approaches to describing genomes using evolutionary values are (a) to plot and correlate evolutionary parameters (ω, etc.) with other quantitative features of genes (e.g., expression levels) and (b) to group genes by various methods (e.g., Gene Ontology) and then look for groups of gene with significantly higher/lower evolutionary parameters.

4.1 Correlating Evolutionary Parameters with Other Features of Genes

It is most often found that ω correlates with the expression "breadth" (the number of tissues it is present in) or expression level of a gene, for example [11], but other correlating features of genes with ω (or dN or dS) could also reveal new features of genomes. It is important to appreciate in these analyses that many aspects of genes are correlated [52], so further analysis will be required to determine which aspect(s) of the gene causes the correlation [53, 54]. A balanced analysis should take into account that statistically significant p-values can be obtained with large data sets, even when the strength of the effect is very weak (i.e., high p-value, but low correlation coefficient *rho*). Plotting data and reporting only the strongest effects will help to distinguish biologically meaningful results from those that are very weak effects that produce very statistically significant p-values merely because of the large number of observations.

4.2 Comparing Evolutionary Parameters Between Group of Genes

There are a variety of ways to group genes that can be revealing. The use of Gene Ontology (GO) is common [1, 11, 55]. Within gene ontology both biological function, biochemical function, and the cellular location aspects can be revealing. The PANTHER software system for inferring the functions of genes based on their evolutionary relationships [56] is another alternative. Clustering genes by their similarity of expression or by principal tissue (or life cycle stage for parasites) they are expressed in can also reveal salient patterns [11, 57]. Genetic or protein interaction maps can also be used to group genes.

Once genes have been grouped, the approach is to show the extent to which different groups of genes differ in their evolutionary features by comparing evolutionary parameters (such as ω, dN, dS, the LRT statistic, or the p-value from LRT tests) between groups of genes. Simply sorting groups by their median values and plotting can be sufficient, for example [5]. To test whether specific groups of genes have more/less constraint a Mann–Whitney test is most often used because it is nonparametric (does not assume that the data have any particular distribution, e.g., normal distribution). In this case one might test whether the genes in a particular group have a different distribution to another group, or differs to all other genes. To locate particular groups of genes that are evolving adaptively, then the likelihood ratio test (LRT) statistics of the genes can be compared with Mann–Whitney tests. Another alternative is to count the number of genes in a group that pass a meaningful LRT significance value, and use a Fisher's exact test to determine if the group is enriched for positively selected genes.

Regardless of the methods used it is important to use a method to correct for multiple comparisons. The Bonferonni correction is in common use, but other methods are available [58, 59]. Finally, we suggest some healthy scepticism about genes that appear to have undergone adaptive evolution (e.g., high ω). If possible manual checks of the alignments and orthology may aid in rejecting false positives.

5 Final Comments

The methods we have described should enable you to calculate ω and detect possible cases of adaptive evolution for all the genes in your genome. We advise care with all steps, particularly collecting sufficient data to have good power (more genomes is better), alignments, and CODEML model testing. Keep in mind that ω is not the only test for non-neutral evolution; some other methods are described in [60], which may require different data types. The methods described are, to the best of our knowledge, up to date when this chapter was written. However, things change, so we encourage readers to post comments on CiteULike at www.citeulike.org/user/danieljeffares/publications.

Supplementary data will be available on http://www.danieljeffares.com/data.

6 Notes

1. To calculate *dN*/*dS* accurately from alignments of orthologous genes the sequences must not be too closely related, or too distant. If sequences are too closely related (e.g., all sequences >95 % identity on the DNA level) there will be little power to accurately estimate *dN* and *dS*, since there will be too few observed changes. Because CODEML (and other tools) estimate parameters such as these from the observed genetic changes, the power of the analysis increases when there are more changes observed. Increased power can be attained in two ways. First, by choosing species that are sufficiently divergent. This approach is helpful up to the point where orthologs cannot be assigned correctly or DNA mutations (substitutions) at fast-evolving sites are saturated. The PAML FAQ also states that the method is reasonable if the synonymous distance over all branches of the tree is >0.5; this approximate figure is supported by simulations [33]. In practice, this means that when looking at an alignment of a protein-coding gene most synonymous sites have a change in one or more of the species. Secondly, the power increases with increasing number of orthologs (species) in the alignment. The PAML FAQ recommends that the absolute minimum number of species is 4 or 5 and that 10 is good, but 20 would be better. Simulations show that good estimates of ω can be obtained with six species, while detection of adaptive evolution has relatively low power with this many taxa [33]. In practice of course, it is nontrivial to add another genome to your analysis after data have been gathered for a project, but it is an important consideration if accurate and sensitive evolutionary analysis is a desired outcome.
2. The ortholog list file that the `ortholog_processor_aligner.pl` script requires should be in this format:

```
GENE_1_SPECIES1,GENE1_SPECIES2,GENE1_SPECIES3
GENE_2_SPECIES1,GENE2_SPECIES2,GENE2_SPECIES3
```

   Since the scripts use each gene ID to find corresponding files, sequences, and annotations, it is crucial to use exactly the same spelling all across (however, note that our scripts can get rid of most non-word characters like "_" or "#"). In case you already have files containing the sets of orthologous DNA sequences, provide a list with only one ID per line, corresponding to the base name of each of the files that contain the orthologs, which in turn should be named `<gene_id>.orthologs.dna.fasta` and, optionally, `<gene_id>.orthologs.prot.fasta`

3. When annotations of protein-coding sequences are unreliable or absent it is possible to produce “de novo” codon-based nucleotide alignments with packages such as MACSE [41]. This software can generate multiple-sequence alignments accounting for disruptions in the reading frame (stop codons or frame shifts arising either from sequencing errors or biological deviations) without knowing the reading frame, or any corresponding amino acid translation, in advance. MACSE recognizes the reading frame and produces the alignment in the FASTA format. Any possible frameshifts and stop codons are detected and the nucleotides are aligned in a way that any alignment gaps are more likely inserted as a multiplication of three. This results in higher quality of the codon-specific alignment for coding regions than could be achieved using alignments tools that are not codon aware.

   Our experience has shown that it is important to adjust the penalty parameters of MACSE depending on the type of data you are using. Transcriptome data (such as assembled RNASeq data) can be aligned with the default parameters since the occurrence of the stop codon in the middle of the gene is less likely than the sequencing error. While exon sequences extracted from a genome may contain single base “overhangs,” so should be given a higher penalty for frameshifts. We advise caution, since inappropriately tuned parameters may result in alignment errors that will affect downstream results.

   The MACSE java application is invoked like so:

   ```
   java -jar macse_v0.8b2.jar -i your_orthologs_file.fa -o your_output_prefix
   ```

4. A full discussion of maximum likelihood, likelihood values, and likelihood ratio tests in molecular evolution is beyond the scope of this chapter. For the purposes of using PAML, the important principles are that *CODEML* and other programs in the PAML package use *maximum likelihood* to try to find parameters that best fit the observed data to the model you have specified (e.g.: model M0). The *likelihood function* is a function of the parameters of the model: the *likelihood* of a set of parameter values given the observed data is the probability of observing the data given the parameter values. It is not necessary to fully understand the theory of maximum likelihood to use PAML effectively. The main point to appreciate is that the log likelihood (lnL) is a probability of data fitting the model. The aim with testing various models is to find a model that better fits the data. The *likelihood ratio test* is used to determine if one data-model fit is significantly better than another. We recommend excellent Wikipedia articles as a short primer on these topics and [42, 61] for further reading.

5. This can be achieved, for example, using the function pchisq in R. Only for the branch test the LRT follows the χ^2 (chi-square) distribution. For the site test or the branch site test the LRT does not follow a χ^2 distribution, but using this distribution in both cases will make your *p*-values conservative [32].
6. A tutorial about using CODEML to calculate ω for 3,261 orthologs from six Plasmodium species is given as supplementary file worked_example.pdf. The data used are as described in [57]. All the files required to follow through this example are provided in the supplementary file worked_example_files.zip. All supplementary data will be available on http://www.danieljeffares.com/data.

Acknowledgements

We thank Ziheng Yang, Caroline Biagosch, and Sanne Nygaard for comments on the manuscript.

References

1. Kosiol C, Vinar T, da Fonseca RR, Hubisz MJ, Bustamante CD, Nielsen R, Siepel A (2008) Patterns of positive selection in six Mammalian genomes. PLoS Genet 4:e1000144
2. Yang W, Bielawski JP, Yang Z (2003) Widespread adaptive evolution in the human immunodeficiency virus type 1 genome. J Mol Evol 57:212–221
3. Lefébure T, Stanhope MJ (2009) Pervasive, genome-wide positive selection leading to functional divergence in the bacterial genus Campylobacter. Genome Res 19:1224–1232
4. Yang Z (1998) Likelihood ratio tests for detecting positive selection and application to primate lysozyme evolution. Mol Biol Evol 15: 568–573
5. Clark AG, Eisen MB, Smith DR, Bergman CM, Oliver B, Markow TA, Kaufman TC, Kellis M, Gelbart W, Iyer VN, Pollard DA, Sackton TB, Larracuente AM, Singh ND, Abad JP, Abt DN, Adryan B, Aguade M, Akashi H, Anderson WW, Aquadro CF, Ardell DH, Arguello R, Artieri CG, Barbash DA, Barker D, Barsanti P, Batterham P, Batzoglou S, Begun D, Bhutkar A, Blanco E, Bosak SA, Bradley RK, Brand AD, Brent MR, Brooks AN, Brown RH, Butlin RK, Caggese C, Calvi BR, de Carvalho AB, Caspi A, Castrezana S, Celniker SE, Chang JL, Chapple C, Chatterji S, Chinwalla A, Civetta A, Clifton SW, Comeron JM, Costello JC, Coyne JA, Daub J, David RG, Delcher AL, Delehaunty K, Do CB, Ebling H, Edwards K, Eickbush T, Evans JD, Filipski A, Szlig SF, Freyhult E, Fulton L, Fulton R, Garcia ACL, Gardiner A, Garfield DA, Garvin BE, Gibson G, Gilbert D, Gnerre S, Godfrey J, Good R, Gotea V, Gravely B, Greenberg AJ, Griffiths-Jones S, Gross S, Guigó R, Gustafson EA, Haerty W, Hahn MW, Halligan DL, Halpern AL, Halter GM, Han MV, Heger A, Hillier L, Hinrichs AS, Holmes I, Hoskins RA, Hubisz MJ, Hultmark D, Huntley MA, Jaffe DB, Jagadeeshan S, Jeck WR, Johnson J, Jones CD, Jordan WC, Karpen GH, Kataoka E, Keightley PD, Kheradpour P, Kirkness EF, Koerich LB, Kristiansen K, Kudrna D, Kulathinal RJ, Kumar S, Kwok R, Lander E, Langley CH, Lapoint R, Lazzaro BP, Lee S-J, Levesque L, Li R, Lin C-F, Lin MF, Lindblad-Toh K, Llopart A, Long M, Low L, Lozovsky E, Lu J, Luo M, Machado CA, Makalowski W, Marzo M, Matsuda M, Matzkin L, McAllister B, McBride CS, McKernan B, McKernan K, Mendez-Lago M, Minx P, Mollenhauer MU, Montooth K, Mount SM, Mu X, Myers E, Negre B, Newfeld S, Nielsen R, Noor MAF, O'Grady P, Pachter L, Papaceit M, Parisi MJ, Parisi M, Parts L, Pedersen JS, Pesole G, Phillippy AM, Ponting CP, Pop M, Porcelli D, Powell JR, Prohaska S, Pruitt K, Puig M, Quesneville H, Ram KR, Rand D, Rasmussen MD, Reed LK, Reenan R, Reily A, Remington KA, Rieger TT, Ritchie MG, Robin C, Rogers Y-H, Rohde C, Rozas J, Rubenfield MJ, Ruiz A, Russo S, Salzberg SL, Sanchez-Gracia A,

Saranga DJ, Sato H, Schaeffer SW, Schatz MC, Schlenke T, Schwartz R, Segarra C, Singh RS, Sirot L, Sirota M, Sisneros NB, Smith CD, Smith TF, Spieth J, Stage DE, Stark A, Stephan W, Strausberg RL, Strempel S, Sturgill D, Sutton G, Sutton GG, Tao W, Teichmann S, Tobari YN, Tomimura Y, Tsolas JM, Valente VLS, Venter E, Venter JC, Vicario S, Vieira FG, Vilella AJ, Villasante A, Walenz B, Wang J, Wasserman M, Watts T, Wilson D, Wilson RK, Wing RA, Wolfner MF, Wong A, Wong GK-S, Wu C-I, Wu G, Yamamoto D, Yang H-P, Yang S-P, Yorke JA, Yoshida K, Zdobnov E, Zhang P, Zhang Y, Zimin AV, Baldwin J, Abdouelleil A, Abdulkadir J, Abebe A, Abera B, Abreu J, Acer SC, Aftuck L, Alexander A, An P, Anderson E, Anderson S, Arachi H, Azer M, Bachantsang P, Barry A, Bayul T, Berlin A, Bessette D, Bloom T, Blye J, Boguslavskiy L, Bonnet C, Boukhgalter B, Bourzgui I, Brown A, Cahill P, Channer S, Cheshatsang Y, Chuda L, Citroen M, Collymore A, Cooke P, Costello M, D'Aco K, Daza R, De Haan G, DeGray S, DeMaso C, Dhargay N, Dooley K, Dooley E, Doricent M, Dorje P, Dorjee K, Dupes A, Elong R, Falk J, Farina A, Faro S, Ferguson D, Fisher S, Foley CD, Franke A, Friedrich D, Gadbois L, Gearin G, Gearin CR, Giannoukos G, Goode T, Graham J, Grandbois E, Grewal S, Gyaltsen K, Hafez N, Hagos B, Hall J, Henson C, Hollinger A, Honan T, Huard MD, Hughes L, Hurhula B, Husby ME, Kamat A, Kanga B, Kashin S, Khazanovich D, Kisner P, Lance K, Lara M, Lee W, Lennon N, Letendre F, Levine R, Lipovsky A, Liu X, Liu J, Liu S, Lokyitsang T, Lokyitsang Y, Lubonja R, Lui A, MacDonald P, Magnisalis V, Maru K, Matthews C, McCusker W, McDonough S, Mehta T, Meldrim J, Meneus L, Mihai O, Mihalev A, Mihova T, Mittelman R, Mlenga V, Montmayeur A, Mulrain L, Navidi A, Naylor J, Negash T, Nguyen T, Nguyen N, Nicol R, Norbu C, Norbu N, Novod N, O'Neill B, Osman S, Markiewicz E, Oyono OL, Patti C, Phunkhang P, Pierre F, Priest M, Raghuraman S, Rege F, Reyes R, Rise C, Rogov P, Ross K, Ryan E, Settipalli S, Shea T, Sherpa N, Shi L, Shih D, Sparrow T, Spaulding J, Stalker J, Stange-Thomann N, Stavropoulos S, Stone C, Strader C, Tesfaye S, Thomson T, Thoulutsang Y, Thoulutsang D, Topham K, Topping I, Tsamla T, Vassiliev H, Vo A, Wangchuk T, Wangdi T, Weiand M, Wilkinson J, Wilson A, Yadav S, Young G, Yu Q, Zembek L, Zhong D, Zimmer A, Zwirko Z, Alvarez P, Brockman W, Butler J, Chin C, Grabherr M, Kleber M, Mauceli E, MacCallum I (2007) Evolution of genes and genomes on the Drosophila phylogeny. Nature 450:203–218

6. Kimura M (1983) The neutral theory of molecular evolution. Cambridge University Press, Cambridge, 1968. ISBN 0-521-23109-4
7. Bustamante CD (2005) Population genetics of molecular evolution. In: Nielsen R (ed) Statistical methods in molecular evolution. Springer, New York
8. Sharp PM, Bailes E, Grocock RJ, Peden JF, Sockett RE (2005) Variation in the strength of selected codon usage bias among bacteria. Nucleic Acids Res 33:1141–1153
9. dos Reis M, Wernisch L (2009) Estimating translational selection in eukaryotic genomes. Mol Biol Evol 26:451–461
10. Yang Z, Nielsen R (2008) Mutation-selection models of codon substitution and their use to estimate selective strengths on codon usage. Mol Biol Evol 25:568–579
11. Jeffares DC, Pain A, Berry A, Cox AV, Stalker J, Ingle CE, Thomas A, Quail MA, Siebenthall K, Uhlemann A-C, Kyes S, Krishna S, Newbold C, Dermitzakis ET, Berriman M (2007) Genome variation and evolution of the malaria parasite Plasmodium falciparum. Nat Genet 39:120–125
12. Ziheng Y, Bielawski JP (2000) Statistical methods for detecting molecular adaptation. Trends Ecol Evol 15:496–503
13. Swanson, W. J., Z. Yang, M. F. Wolfner, and C. F. Aquadro. 2001. Positive Darwinian selection drives the evolution of several female reproductive proteins in mammals. Proc. Natl. Acad. Sci. USA 98:2509-2514
14. Anisimova M, Bielawski JP, Yang Z (2001) Accuracy and power of the likelihood ratio test in detecting adaptive molecular evolution. Mol Biol Evol 18:1585–1592
15. Anisimova M, Bielawski JP, Yang Z (2002) Accuracy and power of bayes prediction of amino acid sites under positive selection. Mol Biol Evol 19:950–958
16. Yang Z, Nielsen R (2002) Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. Mol Biol Evol 19:908–917
17. Yang Z (2007) PAML 4: phylogenetic analysis by maximum likelihood. Mol Biol Evol 24:1586–1591
18. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539
19. Löytynoja A, Goldman N (2008) Phylogeny-aware gap placement prevents errors in

sequence alignment and evolutionary analysis. Science 320:1632–1635

20. Stamatakis A, Ludwig T, Meier H (2005) RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics 21:456–463
21. Fitch WM (1970) Distinguishing homologous from analogous proteins. Syst Zool 19:99
22. Koonin EV (2005) Orthologs, paralogs, and evolutionary genomics. Annu Rev Genet 39: 309–338
23. Kuzniar A, van Ham RCHJ, Pongor S, Leunissen JAM (2008) The quest for orthologs: finding the corresponding gene across genomes. Trends Genet 24:539–551
24. Altenhoff AM, Dessimoz C (2012) Inferring orthology and paralogy. Methods Mol Biol 855:259–279
25. Trachana K, Larsson TA, Powell S, Chen W-H, Doerks T, Muller J, Bork P (2011) Orthology prediction methods: a quality assessment using curated protein families. Bioessays 33: 769–780
26. Kristensen DM, Wolf YI, Mushegian AR, Koonin EV (2011) Computational methods for gene orthology inference. Brief Bioinform 12:379–391
27. Salichos L, Rokas A (2011) Evaluating ortholog prediction algorithms in a yeast model clade. PLoS One 6:e18755
28. Moreno-Hagelsieb G, Latimer K (2008) Choosing BLAST options for better detection of orthologs as reciprocal best hits. Bioinformatics 24:319–324
29. Bork P, Dandekar T, Diaz-Lazcoz Y, Eisenhaber F, Huynen M, Yuan Y (1998) Predicting function: from genes to genomes and back. J Mol Biol 283:707–725
30. Tatusov RL, Koonin EV, Lipman DJ (1997) A genomic perspective on protein families. Science 278:631–637
31. Camacho C, Coulouris G, Avagyan V (2009) BLAST+: architecture and applications. BMC Bioinformatics 10:421
32. Yang Z, dos Reis M (2011) Statistical properties of the branch-site test of positive selection. Mol Biol Evol 28:1217–1228
33. Jordan G, Goldman N (2012) The effects of alignment error and alignment filtering on the sitewise detection of positive selection. Mol Biol Evol 29:1125–1139
34. Markova-Raina P, Petrov D (2011) High sensitivity to aligner and high rate of false positives in the estimates of positive selection in the 12 Drosophila genomes. Genome Res 21: 863–874
35. Fletcher W, 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
36. Castresana J (2000) Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. Mol Biol Evol 17:540–552
37. Notredame C, and Abergel C (2003) Using Multiple Alignment Methods to Assess the Quality of Genomic Data Analysis, in Bioinformatics and Genomes: Current Perspectives, M. Andrade, Editor. 2003, Horizon Scientific Press. p. 30–50
38. Penn O, Privman E, Landan G, Graur D, Pupko T (2010) An alignment confidence score capturing robustness to guide tree uncertainty. Mol Biol Evol 27:1759–1767
39. Privman E, Penn O, Pupko T (2012) Improving the performance of positive selection inference by filtering unreliable alignment regions. Mol Biol Evol 29:1–5
40. Suyama M, Torrents D, Bork P (2006) PAL2NAL: robust conversion of protein sequence alignments into the corresponding codon alignments. Nucleic Acids Res 34: W609–W612
41. Ranwez V, Harispe S, Delsuc F, Douzery EJP (2011) MACSE: multiple alignment of coding SEquences accounting for frameshifts and stop codons. PLoS One 6:e22594
42. Yang Z (2006) Computational molecular evolution. Oxford University Press, UK
43. Yang Z, Nielsen R, Goldman N (2009) In defense of statistical methods for detecting positive selection. Proc Natl Acad Sci U S A 106:E95–E95
44. Sergei L, Pond S, Frost S (2005) HyPhy: hypothesis testing using phylogenies. Bioinformatics 21:676–679, Advance Access published on March 1, 2005
45. Massingham T, Goldman N (2005) Detecting amino acid sites under positive selection and purifying selection. Genetics 169:1753–1762
46. Messier W, Stewart CB (1997) Episodic adaptive evolution of primate lysozymes. Nature 385:151–154
47. Swanson WJ, Nielsen R, Yang Q (2003) Pervasive adaptive evolution in mammalian fertilization proteins. Mol Biol Evol 20:18–20
48. Wong WSW, Yang Z, Goldman N, Nielsen R (2004) Accuracy and power of statistical methods for detecting adaptive evolution in protein coding sequences and for identifying positively selected sites. Genetics 168:1041–1051
49. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate—a practical and power-