Population Genetics Inference in R

Erik S. Wright

April 28, 2026

Contents

1 Introduction

Population genetics provides the theoretical foundation for the study of evolutionary processes. At the core of modern population genetics is the neutral mutation hypothesis, which challenged prior notions that most molecular variation was the result of classical Darwinian selection [1]. Neutral theory now provides a null model for many evolutionary investigations. By the late 20th century, there was sufficient data to make it clear that evolution was a balance between the processes of drift and selection that depended on the effective population size. This balance manifests in stark patterns of rate variation across different subsets of nucleotide sites in the genome (Fig. 1).

There are many different population genetics methods available, and it is often unclear which to use for a given purpose or dataset. This vignette showcases a select subset of methods that extract particularly useful information from sequence alignments. The purpose of this vignette is to illustrate how to apply and interpret these powerful population genetics functions. All of the DECIPHER functions for population genetics are named starting with "*Infer*" followed by the objective of their inference, because each function uses the first principles of population genetics to infer estimates of fundamental variables about a population.

Many assumptions underly population genetics functions, and those assumptions are thoroughly advertised on the help page for each function used here. An assumption made by all methods is that some sites evolve neutrally, including the third position of codons.

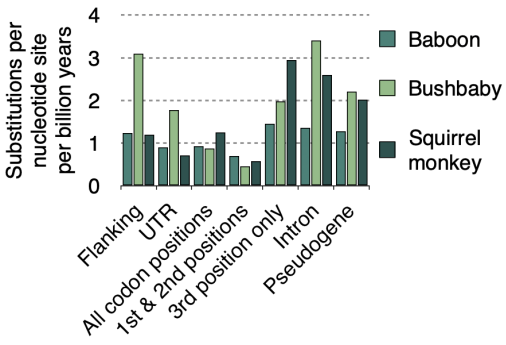

Figure 1: A comparison of estimated evolutionary rates for different classes of nucleotide sites based on the divergence of β-globin gene (HBB) sequences between humans and other primates. The β-globin pseudogene (HBBP1) is included as a reference of neutral evolution. Although there is considerable variation across organisms on this example from a single gene, third codon positions tend to evolve at high rates comparable to the neutral reference and much faster than the other codon positions.

This assumption is almost certainly violated in some cases [2]. However, using pseudogenes as a likely-neutral reference [3], it is apparent the third position in codons is generally under far less selection pressure than the other two codon positions (Fig. 1). Therefore, we will treat the third codon position as neutral throughout the examples below. This approximation has been exploited by population genetics approaches for many decades.

As the name implies, population genetics is applied to samples from populations. It is crucial to apply the functions in their applicable regime (Fig. 2). All of DECIPHER's population genetics functions are designed to be used on samples of sequences taken from within a species, not between different species. The `InferDemography` function infers population size changes from a large amount of sequence data with relatively few polymorphisms. The `InferRecombination` function needs slightly more polymorphisms, such that some are located nearby each other on the sequence. In contrast, the `InferSelection` function can handle amino acid changes, which requires an even greater degree of polymorphism.

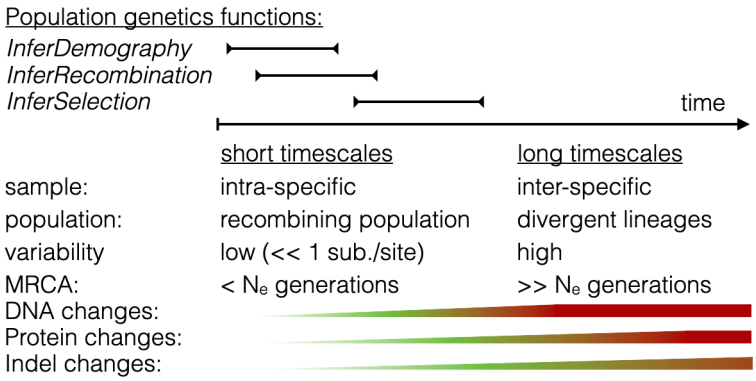

Figure 2: Population genetics functions are intended to be applied to sequences sampled from a population. In this regime, nucleotide differences have accumulated but the number of changes per site is far less than saturation (red). The `InferDemography` and `InferRecombination` functions require little nucleotide diversity within the population, but the `InferSelection` function requires sufficient variation for there to also possibly be changes to the corresponding protein sequence.

All three population genetics functions in DECIPHER take multiple sequence alignments as input and return a vector of numbers. They differ in the patterns of polymorphism used for inference. The intended use of each function is summarized in the table below. Although these functions are new implementations, please cite the relevant method's original publication when reporting results. A typical use case would be to run the function(s) on a set of different gene alignments and then aggregate or compare the results. Statistical significance for a single input can be obtained by bootstrapping the input sequences. For simplicity, this vignette provides an example analysis of a single alignment, which could be modified to loop through multiple different alignments.

Function	Input(s)	Output(s)	Reference
`InferDemography`	DNA alignment or site frequencies	Timeline of population size changes	[4]
`InferRecombination`	DNA alignments of 2+ sequences	Recombination parameters	[5]
`InferSelection`	DNA alignment of coding sequences	ω per codon or region	[6]

2 Getting Started

To get started we need to load the DECIPHER package, which automatically loads a few other required packages.

```
> library(DECIPHER)
```

Help for a function can be accessed through:

```
> ? InferDemography
```

Once DECIPHER is installed, the code in this tutorial can be obtained via:

```
> browseVignettes("DECIPHER")
```

For simplicity, all of the examples below will use the same sequences. Protein coding sequences are required for inferring selection, but demographic and recombination inference can use any aligned nucleotide sequences. In this vignette, we will load a set of 50S ribosomal protein L2 gene sequences.

```
> # specify the path to your file of sequences:
> fas1 <- "<<path to FASTA file>>"
> # OR use the example protein coding sequences:
> fas <- system.file("extdata",
          "50S_ribosomal_protein_L2.fas",
          package="DECIPHER")
> # read the sequences into memory
> dna <- readDNAStringSet(fas)
> dna
DNAStringSet object of length 317:
      width seq                                               names
  [1]   819 ATGGCTTTAAAAAATTTTAATC...ATTTATTGTAAAAAAAAGAAAA Rickettsia prowaz...
  [2]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [3]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [4]   822 ATGGGAATACGTAAACTCAAGC...CATCATTGAGAGAAGGAAAAAG Porphyromonas gin...
  [5]   819 ATGGCTATCGTTAAATGTAAGC...CATCGTACGTCGTCGTGGTAAA Pasteurella multo...
  ...   ... ...
[313]   819 ATGGCAATTGTTAAATGTAAAC...TATCGTACGTCGCCGTACTAAA Pectobacterium at...
[314]   822 ATGCCTATTCAAAAATGCAAAC...TATTCGCGATCGTCGCGTCAAG Acinetobacter sp....
[315]   864 ATGGGCATTCGCGTTTACCGAC...GGGTCGCGGTGGTCGTCAGTCT Thermosynechococc...
[316]   831 ATGGCACTGAAGACATTCAATC...AAGCCGCCACAAGCGGAAGAAG Bradyrhizobium ja...
[317]   840 ATGGGCATTCGCAAATATCGAC...CAAGACGGCTTCCGGGCGAGGT Gloeobacter viola...
```

It is important to carefully consider what population to analyze. All of the functions described here assume the input sequences were randomly sampled from an unstructured population of closely related individuals that may (or may not) be exchanging genetic material. It is crucial to randomly sample sequences from the same population, as bias in the input sequences will lead to bias in the results. For this vignette, we will limit the analysis to only sequences sampled from the species *Helicobacter pylori*. This can be accomplished by selecting a subset of sequences by their names.

```
> dna <- dna[startsWith(names(dna), "Helicobacter pylori")]
> dna
DNAStringSet object of length 75:
     width seq                                               names
 [1]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
 [2]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [3]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [4]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [5]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 ...   ... ...
[71]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[72]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[73]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[74]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[75]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
```

Next, it is necessary to align the input sequences. Since they are protein coding in this example, it is best to use the `AlignTranslation` function to preserve the reading frame by aligning the sequences via their amino acid translations. If the sequences were noncoding, `AlignSeqs` could be used instead.

```
> dna <- AlignTranslation(dna, verbose=FALSE)
> dna # all sequences have the same width
DNAStringSet object of length 75:
     width seq                                                names
 [1]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
 [2]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [3]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [4]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 [5]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
 ...   ... ...
[71]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[72]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[73]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[74]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACATAAA Helicobacter pylo...
[75]   828 ATGGCGATTAAAACTTATAAGCC...CATTTCCAGAAAGAAACACAAA Helicobacter pylo...
```

All three population genetics functions allow the specification of a *readingFrame* for protein coding inputs, which is denoted by the *first* position of the *first* codon in the alignment (i.e., `1`, `2`, or `3`). Analyzing individual codon positions requires that the reading frame be maintained throughout the alignment. Therefore, any frameshifts must be corrected beforehand by using `CorrectFrameshifts` on the (unaligned) coding sequences.

3 Inferring Demography

The `InferDemography` function fits a population genetics model [4] to the distribution of minor allele frequencies per alignment column, also known as the folded site frequency spectrum. This distribution will largely be affected by past changes in population size if we can assume the sites are evolving neutrally. Therefore, we will only use the third position of codons in our analysis by specifying that the *readingFrame* starts from the first position in the alignment (i.e., the "*A*" in the "*ATG*" start codon).

The function returns the estimated effective population sizes over different time intervals since the last common ancestor. This analysis requires the mutation rate and ploidy to calibrate the quantitative output, although these will have no effect on the qualitative picture of relative population size changes. The `InferDemography` function can also produce a plot showing the inferred step function of effective population sizes (Fig. 3).

The output shows the most likely step function only contained a single change in population size, resulting in two intervals. We see that the effective population size increased approximately ten million generations in the past, which would result in a higher abundance of singleton alleles than for a constant size population evolving under neutrality. The numeric output gives the three times (in units of generations) specifying the bounds of each of the two effective population size intervals. It then provides the observed and estimated folded site frequency spectra.

`InferDemography` would typically be used on an alignment with far more sites to increase accuracy. For this reason, it is possible to apply the function to many different gene alignments from the same set of organisms, then add the observed site frequencies together for use as input to the function. It is important to always confirm that the observed and spectra closely match, implying a satisfactory fit (as shown here).

```
> x <- InferDemography(dna, readingFrame=1, mu=1e-9, ploidy=1, show=TRUE)
Intervals = 1: LnL = -64.464
Intervals = 2: LnL = -57.908
Intervals = 3: LnL = -57.669

Time difference of 2.66 secs
> head(x, 20)
```

Intervals	LogLikelihood	Time 75	Time 13	Time 2
2.000000e+00	-5.790763e+01	2.425122e+04	9.421600e+06	5.384285e+07
Ne 75	Ne 13	Observed 0	Observed 1	Observed 2
6.729714e+07	2.422977e+07	1.620000e+02	3.400000e+01	1.500000e+01
Observed 3	Observed 4	Observed 5	Observed 6	Observed 7
9.000000e+00	5.000000e+00	6.000000e+00	3.000000e+00	2.000000e+00
Observed 8	Observed 9	Observed 10	Observed 11	Observed 12
2.000000e+00	3.000000e+00	2.000000e+00	4.000000e+00	1.000000e+00

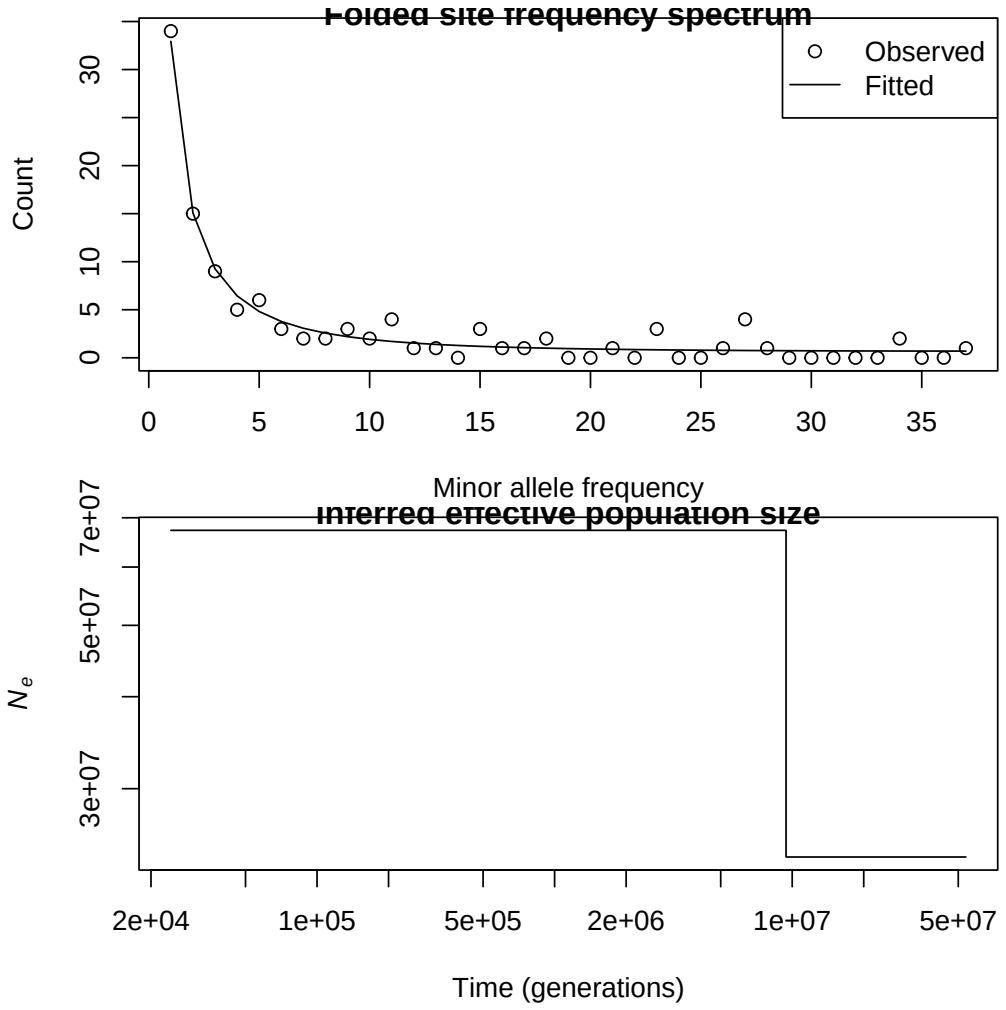

Figure 3: Fitted distribution of allele frequencies (top) and inferred past demographic changes (bottom).

4 Inferring Recombination

The `InferRecombination` fits a population genetics model to a "correlation profile" [5] derived from the decaying linkage between sites that are further apart. This method can be applied to pairs of aligned genomes (e.g., from `AlignSynteny`) or a multiple sequence alignment. As in the previous example above, we will assume neutrality by specifying *readingFrame* and only assessing the third position of codons.

The function fits a three parameter model and then uses these to infer multiple other derived parameters describing recombination. Since we specified a `readingFrame`, the analysis is applied to all three codon positions by default. The third codon position displays a curved ("L" shaped) correlation profile that is characteristic of recombination between the sampled genes and an external genetic pool (Fig. 4). The other two codon positions display much lower substitution rates and can be disregarded since we cannot assume neutrality.

The function's outputs are described on its help page. Possibly the most interesting parameter is the `ratio` of recombination to mutation, which was estimated here as approximately `2` using the third codon position. The fitted data shows a fair amount of noise, implying we could obtain better estimates by sampling more sequences from this population.

```
> y <- InferRecombination(dna, readingFrame=1, showPlot=TRUE)
================================================================================

Time difference of 2.21 secs
> head(y, 10)
                Position 1    Position 2   Position 3
fragment      6.324291e+04  1.000000e+03 2.621440e+05
theta_sample  2.031163e-23  1.000000e-05 1.825257e-04
phi_sample    1.522300e-05  5.000000e-05 2.636000e-04
theta_pool    1.154173e-02 -2.998800e-04 7.545867e-02
phi_pool      8.650204e+15 -1.499400e-03 1.089759e-01
ratio         7.494722e+17  5.000000e+00 1.444180e+00
coverage      3.909240e-01  3.225765e-02 9.787488e-01
d_pool        1.136680e-02 -3.000000e-04 6.856068e-02
d_clonal      2.031163e-23  9.999867e-06 1.824813e-04
d_sample      4.443557e-03  0.000000e+00 6.710756e-02
```

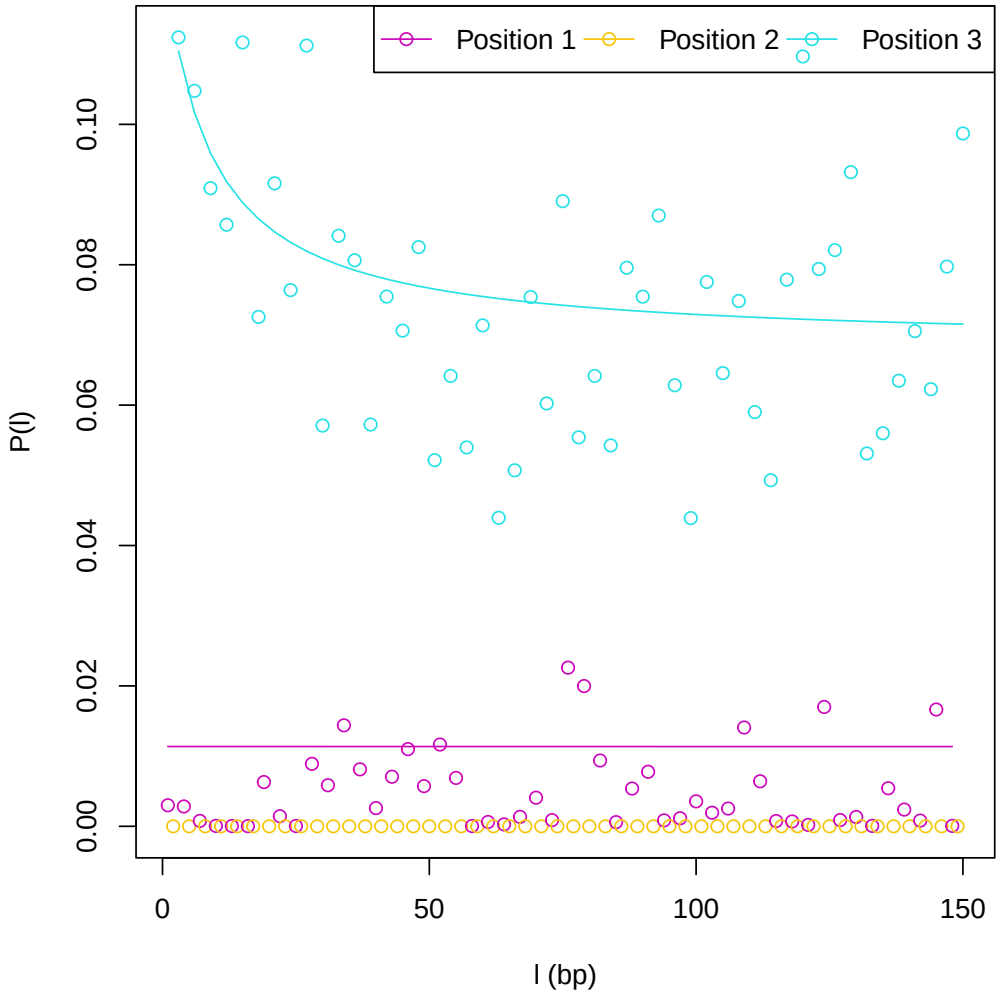

Figure 4: Fitted correlation profile at each codon position.

5 Inferring Selection

The `InferSelection` function estimates ω (also known as the K_a/K_s ratio or d_N/d_S ratio) by fitting a population genetics model [6]. Values of ω above `1` provide evidence of positive (Darwinian) selection, whereas ω values below `1` are indicative of negative (purifying) selection. This method can be applied to multiple sequence alignments of many sequences sampled from the same population and, unlike the other two methods, *requires* specification of a `readingFrame`.

The function provides maximum likelihood estimates of the transition/transversion ratio (κ), expected substitution rate (θ), and ω in non-overlapping regions across the alignment. The default behavior is to only calculate a single ω value for the whole alignment, but this will rarely be greater than `1` as negative selection generally overwhelms any signal of positive selection. With sufficient data, it is possible to specify a *windowSize* to estimate ω for adjacent groups of codons. The question is how much data is sufficient for inferring selection?

Simulations of codon evolution under the coalescent process with different values of ω provide a means of answering this question. Figure 5 shows the statistical power of detecting selection ($\omega \neq 1$) for increasingly larger numbers of codons per window at a significance level of `0.05`. It is apparent that detecting negative selection ($\omega < 1$) takes fewer codons than detecting positive selection ($\omega > 1$), and more extreme values of ω improve detection power. Detecting positive selection requires about `200` or more sampled codons per window. Therefore, for the `75` example sequences used here, it would make sense to specify a *windowSize* of at least `3` (i.e., $\lceil 200/75 \rceil$), as shown in the example below (Fig. 6).

Changing the `windowSize` to `1` would cause `InferSelection` to estimate ω for every codon site in the alignment. Statistically significant sites (or regions) with values of ω significantly above `1` are possibly of biological interest, as these represent where the protein is evolving under positive selection. It is particularly informative to apply this analysis to many different genes from the same population and rank the genes by the fraction (or number) of significantly positively selected codons.

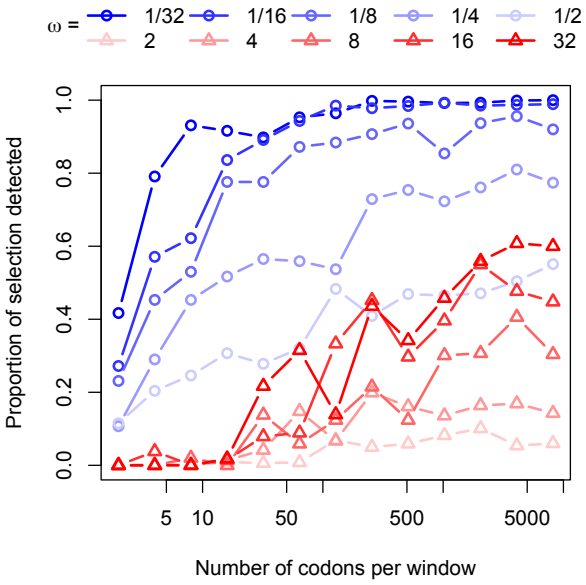

Figure 5: Statistical power of `InferSelection` for detecting different degrees of selection based on the total number of codons per window (i.e., number of codon sites times the number of sequences). Detecting positive selection (red) requires more codons than detecting negative selection (blue). Each point represents the average result of 1000 codon simulations under a coalescent process.

```
> z <- InferSelection(dna, windowSize=3, showPlot=TRUE)
LnL = -1734.064

Time difference of 70.03 secs
> head(z, 5)
LogLikelihood         theta         kappa     omega 1-3     omega 4-6
-1.734064e+03  1.265949e-01  1.041657e+01  2.478752e-03  6.737947e-03
> # fraction of windows under significant positive selection (> 2)
> mean(z[startsWith(names(z), "omega")] > 2 &
         z[startsWith(names(z), "pvalue")] < 0.05)
[1] 0
> # fraction of windows under significant negative selection (< 1/2)
> mean(z[startsWith(names(z), "omega")] < 1/2 &
         z[startsWith(names(z), "pvalue")] < 0.05)
[1] 0.7282609
```

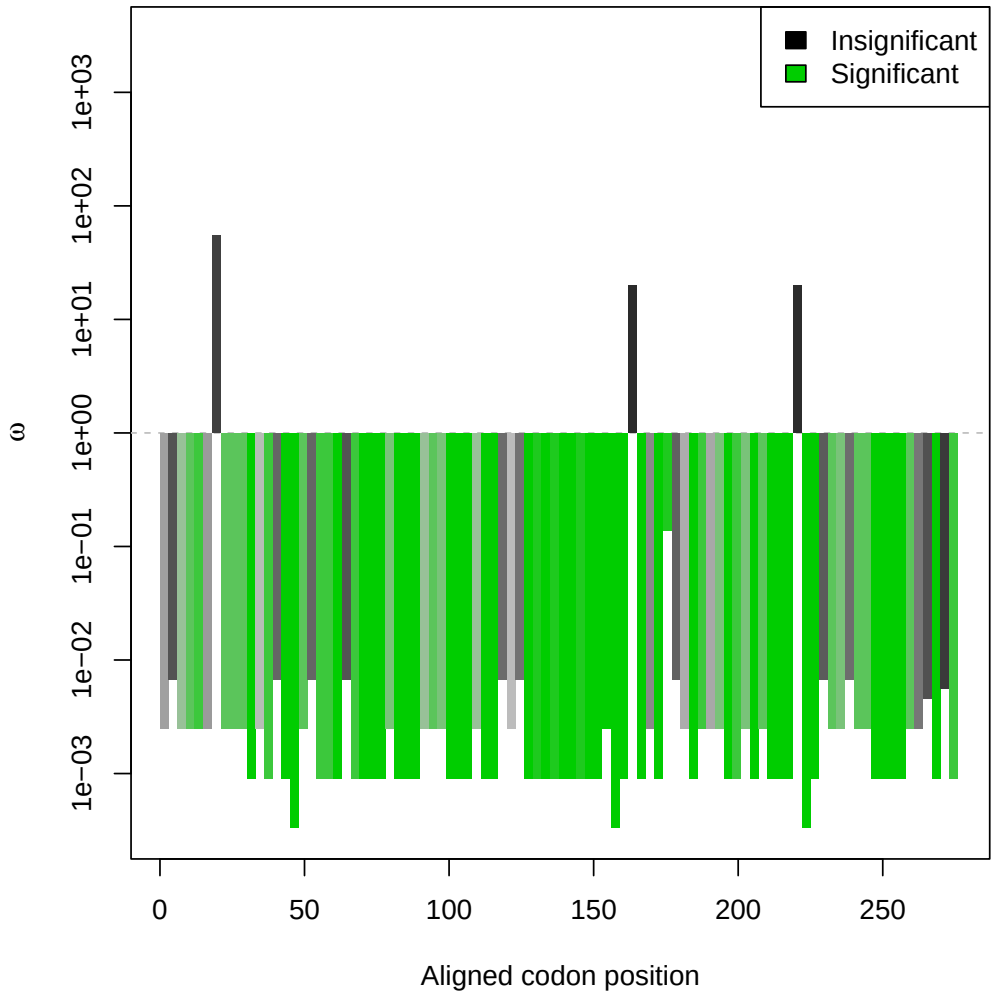

Figure 6: Estimates of ω across the alignment.

6 Session Information

All of the output in this vignette was produced under the following conditions:

- R version 4.6.0 RC (2026-04-17 r89917), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 24.04.4 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.24-bioc/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: BiocGenerics 0.59.0, Biostrings 2.81.0, DECIPHER 3.9.0, IRanges 2.47.0, S4Vectors 0.51.0, Seqinfo 1.3.0, XVector 0.53.0, generics 0.1.4
- Loaded via a namespace (and not attached): DBI 1.3.0, KernSmooth 2.23-26, compiler 4.6.0, crayon 1.5.3, otel 0.2.0, tools 4.6.0

References

[1] Kimura, M. Evolutionary rate at the molecular level. *Nature*, 217(5129), 624-626. doi:10.1038/217624a0, 1968.

[2] Shen, X., Song, S., Li, C., & Zhang, J. Further Evidence for Strong Nonneutrality of Yeast Synonymous Mutations. *Molecular Biology and Evolution*, 41(11), msae224. doi:10.1093/molbev/msae224, 2024.

[3] Douglas, G. M. & Shapiro, B. J. Pseudogenes act as a neutral reference for detecting selection in prokaryotic pangenomes. *Nature Ecology and Evolution*, 8(2), 304-314. doi:10.1038/s41559-023-02268-6, 2024.

[4] Lynch, M., Haubold, B., Pfaffelhuber, P., & Maruki, T. Inference of Historical Population-Size Changes with Allele-Frequency Data. *G3*, 10(1), 211-223. doi:10.1534/g3.119.400854, 2020.

[5] Lin, M. & Kussell, E. Inferring bacterial recombination rates from large-scale sequencing datasets. *Nature Methods*, 16(2), 199-204. doi:10.1038/s41592-018-0293-7, 2019.

[6] Wilson, D. & CRyPTIC Consortium GenomegaMap: within-species genome-wide dN/dS estimation from over 10,000 genomes. *Molecular Biology and Evolution*, 37(8), 2450-2460. doi:10.1093/molbev/msaa069, 2020.