Guide for the use of the "CodonInfo" package

OCTAVIO MARTÍNEZ
COMPUTATIONAL BIOLOGY LABORATORY, CINVESTAV - IRAPUATO, MÉXICO.

ABSTRACT. The R package "**CodonInfo**" contains a sample of raw codon frequencies in 35 species as well as a set of functions to estimate in detail the properties of those data in the context of the genetic code. Even when the (nuclear) genetic code is *universal*, each taxon at species or higher taxonomic level presents particularities in the frequency of codon use (codon bias) or even in the use of different bases within codons. Using Shannon's entropy formula under different hypotheses it is possible to estimate and dissect the information present in different strata within each species. This guide presents the basic concepts needed to estimate and understand the different uses of the genetic code in each species, preparing the reader to make relevant comparisons between species. Throughout this document we exemplify the use of the functions with the data for our own specie.

Even when data for only 35 species are in the package, the reader can include further species of her/his interest.

Here we will not discuss the biological implications of the use of the functions; that will be done in a manuscript in preparation putatively entitled "*Sampling informational properties of codon use through the tree of life*"; however, it appears that interesting and relevant biological knowledge could be obtained with this package.

Content

1. Introduction 2
2. Preliminary Concepts 3
2.1. Entropy (H) 3
2.2. Information (I) 5
3. Data and Functions 7
4. Running each one of the functions 11
4.1. `est.H()`: Estimates informational properties from a vector of relative frequencies 11
4.2. `codons2bases()`: Frequencies of bases in a species 12
4.3. `prop.bases()`: Relative frequencies and informational properties per bases 14
4.4. `prop.codons()`: Estimates informational properties at codon level 16
4.5. `codon.bias()`: Informational properties for amino acids coded by more than one codon 22
5. Further data and analyses 27
References 29

Date: July 8, 2024.

1. Introduction

The aim of this document is to explain the data and functions available in the **CodonInfo** R package. It is assumed that the reader is familiar with the R environment for statistical computing (R Core Team, 2013); if that is not your case it will be difficult to follow and understand the material.

Previously you must have installed the **CodonInfo** R package from the file "`CodonInfo_1.0.tar.gz`".

It is also assumed that you have at hand the manual of the package (file "`CodonInfo-manual.pdf`"), which complements the on-line help for the package.

It is recommended that the user will input into the R window the commands that will be presented in text boxes. The first of those boxes is

```
------------------------------------------------------------------------------------------
# Box 1.
# Initiate a session with the CodonInfo package
# (you must have installed the package previously)
> library(CodonInfo) # Load the package
# In all boxes in is assumed that you have done this step

# Explore the main help page of the package
> ? CodonInfo

> ? gen.code # See the help for the gen.code data.frame

> head(gen.code) # See the first rows of that data.frame
  codon  aa fb sb tb
1   GCT Ala  G  C  T
2   GCC Ala  G  C  C
3   GCA Ala  G  C  A
4   GCG Ala  G  C  G
5   CGT Arg  C  G  T
6   CGC Arg  C  G  C
------------------------------------------------------------------------------------------
```

First, the comment sign, "#", as well as text following that in the same row do not have effect in the commands, and the rows beginning with comments are included only as a guide of what is going on. Second, note that the R prompt, ">", is included at the beginning of some of the rows where a command is going to be given to the program. You could copy the text following that prompt and past it in the R command window. In some cases, as when giving the command "`head(gen.code)`", the answer given by R is presented in the `Box`.

It is assumed that you are going to follow sequentially the calculations in this document. For example, always that you initiate o continue a session where the package will be used you must input the following lines:

```
------------------------------------------------------------------------------------------
# Compulsory command BEFORE beginning a section:
library(CodonInfo) # Load the package
------------------------------------------------------------------------------------------
```

If you have queries that could not be solved with the help included in the manual or in this document, please send me an e-mail to "octavio.martinez@cinvestav.mx" with subject "**CodonInfo query**". I will also be grateful to receive criticisms or suggestions for further versions of the package. Some external web links are also shown in blue in this document.

Unless you are familiar with the use of information theory (see for example the book "*The Evolution of Biological Information: How Evolution Creates Complexity, from Viruses to Brains*" Adami (2024)), it is highly recommended that you read carefully the next section of this guide.

2. Preliminary Concepts

2.1. **Entropy (H).** In 1948 Claude Shannon set the bases of communication theory (Shannon, 1948). Central to his approach was the concept of *entropy*, defined for discrete channels as

$$H = -\sum_{i=1}^{i=s} p_i \log_2(p_i) \tag{1}$$

where H is entropy, s is the number of symbols that exist in the alphabet of interest; $s \in 2, 3, \cdots$, each p_i is the probability that the i-th symbol appears in a message ($\sum_i p_i = 1$) and we use the function $log_2()$, to denote the logarithm with base 2, i.e., $log_2(1) = 0, log_2(2) = 1, log_2(4) = 2, \cdots$ We also define

$$\lim_{p \to 0} (p \log_2(p)) = 0$$

What is H measuring? First, note that the formula in equation (1) includes a set of probabilities, p_i, thus we are implying the existence of a random variable, say X, which can take values $x_1, x_2, \cdots, x_s$ with probabilities $P[X = x_i] = p_i$, thus H is a statistical property of that variable, in the sense that it depends on the set of probabilities.

To begin to understand the nature of H, let's consider the simplest case, where we have only two symbols in the alphabet, i.e., $s = 2$ in equation (1). In that case we can also simplify our notation setting $p_1 = p$ and $p_2 = q = 1 - p$ thus

$$H = -(p \log_2(p) + q \log_2(q))$$

This case illustrate the transmission in a digital channel, where the symbols are for example 1 and 0 (binary digits) as used in computers and other digital devices. Figure 1 presents the plot of H as function of p in this particular case.

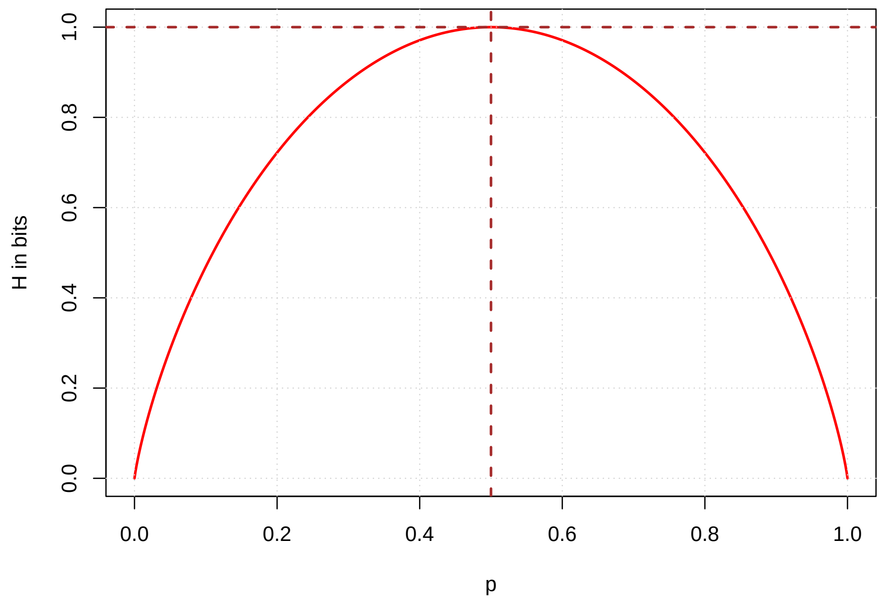

Figure 1. Plot of $H = -(p \log_2(p) + q \log_2(q))$; $q = 1 - p$. Presented as "*Fig. 7 – Entropy in the case of two possibilities with probabilities p and $1 - p$.*" in Shannon (1948).

In Figure 1 we can see that H varies between 0 and 1, and that the maximum value of H, say, $H^m = 1$, is reached in the point $p = 0.5$, where the two brown dashed lines intercept.

From this example we can see that entropy, H, is a measure of disorder, randomness, or uncertainty in the messages transmitted in a given channel. H gives values of 0 when the messages consist of a single symbol, corresponding to the values of $p = 0$ and $p = 1$, because in those cases there is no uncertainty. For example, assume that messages of ten symbols coded in the alphabet 0 and 1 are obtained, and consider the following examples:

$$(1) : \{0000000000\} \to \hat{p} = 0.0, \hat{H} = 0$$

$$(2) : \{1001100101\} \to \hat{p} = 0.5, \hat{H} = 1$$

$$(3) : \{1111111111\} \to \hat{p} = 1.0, \hat{H} = 0$$

If the message (1) is the only one that we have, it is clear that the better estimate of the probability to obtain 1 is $\hat{p} = 0.0$, and we can consider that there is no uncertainty, $\hat{H} = 0$. Also if the only message that we know is (3) we obtain $\hat{p} = 1.0$, and in that case there is again no uncertainty, thus $\hat{H} = 0$. Finally, if the message is as (2) we have maximum uncertainty, i.e., in that case $\hat{p} = 0.5$ and this is reflected in the maximum value of H, say, $\hat{H} = 1$

In the examples above we have put the "*hat*" above p and H to denote that those quantities are *estimated* from some data (messages). In fact, we rarely –if ever, know the true values of the probabilities p_i, and thus the better that we could do is to estimate those values from a set of data and thus obtain also an estimate of the entropy, $\hat{H}$.

In summary, H measures the *uncertainty* or also —in some sense— the *disorder* that exist in a given context.

Note that in Figure 1 we could plot the relation between the single parameter, p, and H, because in that case the frequency of the two symbols (say "1" and "0") could be collapsed in a single value, by considering that the probability of 0 was just $1 - p$. In general when the number of symbols in the alphabet being considered is larger than 2, i.e., when $s = 3, 4, \cdots$, it is imposible to plot the multivariate parameter space in a simple way. Nonetheless, in the cases for $s > 2$ we can still use equation (1) to estimate the entropy or uncertainty in a given situation.

In the case of an alphabet of two letters we have seen that the maximum uncertainty was of one bit, i.e., when $s = 2$, the maximum value of H, say $H^m = 1$ bit or $\log_2(2) = 1$. That fact can be generalized; when we have an alphabet of s symbols the maximum uncertainty will happens when all the s symbols have the same probability of occur, $p_i = 1/s$. Thus, in general, the maximum value of H^m will be

$$(2) \qquad H^m = -\sum_{i=1}^{i=s} \frac{1}{s} \log_2(\frac{1}{s}) = -s(\frac{1}{s}) \log_2(\frac{1}{s}) = -(\log_2(1) - \log_2(s)) = \log_2(s)$$

Briefly, we have a measure, H, which is a function of the vector of probabilities $\mathbf{p} = (p_1, p_2, \cdots, p_s)$ and which can take values in the interval $[0, \log_2(s)]$, being an average in bits of the uncertainties that happens in each one of the characters of a message.

It is also important to note that H is a measure per symbol read in a message, and its value is a weighted average of what happens when messages with an infinite number of characters are known.

We have set the unit of measure of H in bits, by taking in equation (1) logarithms of base 2. However this is arbitrary and, has mentioned in (Adami, 2004), if there are s symbols in the alphabet, we can take logarithms of base s, instead of $\log_2()$, and in that way the values of $H^* = -\sum_i p_i \log_s(p_i)$ will be always between 0 and 1. Adami (2004) use the term "*mers*" to denote the units of H^* when using $\log_s()$ in Shannon's formula. However here we will keep the measurement of H in bits, as presented in equation (1). Bits (see Bit in the Wikipedia) are the basic unit of information in computing and digital communication.

2.2. **Information (I).** *Information* is a complex concept (see for example Information in the Wikipedia), common definitions are, from the Oxford dictionary "*facts or details about somebody/something*" or, from the Merriam-Webster dictionary, "*knowledge obtained from investigation, study, or instruction*", etc.

In general we can think that we are *informed* about something when we pass from an state of absolute uncertainty to other one when we have less uncertainty, or, in more general terms, when the degree of uncertainty about a phenomenon changes.

Given that we have a way to measure uncertainties via entropy, H, it appears reasonable to measure information as differences of H under different circumstances or hypotheses. In general we will measure information as a change of entropies, say as

$$I_{a \to b} = H_a - H_b$$

where the term "$I_{a \to b}$" could be read as "*information from a to b*", and H_a and H_b are the entropies (uncertainties) calculated under those conditions, respectively.

In particular, if we are in the framework of an alphabet with s symbols (letters or characters), we have seen that the maximum value of H, H^m, is $H^m = \log_2(s)$ and it occurs when each one of the probabilities, p_i, is equal to $p_i = 1/s$, that is, it is *assumed* that all the s symbols will occur at the same frequency. The next stage of knowledge occurs if we know —or at least have very good estimates— of the values of the p_i's. In that case we have that the information gained by the knowledge of the value of the probabilities is given by

$$\hat{I} = H^m - \hat{H} = \log_2(s) - \hat{H} \tag{3}$$

where the estimated entropy, $\hat{H}$, is calculated from the estimated probabilities,

$$\hat{H} = -\sum_{i=1}^{i=s} \hat{p}_i \log_2(\hat{p}_i) \tag{4}$$

Note that $\hat{I}$ in equation (3) will be always positive.

For discussion and better understanding let's put a numerical example in the framework of genes for 4 species. Figure 2 and Table 1 present such example.

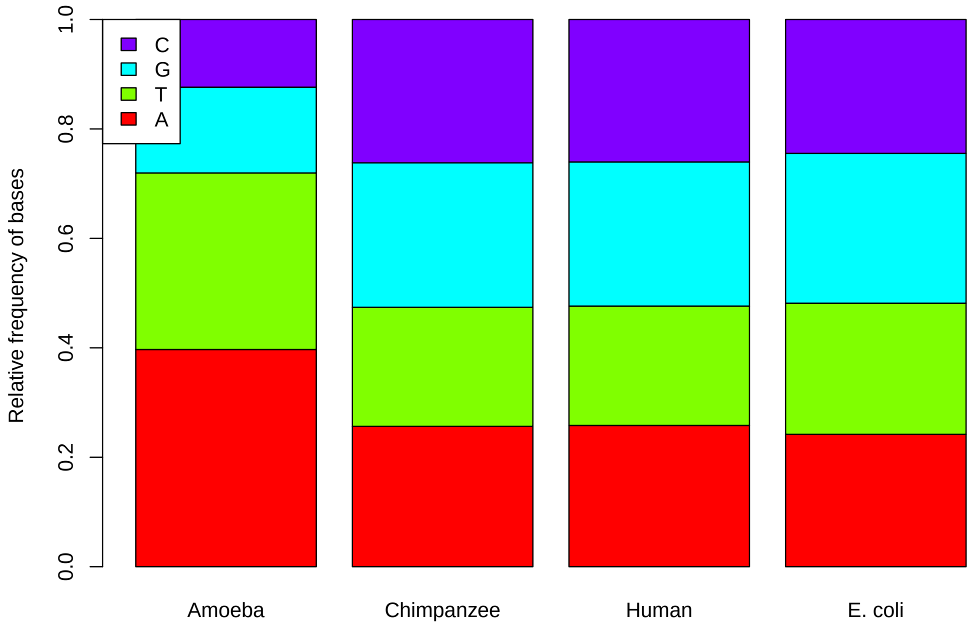

FIGURE 2. Relative frequencies for the four DNA bases, $\hat{p}_i$, in four species (see also Table 1).

Table 1. Relative frequencies for the four DNA bases, $\hat{p}_i$, and estimates of H and I for four species.

Species	Estimated frequencies ($\hat{p}_i$)				$\hat{H}$	$\hat{I}$
	A	T	G	C		
Entamoeba histolytica (Amoeba)	0.3969	0.3226	0.1568	0.1237	1.8478	0.1522
Pan troglodytes (Chimpanzee)	0.2564	0.2176	0.2640	0.2620	1.9957	0.0043
Homo sapiens (Human)	0.2581	0.2181	0.2634	0.2604	1.9959	0.0041
E. coli (bacteria)	0.2419	0.2396	0.2738	0.2447	1.9978	0.0022

Figure 2 presents the stacked relative frequencies of the four DNA bases in genes of four species, while Table 1 presents the relative frequencies of those bases, $\hat{p}_i$, and estimates of H and I in the same four species. The criteria to include the species in the figure and table were first, that the species presented an extreme value of $\hat{H}$, thus the first row in the table present the case with the minimum and the fourth row the case with the maximum value of that parameter among the 35 species studied. Second, the cases of the chimpanzee and human (rows 2 and 3 in the table) are presented for their particular interest.

In Table 1 we can notice first that the value of $\hat{H}$ efficiently summarizes the average differences between the estimated frequencies of the four bases, $\hat{p}_i$, and the expected value under the assumption of identical frequency of the four bases, $p_i = 0.25$; in fact, there is an inverse linear relation between $\hat{H}$ and $\sum_i(\hat{p}_i - p_i)^2$. For the cases shown in Table 1 Pearson's correlation coefficient between those quantities is estimated as $\hat{r} \approx -1$, i.e., large differences between $\hat{p}_i$ and $p_i = 0.25$ imply a smaller value of $\hat{H}$ and vice versa.

This in turn implies that the estimated information for each specie, $\hat{I} = \log_2(4) - \hat{H} = 2 - \hat{H}$, is larger when the estimated frequencies are far away from the value $p_i = 0.25$ –which implies the maximum entropy, $H^m = 2$ bits.

In the first row of Table 1 we see that $\hat{I} = 0.1522$ bits for the amoeba, which present a very high frequency of A and T, compared with G and C. This means that the knowledge of the relative frequencies of the bases in the amoeba genes substantially alters the probabilities of finding specific sequences of bases within the genes of this species. For example, assume that we are interested in estimating the probability of finding the sequence "ATGC" within the genes of that species. Without the knowledge of the relative frequencies the estimate of that probability will be $P[\text{ATGC}] = 0.25 \times 0.25 \times 0.25 \times 0.25 = 0.00390625$, which is the same estimated probability for any sequence of 4 bases in the genes of amoeba. In contrast, by the knowledge of the relative frequencies of the bases in that specie, the probability of finding the sequence "ATGC" can be calculated with much higher precision as $P[\text{ATGC}] = 0.3969 \times 0.3226 \times 0.1568 \times 0.1237 = 0.002483421$ Note that the ratio of those quantities, $0.00390625/0.002483421 \approx 1.57$ is large, meaning that our estimates substantially differ by the knowledge of the true frequencies of the bases[1].

To complete this example, let's consider the differences in expected frequencies in sequences of four bases that have the same base in the amoeba's genes. Without the knowledge of the true bases's frequencies, the probabilities of any sequence having the same base is the same, in particular

$P[\text{AAAA}] = P[\text{CCCC}] = 0.00390625$ and thus the ratio $P[\text{AAAA}]/P[\text{CCCC}] = 1$. However, by calculating those probabilities with the true frequencies we find

$P[\text{AAAA}] \approx (0.3969)^4 = 0.0248$ while $P[\text{CCCC}] \approx (0.1237)^4 = 0.0002$ and thus the ratio

$P[\text{AAAA}]/P[\text{CCCC}] \approx 106$, i.e., knowledge of the true frequencies of the bases allow us to determine that finding a sequence of four consecutive A's is more than 100 times more likely than finding a sequence of four consecutive C's in that species.

[1]These calculations assume fully independence between frequencies of subsequent bases, which is a good first approximation.

In other words, without the knowledge of the true frequencies of the bases any sequence of a given size, say of k bases, will have the same probability of being found, $(0.25)^k$, giving a fully flat probability distribution for the probabilities of sequences of any given size k. In contrast, the knowledge of the true frequencies of the bases in a specie, which is measured by the estimated information, $\hat{I}$, tell us how much the heterogeneity in the frequencies of the bases gives differences in the probabilities of finding specific sequences; larger values of $\hat{I}$ in one species compared with another means that we have larger differences in those probabilities.

Now, let's compare the values of $\hat{I}$ for the four species presented in Table 1; those values are 0.1522, 0.0043, 0.0041and 0.0022 for the amoeba, chimpanzee, human and bacteria, respectively, and the ratios of $\hat{I}$ comparing the largest value ($\hat{I} = 0.1522$ for the amoeba) with the ones for the other three species are approximately 36, 37 and 70 times larger in the amoeba, compared with the chimpanzee, human and bacteria, respectively.

3. Data and Functions

Table 2 presents the 9 objects that comprises the **CodonInfo** package with the class at which each one of the objects belongs and a brief description.

In Table 2 we can see that there are 4 objects that contain data and 5 functions to obtain estimates from the original data (`codon.freq`) which consist of the raw numbers of codons for 35 species.

Each one of the 35 columns of `codon.freq` was obtained by a single query to the web site

"http://codonstatsdb.unr.edu/", which is documented in (Subramanian et al., 2022).

The selection of the 35 species was somehow arbitrary; I tried to obtain a broad sample from the immense tree of life, but sampling only a small number of different species from diverse kingdoms and including a minimum of two species per kingdom. The aim of this sampling scheme was to demonstrate the usefulness of studying the informational properties of codon data.

Nevertheless, the number of species sampled can be increased in an unlimited way by downloading more files with raw data of codon frequencies from the web site above or other convenient source. To include more data into the package, please see the document "`IncludeFurtherDataInCodonInfo.pdf`" –which at this point is in preparation; please visit the same web address (URL) from which you downloaded the package **CodonInfo** for updates.

Table 3 contains a summary of the content of the object "`desc.codon.freq`"; that object is a `data.frame` which briefly describes the species for which the package has codon data. I acknowledge **Guillermo Martnez de la Vega** for help in the selection of species presented in Table 3 and corrections of some initial classification mistakes. However, any remaining errors or inaccuracies are of my exclusive responsability.

Table 3 includes 8 columns: "`nc`" –the number of column in the "`codon.freq`" object (that number serves as input parameter in various functions, allowing the selection of the data to be analyzed); "`Kingdom`" –the kingdom to which the species belongs; "`Species`" –the scientific name of the species; "`common.name`" –the common name of the organism; "`tax.id`" –the numeric identifier of the species, which can be used for example in the NCBI taxonomy web site; "`key`" –the key formed by the first two letters of the scientific name (`Species`) separated by a full stop; "Genes" –the total number of genes from which the data were obtained and, finally "Codons (K)" –the number codons in thousands (K) from which the data were obtained.

Rows in Table 3 are ordered by `Kingdom`, going from the more to the less numerous cases, thus we have 18 rows with "Animal", 6 rows with "Plant", 3 rows with "Protoctista" and two rows for the remaining 4 kingdoms, "Fungi", "Monera", "Archaea" and "Virus".

Table 2. Objects that exist in the **CodonInfo** R package.

Object	`class`	Description
`codon.freq`	`matrix`	A matrix of 64 rows × 35 columns which rows are codons and columns are libraries. This is the core source of data for the package. Many functions use the number of column (`nc`) as input. Library names (columns) are formed by the first letters of the species followed by the taxa.id. For example `nc=16` has name `H.s.9606`, etc.
`desc.codon.freq`	`data.frame`	A `data.frame` with 35 rows and 12 columns that describe the data in `codon.freq` Variables `Kingdom`, `Species` and `common.name` are self explanatory. `key` is formed by the first two letter of the species and other variables give general information.
`gen.code`	`data.frame`	A `data.frame` with description of the genetic code for the 64 codons. Variables `codon`, `aa` (amino acid), and the first, second and third base of the codon in variables `fb`, `sb` and `tb`, respectively.
`gen.code.n.cod`	`data.frame`	A `data.frame` with 21 rows, one for each amino acid (`aa`), and info about the number of codons and codons for each `aa`. `aa` - Three letters code names for each `aa`; `n.cod` - Number of codons coding for the `aa`; `codons` - The codons that result in each `aa`.
`est.H`	`function`	Estimates information properties from a vector of relative frequencies, `x`. Output: `n` - length of the vector `x`; `Hest` - Estimated value of the entropy, $\hat{H}$; `EN` - Effective Number of units, 2 to the $\hat{H}$ power; `Info` $= \log_2(n) - \hat{H} = H^m - \hat{H}$.
`prop.bases`	`function`	Parameters: `nc` - Number of Column; `print.data` (logic); `by.row` (logic) Output: If `by.row=F`, a `data.frame` with relative frequencies of each base in the first (`F`), second (`S`) and third (`T`) codon positions as well as in the `Total` of all three codon positions, also columns with information properties derived from that matrix. If `by.row=T` same information in a single row.
`codons2bases`	`function`	Parameters: `nc` - Number of Column; `print.data` (logic); `only.summary` (logic) Output: If `only.summary=T`, a matrix with a summary of frequencies; otherwise also a data.frame named `f.c.b` with details per codon.
`prop.codons`	`function`	Input: `nc` (Number of Column of `codon.freq`) Output: a list with two components, the first `H.est`, a 3 rows by 4 columns `matrix` with informational properties for three frequencies and the second, `frequencies`, a `data.frame` with 64 rows (one for each codon) with information for three frequencies.
`codon.bias`	`function`	Input: `nc` (Number of Column of `codon.freq`) Output: a `data.frame` with 19 rows (one for each amino acid coded by at least two codons) and 11 columns with information properties, including `CodBias.aa`

The other two objects that contain data in the package are "`gen.code`" and "`gen.code.n.cod`". Both objects give representations of the nuclear genetic code, the difference being that `gen.code` has 64 rows –one for each codon, with columns "`codon`" –the three bases that constitute the codon; "`aa`" –the three letter representation of each one of the 20 amino acids, plus the key word "`Stop`" for the codons that terminate the protein synthesis. The last three columns in "`gen.code`" are "`fb`" "`sb`" and "`tb`" and contain the first, second and third base of the codon, respectively.

On the other hand, object "`gen.code.n.cod`" gives the inverse nuclear genetic code. That object is presented in Table 4, and has columns: "`aa`" –the amino acid, "`n.cod`" –the number of codons coding for the corresponding `aa` and "`codons`" a character string giving the codons.

Table 3. Summary of the content of object "`desc.codon.freq`".

`nc`	`Kingdom`	`Species`	`common.name`	`tax.id`	`key`	Genes	Codons (K)
1	Animal	*Acropora digitifera*	coral	70779	A.d	26,060	11,650
4	Animal	*Canis lupus familiaris*	dog; carnivores	9615	C.l	21,148	11,784
5	Animal	*Capra hircus*	goat; ungulates	9925	C.h	20,659	11,568
7	Animal	*Ciona intestinalis*	tunicates	7719	C.i	13,633	7,571
8	Animal	*Danio rerio*	zebrafish; bony fishes	7955	D.r	26,239	15,023
9	Animal	*Drosophila melanogaster*	fruit fly; insecta	7227	D.m	13,930	7,472
13	Animal	*Felis catus*	domestic cat; carnivores	9685	F.c	19,802	11,463
15	Animal	*Gallus gallus*	chicken; birds	9031	G.g	17,833	10,412
16	Animal	*Homo sapiens*	human; primates	9606	H.s	19,850	11,577
17	Animal	*Monodelphis domestica*	opossum; marsupials	13616	M.d	21,045	11,846
18	Animal	*Mus musculus*	mouse; rodents	10090	M.m	22,515	12,248
19	Animal	*Octopus sinensis*	octopus; cephalopods	2607531	O.s	18,873	9,880
20	Animal	*Penaeus vannamei*	shrimp; crustaceans	6689	P.v	24,818	10,791
22	Animal	*Pomacea canaliculata*	gastropods	400727	P.c	21,126	11,578
25	Animal	*Stegodyphus dumicola*	spiders	202533	S.d	20,837	9,157
26	Animal	*Strongylocentrotus purpuratus*	sea urchin	7668	S.p	27,430	15,852
29	Animal	*Xenopus laevis*	toad	8355	X.l	34,660	19,703
34	Animal	*Pan troglodytes*	Chimpanzee; Homininae	9598	P.t	21,638	12,062
3	Plant	*Arabidopsis thaliana*	eudicots	3702	A.t	27,245	11,095
6	Plant	*Chlamydomonas reinhardtii*	green algae	3055	C.r	17,742	13,031
21	Plant	*Physcomitrium patens*	mosses	3218	P.p	20,325	10,309
23	Plant	*Quercus suber*	oak; eudicots	58331	Q.s	49,316	22,318
30	Plant	*Zea mays*	maize; monocots	4577	Z.m	34,063	14,093
35	Plant	*Triticum dicoccoides*	Wheat; Poaceae	85692	T.d	66,780	28,419
10	Protoctista	*Entamoeba histolytica*	Entamoeba	294381	E.h	8,163	3,427
14	Protoctista	*Galdieria sulphuraria*	red algae	130081	G.s	6,622	2,794
24	Protoctista	*Salpingoeca rosetta*	choanoflagellates	946362	S.r	11,618	7,737
2	Fungi	*Alternaria alternata*	ascomycete	5599	A.a	13,466	6,162
28	Fungi	*Ustilago maydis*	smut fungi; Basidiomycete	237631	U.m	6,764	3,998
12	Monera	*Escherichia coli*	enterobacteria	562	E.c	5,219	1,515
33	Monera	*Endomicrobium proavitum*	as-yet uncultured organism	1408281	E.p	1,333	483
27	Archaea	*Thermococcus barophilus*	euryarchaeotes	391623	T.b	2,208	630
32	Archaea	*Haladaptatus cibarius*	Methanobacteriota salt resistent	1455608	H.c	4,004	1,098
11	Virus	Enterovirus C	virus	138950	E.C	1	2
31	Virus	SARS coronavirus	COVID virus	227984	S.c	13	10

Table 4. Inverse nuclear DNA codon table (object “`gen.code.n.cod`”).

`aa` (amino acid)	`n.cod`	`codons`
Arg	6	AGA, AGG, CGA, CGC, CGG, CGT
Leu	6	CTA, CTC, CTG, CTT, TTA, TTG
Ser	6	AGC, AGT, TCA, TCC, TCG, TCT
Ala	4	GCA, GCC, GCG, GCT
Gly	4	GGA, GGC, GGG, GGT
Pro	4	CCA, CCC, CCG, CCT
Thr	4	ACA, ACC, ACG, ACT
Val	4	GTA, GTC, GTG, GTT
Ile	3	ATA, ATC, ATT
Stop	3	TAA, TAG, TGA
Asn	2	AAC, AAT
Asp	2	GAC, GAT
Cys	2	TGC, TGT
Gln	2	CAA, CAG
Glu	2	GAA, GAG
His	2	CAC, CAT
Lys	2	AAA, AAG
Phe	2	TTC, TTT
Tyr	2	TAC, TAT
Met	1	ATG
Trp	1	TGG

4. Running each one of the functions

Here we will analyze in detail the output given by each one of the functions of the package. The idea is to gain a good understanding of the meaning of each one of the components of the output, to be able to use those functions to obtain knowledge about the informational properties derived from the raw frequencies of codons in a determined specie.

For all functions, except "`est.H`" which is general, the functions need as input parameter the number of column, say, "`nc`", of the matrix "`codon.freq`" where the data for the 64 codons of a particular species are located. Table 3 can be consulted to select a specific species to be analyzed, however here we will exemplify the functions with our own specie, humans, which data are in column `nc = 16` of "`codon.freq`".

For each function there is a `Box` presenting the R calculations to be performed. In those boxes we will use a temporary object, named "`temp`" to allocate output. The first command within each box will be to consult the in line help for that function; please, briefly review such help to be introduced to the corresponding function.

4.1. `est.H()`: Estimates informational properties from a vector of relative frequencies.

`est.H()` is a generic function to give informational properties from a vector of relative frequencies. It implements equations (4), "$\hat{H}$". Let's do calculations and discuss the output.

```
------------------------------------------------------------------------------------------
# Box 2.
# Running est.H(). First try.
> ? est.H # See the help for the function.
# Try the function with the data for relative frequencies for the bases in the specie
# Entamoeba histolytica (Amoeba), given in the first row of Table 1:
> est.H(c(0.3969, 0.3226, 0.1568, 0.1237))
         n       Hest          EN       Info      V.est
4.00000000 1.84776256 3.59941527 0.15223744 0.01716277
# Round the output to 4 decimal places:
> round(est.H(c(0.3969, 0.3226, 0.1568, 0.1237)), 4)
     n   Hest     EN   Info  V.est
4.0000 1.8478 3.5994 0.1522 0.0172

# Now, let's see the result of running the function in the case where all 4 bases are
# exactly at the same frequency: 1/4
> est.H(c(1/4, 1/4, 1/4, 1/4))
    n  Hest    EN  Info V.est
    4     2     4     0     0
------------------------------------------------------------------------------------------
```

In `Box 2` we see that the output of `est.H` when using the relative frequencies of the four bases of the species *Entamoeba histolytica* (Amoeba), which were presented in the first row in Table 1 are: `n = 4` which is just the number of elements in the input, i.e., the number of relative frequencies; `Hest = 1.8478` that gives the estimated value of the entropy, $\hat{H}$ as in Table 1. The next element in the vector of results is `EN = 3.5994`. The name "EN" is the acronym for Effective Number, and denotes the effective number of units that are transmitted in a message when the relative frequencies are the ones in the input. `EN = 3.5994` results from $2^{1.8478} \approx 3.5994$, or in general $2^{\hat{H}}$.

The concept of effective number of units is employed in various contexts, for example, in ecology it is used as the *effective number of species* or in population genetics as the *effective number of alleles*, etc. In fact `EN` is just another way to examine the entropy of a system; note that if $\hat{H} = 0$ (one of the symbols is in frequency 1 and the remaining in frequency 0) we have `EN = 1`, telling us that in fact the system is transmitting a single symbol. On the other extreme, when we have the maximum entropy, in this case $H^m = 2$ bits, then `EN = 4` (that is $2^2 = 4$), indicating that the effective number of units is equal to the length of the vector of relative frequencies, i.e., that the channel is transmitting at their maximum capacity or, that the coding is optimal.

The result `Info = 0.1522` gives the information (in bits) that is present in the relative frequencies with reference to the maximum entropy, i.e., as presented in equation (3), `Info =` $\hat{I} = H^m - \hat{H} = 2 - 1.8478 = 0.1522$ As we have previously seen, this term is relevant to evaluate in a quantitative way how the expected frequencies of sequences will vary depending on its base composition.

The last term in the result, `V.est = 0.0172`, is just the estimated variance of the relative frequencies in the input. This quantity is inversely related to $\hat{H}$.

It is important to remember that when using `est.H()` the number of elements of the input vector that are different from zero (`n` in the output) is considered to be the number of symbols that the alphabet has, and that number is used to calculate the maximum entropy in bits, $\log_2(n)$, which in turn is used to calculate the information in the output (`Info`) as a difference between the maximum and estimated entropies.

4.2. `codons2bases()`**: Frequencies of bases in a species.** The `codons2bases()` function is not usually run directly by the user, but is used by other functions to obtain intermediate results. Thus, we only briefly present a pair of examples of its use in `Box 3`.

```
-----------------------------------------------------------------------------------------
# Box 3.
# Running function codons2bases
> ? codons2bases # Seek help about the function.

# Run the function with the defaults
> codons2bases(nc = 16, print.data = TRUE, only.summary = TRUE)

Data for column: 16
Basic information:
   Kingdom      Species     common.name tax.id key
16  Animal Homo sapiens human; primates   9606 H.s
Total of codons: 11,577,026; Total of bases: 34,731,078
Stop codons are present in the data.

            A       T       G       C
F     3086621 1985713 3627359 2877333
S     3597976 2994666 2222416 2761968
T     2280958 2594764 3296783 3404521
Total 8965555 7575143 9146558 9043822
# This matrix presents the number of cases for
# F = First base in the codon, S = Second base in the codon, T = Third base in the codon.
# And the last row is the total of the first three rows.

# Obtain full results for humans (do not print the basic info)
> temp <- codons2bases(nc=16, print.data=F, only.summary=F)
> class(temp)
[1] "list"
> length(temp)
[1] 2
```

```
> names(temp)
[1] "summ.tab" "f.c.b"
> temp$summ.tab # The first element (as above)
            A       T       G       C
F     3086621 1985713 3627359 2877333
S     3597976 2994666 2222416 2761968
T     2280958 2594764 3296783 3404521
Total 8965555 7575143 9146558 9043822

# Check that the fourth row is the sum of the first three:
> apply(temp$summ.tab[1:3,], 2, sum)
      A       T       G       C
8965555 7575143 9146558 9043822
> sum(apply(temp$summ.tab[1:3,], 2, sum)) # Total of bases in the data
[1] 34731078
> sum(temp$summ.tab[4,]) # Total of bases in the data (in another way)
[1] 34731078
# Checking equality using the prod (product) funcion:
> prod(apply(temp$summ.tab[1:3,], 2, sum)==temp$summ.tab[4,])
[1] 1

> temp2 <- temp$f.c.b # Isolate the second component
> class(temp2)
[1] "data.frame"
> dim(temp2)
[1] 64 18
> names(temp2)
 [1] "codon"    "aa"       "fb"       "sb"       "tb"       "freq.cod"
 [7] "fb.A"     "fb.T"     "fb.G"     "fb.C"     "sb.A"     "sb.T"
[13] "sb.G"     "sb.C"     "tb.A"     "tb.T"     "tb.G"     "tb.C"
> temp2[1, ] # See the first row
  codon  aa fb sb tb freq.cod fb.A fb.T   fb.G fb.C sb.A sb.T sb.G   sb.C
1   GCT Ala  G  C  T   213559    0    0 213559    0    0    0    0 213559
  tb.A   tb.T tb.G tb.C
1    0 213559    0    0
> head(temp2[,1:5]) # The first 5 columns
  codon  aa fb sb tb
1   GCT Ala  G  C  T
2   GCC Ala  G  C  C
3   GCA Ala  G  C  A
4   GCG Ala  G  C  G
5   CGT Arg  C  G  T
6   CGC Arg  C  G  C

> temp3 <- apply(temp2[,6:18], 2, sum) # Sum of numeric columns
> temp3
freq.cod     fb.A     fb.T     fb.G     fb.C     sb.A     sb.T     sb.G
11577026  3086621  1985713  3627359  2877333  3597976  2994666  2222416
    sb.C     tb.A     tb.T     tb.G     tb.C
 2761968  2280958  2594764  3296783  3404521
> temp$summ.tab # Compare with the first element...
            A       T       G       C
```

```
F     3086621 1985713 3627359 2877333
S     3597976 2994666 2222416 2761968
T     2280958 2594764 3296783 3404521
Total 8965555 7575143 9146558 9043822
> names(temp3)
 [1] "freq.cod" "fb.A"     "fb.T"     "fb.G"     "fb.C"     "sb.A"
 [7] "sb.T"     "sb.G"     "sb.C"     "tb.A"     "tb.T"     "tb.G"
[13] "tb.C"
> sum(temp3[c(2, 6, 10)]) # The sum of all the A's; must be 8965555
[1] 8965555

> rm(temp, temp2, temp3) # Removes temporal objects
--------------------------------------------------------------------------------------
```

From the content in Box 3 we see that the function `codons2bases()` gives an output that extract the number of bases that exist using the raw frequencies of the 64 codons in an orderly manner. Given that the output are just ordered frequencies of bases within codons, there is not a direct use of those components. Nevertheless, the output of the function is employed as an intermediate step to calculate informational properties at base or codon levels.

4.3. `prop.bases()`: **Relative frequencies and informational properties per bases.** The function `prop.bases()` gives a `data.frame` with the estimates of the relative frequencies as well as the informational properties but segregating the results by the base within the codon. Let's see an example

```
--------------------------------------------------------------------------------------
# Box 4.
# Running function prop.bases()
> ? prop.bases # Ask for help for that function
> temp <- prop.bases(nc=16, print.data=T) # Run for humans

Data for column: 16
Basic information:
   Kingdom      Species     common.name tax.id key
16  Animal Homo sapiens human; primates   9606 H.s
Total of codons: 11,577,026; Total of bases: 34,731,078
Stop codons are present in the data.

> class(temp)
[1] "data.frame"
> dim(temp)
[1] 4 8

# See the results (rounded to 5 decimal places)
> round(temp, 5)
            A       T       G       C    Hest     ENB    Info   V.est
F     0.26662 0.17152 0.31332 0.24854 1.96852 3.91367 0.03148 0.00348
S     0.31079 0.25867 0.19197 0.23857 1.97894 3.94204 0.02106 0.00242
T     0.19702 0.22413 0.28477 0.29408 1.98062 3.94663 0.01938 0.00221
Total 0.25814 0.21811 0.26335 0.26040 1.99593 3.98873 0.00407 0.00046

# Note: This data frame show the relative frequencies and informational properties
# segregating in rows that contain, respectively:
# F = First base in the codon, S = Second base in the codon, T = Third base in the codon.
# And the last row is the total of the first three rows.

# Frequencies of the four bases in the first position of the codon:
> temp[1,1:4]
          A         T         G         C
F 0.2666161 0.1715219 0.3133239 0.2485382
# Calculate informational properties from those frequencies
# i.e., using only bases in the first position of the codon:
> round(est.H(temp[1,1:4]),5)
      n    Hest      EN    Info   V.est
4.00000 1.96852 3.91367 0.03148 0.00348
# Note that those values are also present in
> round(temp[1, 5:8], 5)
     Hest     ENB    Info   V.est
F 1.96852 3.91367 0.03148 0.00348

> cor(temp$Info, temp$V.est) # Note the correlation
[1] 0.9993458

> round(temp[, 5:8], 5) # Data to be interpreted
         Hest     ENB    Info   V.est
F     1.96852 3.91367 0.03148 0.00348
S     1.97894 3.94204 0.02106 0.00242
T     1.98062 3.94663 0.01938 0.00221
Total 1.99593 3.98873 0.00407 0.00046
-----------------------------------------------------------------------------
```

The table above gives us results of informational properties for humans. The rows are labeled as `F, S, T` and `Total`. The first three rows, `F, S, T`, refer to the position of the base within the codon, First, Second and Third, respectively while the last row, `Total`, refer of the total of the bases, without taking into account its codon position.

On the other hand, columns `Hest, ENB, Info` and `V.est` are informational properties. First we have estimated entropy, `Hest` (disorder or uncertainty) and in this case we see that the values increment by row, from 1.96852 bits for `F` up to almost 2, 1.99593 bits for the `Total` row. `Hest` tell us that there are less uncertainty in the first two positions of the codons (`F` and `S`) than in the third position (`T`) or in the total (`Total`).

Related with `Hest` we have the column of Effective Number of Bases, `ENB`, which is just another way to see the estimated entropy, given that `ENB` is equal to 2 to the `Hest` power; for example, for the first row (`F`), $2^{1.96852} \approx 3.91366$, etc. The value of `ENB` tell us how many bases are effectively been transmitted by the messages (genes); taking into account the fact that there are 4 bases in the DNA, we see that for the row `Total` we have almost the maximum possible of 4, say the value 3.98873, while for the other rows the numbers are smaller.

The most important and "interpretable" result is given by the column `Info` = 2 - `Hest`, i.e., the information that is given by the first, second and third codon bases (rows `F, S, T`) as well as for the total (row `Total`). The values of `Info` are 0.03148, 0.02106 and 0.01938 for rows `F, S, T`, respectively. And note that $0.03148 > 0.02106 > 0.01938$; the information decreases from the first to the third codon base. In fact comparing the information of the first position with the second one we have that $0.03148/0.02106 \approx 1.49$ meaning that, for the case of humans, the first base is almost 1.5 more important (informative) than the second one, but comparing the second with the third codon position we have that the second base is only $0.02106/0.01938 \approx 1.09$ times more informative than the third one.

Finally by comparing the information given per codon position (`F, S, T`) with the `Total` information, which does not take into account codon position we have the quotients $(0.03148, 0.02106, 0.01938)/0.00407 \approx 7.73, 5.17, 4.76$ indicating the relative importance of the information given when each codon position is taken into account with reference of not taking into account that information. The average of those quantities, $7.73, 5.17, 4.76$, is 5.89 indicating that in humans when codon position is taken into account there is almost 6 times more information than when codon positions are ignored.

The last column, `V.est` gives the estimated values of the variances of the frequencies, and, in general a smaller variance implies larger information values, but that relation is not lineal, thus it is always better to interpret the values of `Info` as done above.

4.4. `prop.codons()`: **Estimates informational properties at codon level.** If we take genes, or more exactly, mRNA's that code for a protein, we have an alphabet that consist of 64 different "codons" —non-overlapping sequences of three consecutive bases— which constitute the genetic code to translate the original message in the DNA to a sequence of amino acids in a protein. Thus, instead of considering the 4 bases in the DNA, we must consider at informational level the message given in a different alphabet that now consist of 64 different "symbols", the codons. A simile will be illustrative here; in English we have 27 symbols which can form many words (of different length); thus we could consider informatics properties at symbol (letter) level or, much more complex, consider each word as an individual symbol and so considering an "alphabet" with around 170,000 "symbols", i.e., the words currently in use.

By considering each one of the 64 codons as a symbol, the maximum entropy, H^m in equation (2) is $H^m = \log_2(64) = 6$ and information, $\hat{I}$ (3), can be measured as the difference $\hat{I} = 6 - \hat{H}$. However, the value of $\hat{H}$ in the previous equation could be obtained under different hypotheses or scenarios, say

$\hat{H}_r$ **:** Use the relative frequencies of codons in the data; results in $\hat{I}_r = 6 - \hat{H}_r$ (denoted as `ObsFreq` in the function output).

$\hat{H}_b$ **:** Use the expected frequencies of codons using the frequencies of bases in the data; results in $\hat{I}_b = 6 - \hat{H}_b$ (denoted as `ExpBase` in the function output).

$\hat{H}_c$ **:** Use the expected frequencies of codons but considering the positions of bases within codons and using the frequencies of bases in the data; results in $\hat{I}_c = 6 - \hat{H}_c$ (denoted as `ExpBinC` in the function output).

The values of the three different estimates of information in the codon case, $\hat{I}_r$, $\hat{I}_b$ and $\hat{I}_c$ consider different informational properties and thus need to be interpreted differently.

$\hat{I}_r$ (`ObsFreq`) is calculated from the relative frequencies of the 64 codons in the original data, and thus tell us how much information (in bits) is given in average by the genes of the species of interest with regard to the maximum possible entropy. In other words, $\hat{I}_r$ tell us how much the distribution of the codons deviate from the hypothetical case where each one of the codons is present at the same relative frequency, $1/64 = 0.015625$

$\hat{I}_b$ (`ExpBase`) is calculated by taking into account the frequencies of the 4 DNA bases in the species, and using those frequencies the expected relative frequencies of each one of the 64 codons are calculated. This calculations imply a step ahead of the ones done for $\hat{I}_r$ —which assumed the same frequency of codons and also of bases in the gene of the species. $\hat{I}_b$ tell us how much information is due to the codon distribution, having adjusted by the true frequencies of the bases in the species.

Finally, $\hat{I}_c$ (`ExpBinC`) takes the calculations one step further than the ones used for $\hat{I}_b$; to estimate $\hat{I}_c$ we do not only take into account the base's frequencies, but also the positions of those bases within the codons.

`Box 5` exemplifies the use and interpretation of the function `prop.codons()` using human data first and, for contrast data of amoeba.

```
--------------------------------------------------------------------------------
# Box 5.
# Using function prop.codons()
> ? prop.codons # Ask help about that function.

# Obtain results for humans (nc=16) in a temporal object.
> temp <- prop.codons(nc=16)
> class(temp)
[1] "list"
> length(temp)
[1] 2
> names(temp)
[1] "H.est"       "frequencies"
> temp$H.est # Or, equivalently, temp[[1]]
            Hest      ENC       Info        V.est
ObsFreq 5.787137 55.22068 0.21286313 7.027120e-05
ExpBase 5.987789 63.46057 0.01221146 4.099026e-06
ExpBinC 5.928085 60.88796 0.07191494 2.492463e-05
> head(temp$frequencies) # Or head(temp[[2]])
    codon  aa ObsFreq  ExpBase  ExpBinC
GCT   GCT Ala  213559 173158.1 193960.4
GCC   GCC Ala  323249 206730.2 254490.3
GCA   GCA Ala  187108 204941.1 170503.2
GCG   GCG Ala   89097 209078.6 246436.8
CGT   CGT Arg   52129 173158.1 123799.6
CGC   CGC Arg  119972 206730.2 162434.2

# Obtain relative frequencies under different hypotheses:
> temp.fr <- temp$frequencies[,3:5]/apply(temp$frequencies[,3:5], 2, sum)
> head(temp.fr, 2)
       ObsFreq    ExpBase    ExpBinC
GCT 0.01844679 0.01495704 0.01675390
GCC 0.02792159 0.01785693 0.02198235
> apply(temp.fr, 2, sum) # Check that are relative frequencies
ObsFreq ExpBase ExpBinC
      1       1       1

# And see the results of est.H for each case:
> est.H(temp.fr$ObsFreq)
           n         Hest           EN         Info        V.est
6.400000e+01 5.787137e+00 5.522068e+01 2.128631e-01 7.027120e-05
> est.H(temp.fr$ExpBase)
           n         Hest           EN         Info        V.est
6.400000e+01 5.987789e+00 6.346057e+01 1.221146e-02 4.099026e-06
> est.H(temp.fr$ExpBinC)
           n         Hest           EN         Info        V.est
6.400000e+01 5.928085e+00 6.088796e+01 7.191494e-02 2.492463e-05
> temp[[1]] # Compare with results of function est.H:
            Hest      ENC       Info        V.est
ObsFreq 5.787137 55.22068 0.21286313 7.027120e-05
ExpBase 5.987789 63.46057 0.01221146 4.099026e-06
ExpBinC 5.928085 60.88796 0.07191494 2.492463e-05
```

```
> temp.fr <- temp$frequencies[,3:5]/apply(temp$frequencies[,3:5], 2, sum)
> cor(temp.fr)
          ObsFreq   ExpBase   ExpBinC
ObsFreq 1.0000000 0.2515532 0.6084043
ExpBase 0.2515532 1.0000000 0.4038376
ExpBinC 0.6084043 0.4038376 1.0000000
> summary(temp.fr)
    ObsFreq              ExpBase            ExpBinC
 Min.   :0.0003832   Min.   :0.01038   Min.   :0.006487
 1st Qu.:0.0109385   1st Qu.:0.01466   1st Qu.:0.011923
 Median :0.0147224   Median :0.01496   Median :0.015116
 Mean   :0.0156250   Mean   :0.01562   Mean   :0.015625
 3rd Qu.:0.0190933   3rd Qu.:0.01767   3rd Qu.:0.018605
 Max.   :0.0401455   Max.   :0.01826   Max.   :0.028636

# Next figure in text (for humans)
plot(temp.fr$ObsFreq, temp.fr$ExpBase, ylim=c(0.005, 0.03), xlab="Observed codon frequency",
     ylab="Expected codon frequency", pch=15, col="black")
points(temp.fr$ObsFreq, temp.fr$ExpBinC, pch=19, col="red")
legend("topleft", bg="white", legend=c("Bases", "Position"),
     pch=c(15, 19), col=c("black", "red"))
grid()
abline(h=c(0.0175, 0.015, 0.0125, 0.01), lty=2,
     col=c("darkblue", "darkgreen", "darkorange", "grey"))
legend("bottomright", bg="white",
legend=c("0.0175", "0.0150", "0.0125", "0.0100"), lty=2,
col=c("darkblue", "darkgreen", "darkorange", "grey"))
# Note the intriguing band pattern, marked with horizontal lines.

# A second example with the amoeba (Entamoeba histolytica) nc=10
> temp <- prop.codons(nc=10)
> temp[[1]]
            Hest      ENC      Info        V.est
ObsFreq 5.157767 35.69790 0.8422327 0.0003242247
ExpBase 5.543250 46.63206 0.4567499 0.0001869895
ExpBinC 5.359205 41.04702 0.6407946 0.0002459722
> temp.fr <- temp$frequencies[,3:5]/apply(temp$frequencies[,3:5], 2, sum)
> summary(temp.fr)
    ObsFreq              ExpBase              ExpBinC
 Min.   :0.0001363   Min.   :0.001893   Min.   :0.0009764
 1st Qu.:0.0020251   1st Qu.:0.006257   1st Qu.:0.0036976
 Median :0.0085977   Median :0.011315   Median :0.0083194
 Mean   :0.0156250   Mean   :0.015625   Mean   :0.0156250
 3rd Qu.:0.0230924   3rd Qu.:0.020074   3rd Qu.:0.0214081
 Max.   :0.0732798   Max.   :0.062532   Max.   :0.0620673
> cor(temp.fr)
          ObsFreq   ExpBase   ExpBinC
ObsFreq 1.0000000 0.7096372 0.8535145
ExpBase 0.7096372 1.0000000 0.8563833
ExpBinC 0.8535145 0.8563833 1.0000000
```

```
# Plot in text for amoeba.
plot(temp.fr$ObsFreq, temp.fr$ExpBase, ylim=c(0.0009, 0.065),
     xlab="Observed codon frequency", ylab="Expected codon frequency",
     pch=15, col="black")
points(temp.fr$ObsFreq, temp.fr$ExpBinC, pch=19, col="red")
legend("topleft", bg="white", legend=c("Bases", "Position"),
     pch=c(15, 19), col=c("black", "red"))
grid()
------------------------------------------------------------------------
```

For the example with data from humans we see that $\hat{I}_r \approx 0.2129$ (`Info` in row `ObsFreq`), $\hat{I}_b \approx 0.0122$ (`Info` in row `ExpBase`) and $\hat{I}_c \approx 0.0719$ (`Info` in row `ExpBinC`) and ordering those values from smaller up to larger we have that $\hat{I}_b < \hat{I}_c < \hat{I}_r$; $(0.0122 < 0.0719 < 0.2129)$. This implies that by taking for calculations the true frequencies of the bases in humans we only gain an information of $\hat{I}_b \approx 0.0122$ bits in average, while when taking also codons position into account the gain is of $\hat{I}_c \approx 0.0719$ in average. The ratio of those two quantities, $0.0719/0.0122 \approx 6$, means that taking codons positions improves in around 6 times our prediction ability with reference of taking only bases frequencies into account. On the other hand, the information gained by knowing the true frequencies of the bases is $\hat{I}_r \approx 0.2129$, implying that there is a large gain of information by considering the true relative frequencies of the codons, instead of the homogeneous relative frequencies of 1/64 we gain in more than 0.2 bits (in average per codon). The quantity $\hat{I}_r \approx 0.2129$ is also a measure of how far is the true distribution of the codons in contrast with the homogeneous distribution.

For the example with data from the amoeba we see that $\hat{I}_r \approx 0.8422$ (`Info` in row `ObsFreq`), $\hat{I}_b \approx 0.4568$ (`Info` in row `ExpBase`) and $\hat{I}_c \approx 0.6407$ (`Info` in row `ExpBinC`). First note that the values of information for amoeba are much larger than for humans, in fact we have for $\hat{I}_r, \hat{I}_b$ and $\hat{I}_c$ such ratios are $(0.8422, 0.4568, 0.6407)/(0.2129, 0.0122, 0.0719) \approx (4, 37, 9)$ or taking the average of those values we get an average ratio of 17; i.e., in average we have more information in the case of the amoeba, and that is a result of the fact that the amoeba has a frequency of DNA bases (and codons) that is far away from homogeneity, i.e., from the case which each DNA base as well as each codon have the same frequency.

Now we can analyze Figure 3 which present the plot of the relative and relative expected values for each one of the 64 codons in the case of humans.

In Figure 3 we can see a dot plot of the observed relative frequencies of codons (relative `ObsFreq`) in the X-axis by the expected relative codon frequencies when taking the true frequencies of bases (relative `ExpBase`, black squares), and the expected relative frequencies (relative `ExpBinC`, red circles) in the Y-axis for our own species, *Homo sapiens*. In Figure 3 the codon with a smaller relative frequency, relative `ObsFreq` ≈ 0.0004, corresponds to "TAG" that codes for Stop, while the largest relative `ObsFreq` ≈ 0.04, corresponds to "GAG" that codes for the amino acid Glu. The median of relative `ObsFreq` is ≈ 0.015 and the mean is approximately at the same value.

In Figure 3 we observe the relations that exist between the observed relative frequency of the codons (X-axis) and the relative frequencies expected by taking into account only the base frequencies (black squares) and taking into account also the codon position of the bases (red circles). In neither case there is a strong linear relation between the relative `ObsFreq` with the frequencies expected taking into account bases (relative `ExpBase`, black squares) or also codon position of those bases (relative `ExpBinC`, red circles); in the first case Pearson's correlation coefficient is $\hat{r} \approx 0.25$ ($\hat{r}^2 \approx 0.06$), and in the second $\hat{r} \approx 0.61$ ($\hat{r}^2 \approx 0.37$); this means that the observed frequency of codons cannot be fully explained just by the frequencies of bases or bases within codons. An interesting horizontal banding pattern is observed in the squares in Figure 3 (such pattern is remarked by the dashed horizontal lines) and that pattern was due to the relative abundances of the four DNA bases.

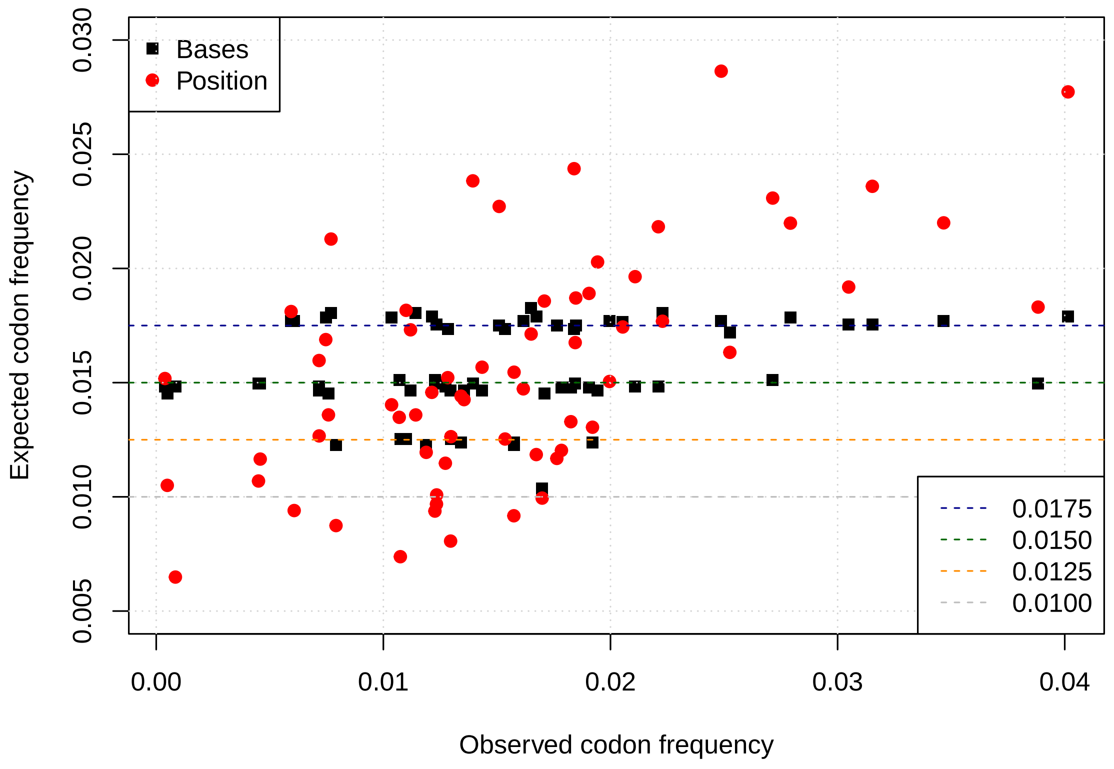

Figure 3. Plot of relative frequency of codons (relative `ObsFreq`) in the X-axis, by the expected relative codon frequencies when taking the true frequencies of bases (relative `ExpBase`, black squares), and the expected relative frequencies (relative `ExpBinC`, red circles) in the Y-axis for humans, *Homo sapiens*.

Figure 4 presents a figure homologous to Figure 3, but instead of data for humans we used the data of the amoeba, which are the ones that present a larger difference with the homogeneous frequency of the four DNA bases (1/4). In this case the correlations between observed and expected frequencies are larger than in humans, $\hat{r} \approx 0.71$ ($\hat{r}^2 \approx 0.50$) for the values of observed and expected by bases (black squares) and $\hat{r} \approx 0.85$ ($\hat{r}^2 \approx 0.73$) for observed and expected by bases and codons (red circles). However, those large correlation values are strongly affected by some large values in both observed and expected values, for example the almost overlapped black square and red circle in the upper right hand side corner of the plot. That point corresponds to the codon "AAA" which is one of the two codons ("AAA" and "AAG") that code for the amino acid Lys. In turn, this high frequencies are explained in part by the fact that the largest base frequency in the genes of the amoeba is for "A", with relative frequency ≈ 0.3969, highly larger than the expected neutral frequency $1/4 = 0.25$

In summary, the results of the function "`prop.codons()`" allows detailed analyses of the informational properties of codon frequencies and their expected frequencies under different hypotheses.

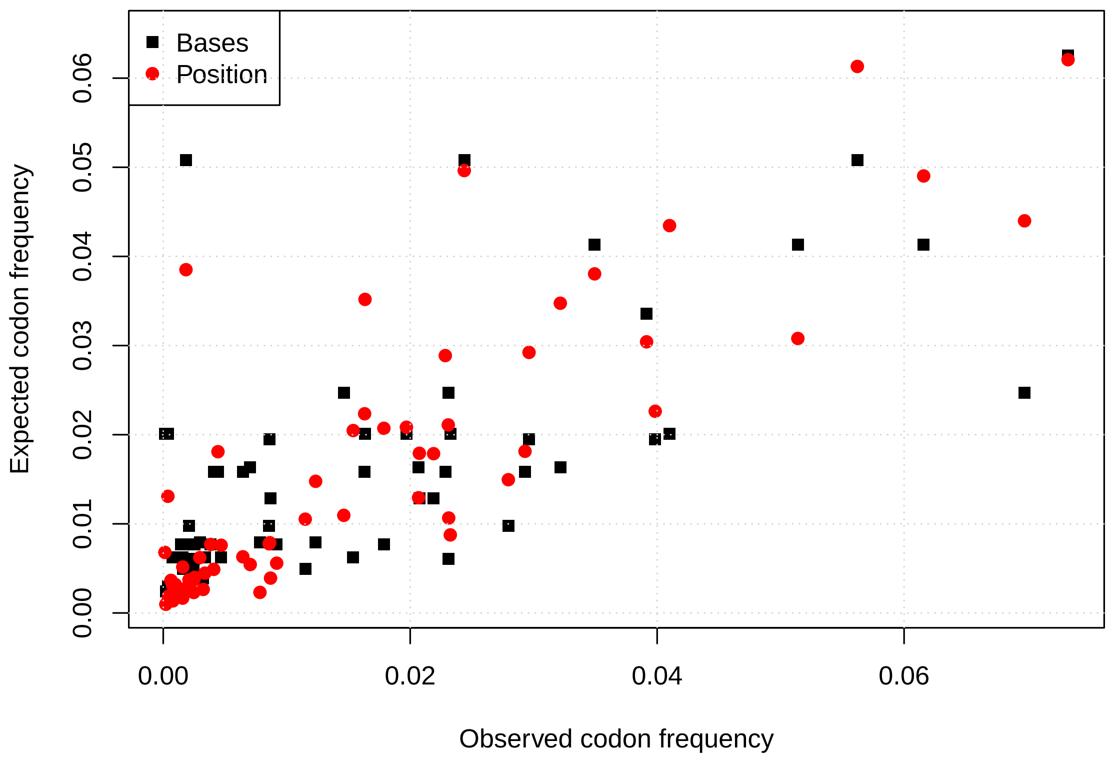

FIGURE 4. Plot of relative frequency of codons (relative `ObsFreq`) in the X-axis, by the expected relative codon frequencies when taking the true frequencies of bases (relative `ExpBase`, black squares), and the expected relative frequencies (relative `ExpBinC`, red circles) in the Y-axis for amoeba, *Entamoeba histolytica*.

4.5. `codon.bias()`: **Informational properties for amino acids coded by more than one codon.** The genetic code consists in the encoding of 21 different "signals" —corresponding to the 20 amino acids plus the "Stop" —by an alphabet consisting of 64 codons. That is why it is said that the genetic code is "degenerated", meaning basically that various codons can result in the same amino acid, or, in another words that for some cases are synonymous codons for the same amino acid or for the Stop signal.

In here we are going to speak about the 19 cases where there are more than one possibility of coding and, to simplify notation we will be considering the "Stop" signal as one "amino acid" more.

In that context emerges the concept of "Codon Bias" as a measure of the heterogeneity of codon use, or more exactly, as a measure of the preference of an specie to use a given codon in favor of other. It is desirable to have a codon bias measure that will vary between 0 and 1, being equal to 0 only when all the posible codons are used in exactly in same frequency and reaching the maximum value of 1 when only one of the possible codons is used.

Codon Bias is relevant only for the cases where more than one codon could be used to code for the amino acid, thus the cases of "Met" and "Trp" (coded by a single codon) are not relevant, and we need to consider only the set of the 19 signals (18 amino acids plus "Stop"):

$S = \{$"Ala", "Arg", "Asn", "Asp", "Cys", "Gln", "Glu", "Gly", "His", "Ile", "Leu", "Lys", "Phe", "Pro", "Ser", "Stop", "Thr", "Tyr", "Val"$\}$.

The set S can be partitioned into sub-sets taking into account the number of codons, say k, which code for the amino acid, thus we obtain Table 5

TABLE 5. Segregation of S by number of codons, k.

k	n_k	S_k: Set of amino acids coded by k codons:
2	9	$S_2 = \{$"Asn", "Asp", "Cys", "Gln", "Glu", "His", "Lys", "Phe", "Tyr"$\}$
3	2	$S_3 = \{$"Ile", "Stop"$\}$
4	5	$S_4 = \{$"Ala", "Gly", "Pro", "Thr", "Val"$\}$
6	3	$S_6 = \{$"Arg", "Leu", "Ser"$\}$

Table 5 shows how the set S is partitioned into 4 disjoint subsets, S_2, S_3, S_4 and S_6 ($S = S_2 \cup S_3 \cup S_4 \cup S_6$), which contain $n_k = 9, 2, 5$ and 3 amino acids, respectively. Note that

$$\sum_k k \times n_k = (2 \times 9) + (3 \times 2) + (4 \times 5) + (6 \times 3) = 62$$

which is the number of codons coding for more than one amino acid, because we are excluding codons "AGT" and "TGG" which code for amino acids "Met" and "Trp", respectively.

It is convenient to have a measure of codon bias that will give us values in the interval $[0, 1]$, in such a way that the value of 0 (minimum codon bias) will indicate a full equilibrium of the relative frequencies of the k codons that code for a given amino acid, and that happens only when all k codons are used in the same frequency, $1/k$, in the amino acid studied (in a specific species), while it will take the value of 1 (maximum codon bias) only when a single codon –of the k available, is employed by the organism.

A coefficient that fulfill such requirements is given by

$$B_{ik} = \frac{k - E_{ik}}{k - 1} \tag{5}$$

where B_{ik} is the codon bias for the ith amino acid coded by exactly k possible codons and E_{ik} is the "Effective Number of Codons" inferred from the data (see below for a definition).

The subindexes i, k in equation (5) can take a total of 19 different values. First, in Table 5 we see that k could be $2, 3, 4$ or 6, while i (given k) can take values $1, 2, \cdots n_k$. For example, for $k = 3$ we see in Table 5 that the amino acids coded by exactly 3 codons are in the set $S_3 = \{$"Ile", "Stop"$\}$, thus we will have coefficients B_{ik} that could be denoted by $B_{Ile\ 3}$ and $B_{Stop\ 3}$ respectively (even when this notation is somehow redundant). Other way to denote that is to agree in that $i = 1,\ k = 3$ will denote the first amino acid in the set S_3 ("Ile"), while $i = 2,\ k = 3$ will denote the second one, i.e., "Stop", etc. Even when this notation could be considered too convoluted at first, in computational practice we can always use only the abbreviation of the amino acid name, which immediately determine the value of k, given by the genetic code, i.e., when we locate an amino acid in one of the rows of Table 5 we immediately know how many codons (k) code for that amino acid.

Now we will see that all values of $B_{ik} \in [0,\ 1]$, that is that all values of the coefficient will be between zero and one. To see that consider that E_{ik} denotes the "*Effective Number of Codons*", defined as

$$E_{ik} = 2^{H_{ik}}$$

where H_{ik} is the entropy for amino acid i which is coded by k codons and depends only on the vector of relative frequencies for the n_k codons for amino acid i, say on the vector $\mathbf{p}_{ik} = (p_{1k}, p_{2k} \cdots p_{n_k})$. To clarify let's see an example. The amino acid "Gln" is coded by two codons, "CAA" or "CAG". In humans the relative frequencies of those two codons are

$$\mathbf{p}_{Gln} \approx (0.2702, 0.7298)$$

for "CAA" and "CAG", respectively. Thus we have that the entropy associated with those relative frequencies, say H_{Gln} is

$$H_{Gln} \approx -(0.2702 \times log_2(0.2702) + 0.7298 \times log_2(0.7298)) \approx 0.8418$$

(keeping all decimal places we obtain a more more precise result $H_{Gln} = 0.8417565$). Now we can calculate the Effective Number of Codons used in our species to code for "Gln", say E_{Gln} as

$$E_{Gln} \approx 2^{0.8417565} = 1.792231$$

This figure means that, even when two codons can code for "Gln" our species is using only approximately 1.79 *effective* codons when coding that amino acid. Finally we obtain the bias for the case of "Gln" substituting in equation (5) the values $k = 2$ and $E_{Gln} = 1.792231$, say

$$B_{Gln} = \frac{2 - 1.792231}{2 - 1} = 0.207769$$

and that value is the largest bias for the 19 possible amino acids in humans (see `Box 6`).

To see in which cases we will have $B_{ik} = 0$, note that H_{ik} will take its maximum value, $H_{ik} = \log_2(k)$, **only** when each one of the k relative frequencies used to obtain H_{ik} is identical to $1/k$ and in that case we will have $H_{ik} = \log_2(k)$ –se equation (2). Now, if $H_{ik} = \log_2(k)$ we will have that

$$E_{ik} = 2^{H_{ik}} = 2^{\log_2(k)} = k$$

Substituting $E_{ik} = k$ in equation (5) we obtain

$$B_{ik} = \frac{k - E_{ik}}{k - 1} = \frac{k - k}{k - 1} = 0$$

or in words, we will obtain a codon bias equal to 0 only when the relative frequencies of each one of the k codons are identical to $1/k$, i.e., only when we have a perfectly balanced use of the k codons.

Now we need to consider which will be the maximum value of B_{ik}. Conceptually, the bias, B_{ik}, must be maximum when one of the relative frequencies for the k codons is equal to 1 and all the other $k - 1$ relative frequencies are equal to 0, because in that case we have the most unbalanced use of codons. If that is the case we will have that $H_{ik} = 0$ and in consequence $E_{ik} = 2^0 = 1$ and in that case

$$B_{ik} = \frac{k - E_{ik}}{k - 1} = \frac{k - 1}{k - 1} = 1$$

or in words, when only one codon of the k ones which code for an amino acid is used, the codon bias coefficient takes its maximum value of 1.

Figure 5 shows the plot of $H_{i\ 2}$ and $B_{i\ 2}$ as function of the relative frequency of the first codon in the cases where two codons code for an amino acid (second sub-index $k = 2$ in the functions).

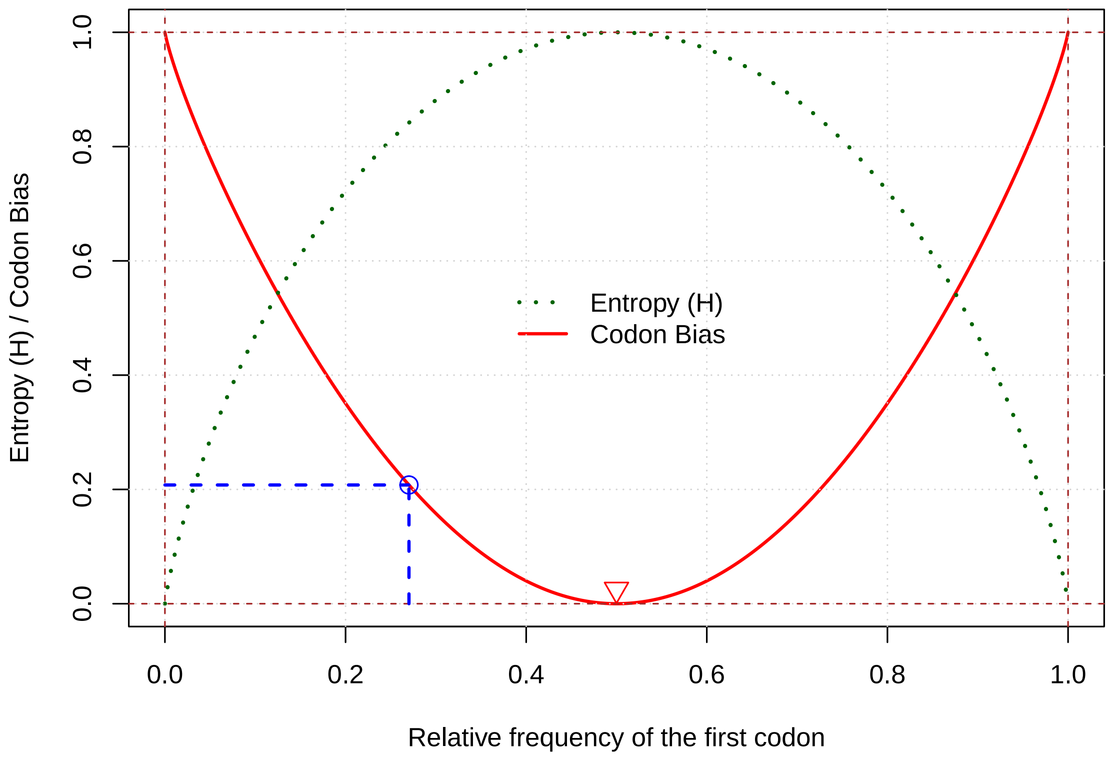

FIGURE 5. Entropy ($H_{i\ 2}(p)$, dotted green line) and Codon Bias ($B_{i\ 2}(p)$, continuous red line) in the Y-axis, as function of the relative frequency of the first codon (p in the X-axis), for the cases of amino acids coded by two codons (second sub-index $k = 2$ in both functions). The red inverted triangle points to the minimum value of $B_{i\ 2} = 0$, which is reached at $X = 1/2$. Blue dashed lines and circle give the coordinates for the example with the amino acid "Gln" in humans where relative frequency of the first codon is $X = 0.2702$ which gives codon bias $Y = B_{Gln} \approx 0.2078$

In the case $k = 2$ both, codon bias ($B_{i\ 2}$) and entropy for codon use ($H_{i\ 2}$) for a given amino acid i can be considered as function of a single parameter, say the relative frequency of the first codon, $p \in [0, 1]$, given that the relative frequency of the other codon is fixed as $1 - p$. Then it is justified (in this case only) to write $B_{i\ 2}(p)$ and $H_{i\ 2}(p)$. For $k > 2; k = 3, 4, 6$ both, B_{ik} and entropy for codon use, H_{ik} are functions of $k - 1$ parameters, given than one of the relative frequencies is fixed by the equality $\sum_i p_i = 1$.

Figure 5 shows that for $k = 2$ codon bias, $B_{i\ 2}(p)$ (red continuous line), and entropy for codon use, $H_{i\ 2}(p)$ (green dotted line), have an inverse behavior in the sense that the minima and maxima occur at the same values of the frequency of the first codon, say $p = 0, 0.5$ and 1, i.e., $\min_p(H_{i\ 2}(p)) = 0$ occurs at two points, $p = 0$ and $p = 1$, while at $p = 0.5$ we have $\min_p(B_{i\ 2}(p)) = B_{i\ 2}(0.5) = 0$ and $\max_p(H_{i\ 2}(p)) = H_{i\ 2}(0.5) = 1$. In fact, there is an almost inverse lineal relation between $B_{i\ 2}(p)$ and $H_{i\ 2}(p)$ with a Pearson's correlation coefficient $r \approx -1$ (not shown).

Note that for the cases where the number of codons that code the amino acid, k, are larger than $k = 2$, i.e., for the cases $k = 3, 4, 6$ it is not possible to make a bi-dimensional plot as the one presented in Figure 5. That is so because for $k = 3, 4, 6$ the bias, B_i, depends on 3, 4 or 6 relative frequencies, respectively, and thus the change in B_i cannot be represented in a bi-dimensional space.

Box 6 presents an example of the use of the function `codon.bias()`.

```
--------------------------------------------------------------------------------------
# Box 6.
# Examples of function codon.bias()
> ? codon.bias # Help for the function

# lets calculate main results for our specie (Homo sapiens)
# cb.Hs short for "Codon Bias Homo sapiens", we will use
# only the main parameters:
> cb.Hs <- codon.bias(nc=16, only.main=TRUE)

# See generalities of the results
> class(cb.Hs)
[1] "data.frame"
> dim(cb.Hs)
[1] 19  5
> names(cb.Hs)
[1] "aa"         "n.cod"      "H.per.aa"   "ENC"        "CodBias.aa"
> head(cb.Hs, 2) # See the first 2 rows.
   aa n.cod H.per.aa      ENC CodBias.aa
1 Ala     4 1.872992 3.662915 0.11236154
2 Arg     6 2.501453 5.662555 0.06748894
> summary(cb.Hs[,2:5])
     n.cod          H.per.aa           ENC          CodBias.aa
 Min.   :2.000   Min.   :0.8418   Min.   :1.792   Min.   :0.001361
 1st Qu.:2.000   1st Qu.:0.9951   1st Qu.:1.993   1st Qu.:0.010492
 Median :3.000   Median :1.4903   Median :2.810   Median :0.067489
 Mean   :3.263   Mean   :1.5031   Mean   :3.064   Mean   :0.067256
 3rd Qu.:4.000   3rd Qu.:1.9096   3rd Qu.:3.757   3rd Qu.:0.091400
 Max.   :6.000   Max.   :2.5015   Max.   :5.663   Max.   :0.207769

> table(cb.Hs$n.cod) # How many rows per number of codons
2 3 4 6
9 2 5 3
# And we are reminded that, of the 19 amino acids coded by more than
# one codon we have 9 that are coded by 2 codons, etc.

 > apply(cb.Hs[,2:5], 2, sd) # S per variable
     n.cod   H.per.aa        ENC CodBias.aa
1.48481594 0.57346512 1.29157156 0.06478971

# And now let's calculate the Coefficient of variation (CV)
# for each one of the variables:
> apply(cb.Hs[,2:5], 2, sd)/apply(cb.Hs[,2:5], 2, mean) # Coefficient of Variance
     n.cod   H.per.aa        ENC CodBias.aa
 0.4550242  0.3815111  0.4215673  0.9633366

# See which aa give the minimum and maximum of CodBias.aa
# With the minimum CodBias.aa we have
> cb.Hs[cb.Hs$CodBias.aa==min(cb.Hs$CodBias.aa),]
   aa n.cod H.per.aa      ENC  CodBias.aa
```

```
3 Asn     2 0.999018 1.998639 0.001360884
# and with the maximum
> cb.Hs[cb.Hs$CodBias.aa==max(cb.Hs$CodBias.aa),]
   aa n.cod  H.per.aa      ENC CodBias.aa
6 Gln     2 0.8417565 1.792231  0.2077692

# Let's see how the number of codons affects the
# mean of the Effective Number of Codons (ENC)
> tapply(cb.Hs$ENC, cb.Hs$n.cod, mean)
       2        3        4        6
1.968585 2.821110 3.720192 5.416857
# Note that mean of ENC is close to n.cod, you could also try
> hist(cb.Hs$ENC) # Not shown in main text

# And thus the strong linear relation between those two variables
> cor.test(cb.Hs$ENC, cb.Hs$n.cod)
Pearson's product-moment correlation
data:  cb.Hs$ENC and cb.Hs$n.cod
t = 37.878, df = 17, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9844297 0.9977920
sample estimates:
      cor
0.9941276

# And see the dot plot of those two variables:
> plot(cb.Hs$n.cod, cb.Hs$ENC) # Not shown in main text

# Now, let's calculate the lineal correlation coefficient
# (Pearson's product-moment correlation) for all
# pairs of numerical variables,
> round(cor(cb.Hs[,2:5]), 4)
            n.cod H.per.aa    ENC CodBias.aa
n.cod      1.0000   0.9845 0.9941     0.5180
H.per.aa   0.9845   1.0000 0.9886     0.4635
ENC        0.9941   0.9886 1.0000     0.4449
CodBias.aa 0.5180   0.4635 0.4449     1.0000

# Let's make a dot plot of all numeric variables
# (Figure 6 in main text)
> plot(cb.Hs[,2:5])
-----------------------------------------------------------------------------------------
```

In `Box 6` we see the main results of the function `codon.bias()` when run with the data for our own species. Command "`summary(cb.Hs[,2:5])`" gives us the summary of all the numeric variables, "`n.cod`" –the number of codons that above we called k and which has values 2, 3, 4 or 5 (see result of "`table(cb.Hs$n.cod)`"), "`H.per.aa`" –The entropy per amino acid that we have denoted by $H_{i\ k}$, the Effective Number of Codons, "`ENC`" –denoted above by E_{ik}, and finally the main result, "`CodBias.aa`" which in our notation is B_{ik}.

As stressed in the abstract, here we will not discuss the biological implications of the values of the variables; that will be done in a manuscript in preparation putatively entitled "*Sampling informational* *properties of codon use through the tree of life*"; however, it appears that interesting and relevant biological knowledge could be obtained from the use of the "`codon.bias()`" function..

In `Box 6` we see that the variable with a higher value of the Coefficient of Variation or "CV" is the codon bias coefficient ("`CodBias.aa`") with a value near to one, CV(`CodBias.aa`) = 0.9633366, that is more than two times larger than the CV's of the other 3 variables ("`n.cod`, "`H.per.aa` and "`ENC`"). This fact can be interpreted in the sense that the codon bias coefficient set apart (discriminates) each one of the 19 amino acids better than any other of the three variables. That is corroborated by estimating all pairs of Pearson's product-moment correlations, $\hat{r}$, resulting from command "`cor(cb.Hs[,2:5])`". By examining that matrix we see that the lower correlations, 0.5180, 0.4635 and 0.4449 result from the pairs containing the variable `CodBias.aa`, while the pairs that do not contain that variable are highly correlated between them, with a minimum value > 0.98 This means that the codon bias coefficient is summarizing in a non-lineal way the values of the other three variables, to which it is functionally related (see equation (5)). Figure 6 present a dot plot of all pairs of variables, resulting from the command "`plot(cb.Hs[,2:5])`".

In Figure 6 we can corroborate the very strong lineal relations that exist between the pairs of variables that do not include `CodBias.aa`. In contrast, the dot-plots including `CodBias.aa` (last column or last row of boxes in Figure 6) present a more disperse and less lineal relation. The set of almost overlapped points in the down left hand side corner of those three boxes correspond to the 9 amino acids coded by two codons ($k = 2$), which obviously have the lowest values of "`n.cod`", "`H.per.aa`" and "`ENC`" in the data set.

5. Further data and analyses

Functions in the R package **CodonInfo** allow a direct analysis of intra-species informational properties of codon use. Specialists in a given species can surly extract interesting biological conclusions from the results of those functions. However, inter-species analyses appear more appealing to the non-specialist, given that sets of 2 or more species could be compared about the informational properties of codon use. That will be boarded in the manuscript in preparation with working title "*Sampling informational properties of codon use through the tree of life*".

As previously mentioned, a limitation of the package is the inclusion of only 35 species arbitrarily selected from the immense number of possibilities. The number of species sampled can be increased by downloading more files with raw data of codon frequencies from the web site mentioned in (Subramanian et al., 2022), or possibly from another sources. I am grateful to the research team that published Subramanian et al. (2022) by making publicly available the Codon Statistics Database, from which the data in this R package were obtained. I am preparing a guide to include more data in the package, the document "`IncludeFurtherDataInCodonInfo.pdf`" which you will find in the same web address (URL) from which you downloaded the package **CodonInfo**.

Finally, I will very much welcome the possibility to collaborate with researchers interested in this topic; if that is your case, please send me an e-mail to "octavio.martinez@cinvestav.mx" with subject "**Collaboration in codon use**".

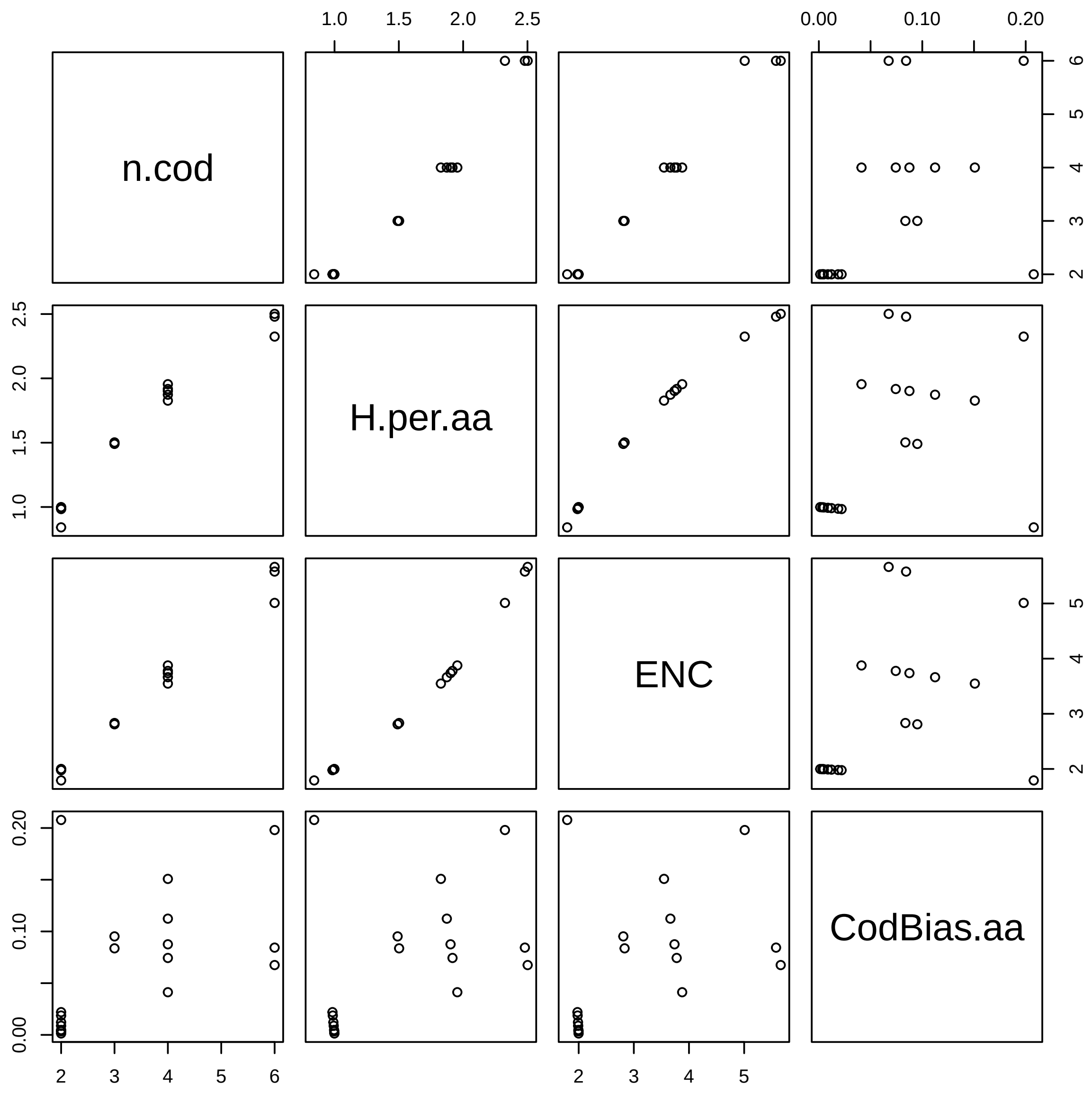

FIGURE 6. Dot plot for all pairs of numeric variables in the object "`cb.Hs`" (main results of the function "`codon.bias()`" for humans (19 points per sub-plot).

References

Adami C (2004) Information theory in molecular biology. *Physics of Life Reviews*, 1, 3–22.

Adami C (2024) *The Evolution of Biological Information: How Evolution Creates Complexity, from Viruses to Brains.* Princeton University Press.

R Core Team (2013) *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.r-project.org`.

Shannon CE (1948) A mathematical theory of communication. *The Bell System Technical Journal*, 27, 379–423.

Subramanian K, Payne B, Feyertag F, and Alvarez-Ponce D (2022) The codon statistics database: a database of codon usage bias. *Molecular Biology and Evolution*, 39, msac157.