ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF STATISTICAL SCIENCES

Second Cycle Degree in Statistical Sciences

CIRCULAR CODES, READING FRAMES AND ERROR CORRECTION IN TRANSLATION

Presented by:
Paolo Dalena
University code: 931888

Supervisor:
Prof. Simone Giannerini
Co-supervisor:
Prof. Lutz Strüngmann

II SESSION

ACADEMIC YEAR 2020/2021

A Luigi Mastrangelo.

Abstract

The presence of error correction mechanisms involved in translation has been ascertained but their elucidation requires a mathematical framework which is still missing. *Comma-free codes* are synchronizable error correcting codes that were introduced by Sir Francis Crick in 1957 to tackle the difficult problem of frame retrieval during translation. Despite its appeal the proposal was discarded but in 1996, thanks to a large scale exploratory analysis of coding sequences, a weaker form of comma-free codes, called *circular codes*, were hypothesized to be involved in the translation machinery. Recent works established a connection between circular codes and group theory and identified a set of 216 circular codes possessing desirable mathematical properties. These, in turn, can be partitioned into 27 equivalence classes according to the 8 nucleotide transformations linked to the dihedral group of symmetry. The *coverage* of a circular code is a measure of its *compliance* with a specific sequence or organism. It has universal properties, is strongly correlated with translation accuracy and behaves differently in the initial and final parts of the sequences. This agrees with the molecular biology of the translation process and raises interesting questions. This thesis moves from these results and studies both the codon usage and the code coverage in 24 different organisms. The analyses are carried out in the R environment and part of the work has been devoted to collaborating at the development of the R package `mathDNA`, which implements some of the functions used in the analysis. The results confirm the different behavior in terms of code coverage and codon usage at the beginning and at the end of the sequences and proves that there is a universal and complex relationship between the coverage of two sets of codes linked by *Keto-Amin*o transformation. The existence of a correlation between the length of a sequence and the code coverage is also found.

Contents

1 Introduction **1**

1.1 Biochemical background 1

1.1.1 The translation process 1

1.1.2 The genetic code 2

1.2 Circular codes, symmetries and transformations 5

1.2.1 Comma free codes and circular codes 5

1.2.2 Codon usage and code coverage 10

1.3 Motivation of the study 11

1.3.1 Bootstrap test 14

1.3.2 Overview of further results of interest 15

2 Data and algorithms description **18**

2.1 Data: 24 different organisms 18

2.2 `mathDNA` R package 21

2.3 Removal of beginning and ending parts of the sequences 22

2.4 Codon usage on whole genomes 24

2.5 Code coverage 26

2.5.1 Considering whole genomes 26

2.5.2 Rolling means 27

2.5.3 Considering every sequence 29

2.5.4 By position 33

2.6 Bootstrap test 35

3 Results **36**

3.1 Differences in codon usage between entire and cut sequences 36

3.2 Differences in code coverage between entire and cut sequences 38

3.3 A critical investigation on the rolling means approach 42

3.4 Relationships between best and worst codes coverage on individual sequences 45

3.5 Sequence length effect 52

3.6 Transient effect in the first positions of the sequences 58

3.7 Evidence of a non-random relationship between the coverage of *best* and *worst* codes 61

4 Conclusion **63**

5 Bibliography **65**

6 Appendix **68**

6.1 Appendix A: additional tables 68

6.2 Appendix B: additional figures 75

6.3 Appendix C: code 88

6.3.1 Useful and recurrent objects 88

6.3.2 Cutting the sequences 89

6.3.3 Codon usage on entire genomes 90

6.3.4 Code coverage considering entire genomes 91

6.3.5 Rolling means 92

6.3.6 Codon usage for every sequence 94

6.3.7 Code coverage considering every sequence 97

6.3.8 Code coverage by position 97

6.3.9 Bootstrap test 99

List of Tables **101**

List of Figures **103**

1 Introduction

This introductory chapter is divided into three sections. In the first part, the biological process of translation and the genetic code are introduced together with some of its fundamental properties. In the second part, the theory of circular codes and how it is linked to biochemical transformations through particular symmetries are explained. In the third and last part, the results suggesting that circular codes could be the key to explain some still unknown biological mechanisms will be presented, with a focus on the main results that provided the basis for this work.

1.1 Biochemical background

1.1.1 The translation process

Protein synthesis is a fundamental biological process that takes place within cells, useful for balancing the loss of cellular proteins (through degradation or export) through the production of new ones. This process can be broadly divided into two stages: *transcription* and *translation*.

During transcription, a section of DNA encoding a protein, known as a gene, is converted into a molecule called messenger RNA (*mRNA*), using one strand of the DNA double helix as a template to copy the information it contains. Once the *mRNA* is ready, it exits the nucleus to reach the cytoplasm, where it interacts with the ribosome, which which acts as a protein assembler in the process.

During translation, the *mRNA* is read by ribosomes, which use the nucleotide sequence of the *mRNA* to determine the amino acid sequence. Once the *mRNA* binds to the ribosome, another RNA molecule, known as transfer RNA (*tRNA*), approaches it. This adapter molecule is loaded with an amino acid and three nucleotides that are complementary to those in the sequence of the *mRNA* molecule. Once the entire mRNA sequence is occupied by tRNA molecules, the corresponding amino acids are linked together and assembled into a protein. The ribosome attaches to the mRNA at the start codon (ATG), where it begins to translate the molecule. The nucleotide sequence of the mRNA is read in triplets: three adjacent nucleotides in the mRNA molecule correspond to a single *codon*. The ribosome attaches to the *mRNA* at the start codon (ATG), where it

begins to translate the molecule. The nucleotide sequence of the *mRNA* is read in triplets: three adjacent nucleotides in the mRNA molecule correspond to a single *codon*. The *tRNA* then binds one *anticodon* (the sequence of three nucleotides complementary to the codon on the *mRNA* that corresponds to an amino acid) after another to assemble the protein chain.

This process is illustrated in Figure 1[1].

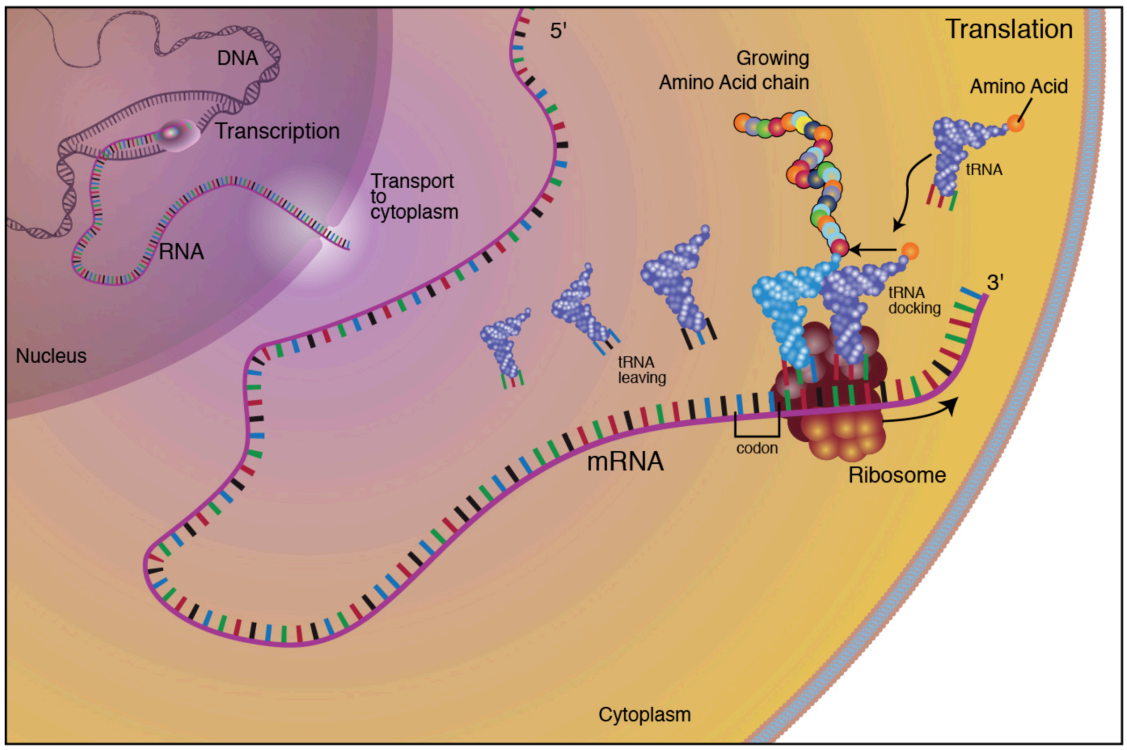


Figure 1: The translation process (from NIH, National Human Genome Research Institute).

1.1.2 The genetic code

In 1953 Watson and Crick revealed the structure of the deoxyribonucleic acid molecule, showing that it is composed of two very long chains coiled into a double helix (Watson & Crick 1953). Phosphate and sugar groups alternate regularly on each chain. Each sugar is linked to one of the four possible nitrogenous bases, namely Adenine (A), Cytosine (C),Thymine (T) and Guanine (G). The two DNA strands are linked together by hydrogen bonds between the nitrogenous bases. The bases are joined in pairs, and only two combinations are allowed: Adenine binds with Thymine

[1]https://www.genome.gov/genetics-glossary/Translation

and Guanine with Cytosine. This property, called *complementarity*, allows the sequence to be replicated from a single strand and is the basis of the replication mechanism.

After this discovery, efforts were made for many years to understand the rules linking the world of nucleotides with that of amino acids and proteins. The physicist Gamow was the first to postulate that groups of 3 bases are used to encode the 20 standard amino acids used by living cells to build proteins, which would allow a maximum of $4^3 = 64$ amino acids (Gamow 1954). Later, in 1961, an experiment was carried out which showed that a synthetic RNA made up of *Uracyl* bases (the RNA equivalent of Thymine) coded for a protein composed entirely of the amino acid *phenylalanine* (Nirenberg & Matthaei 1961). They thereby deduced that the UUU codon (TTT in DNA) encoded for the amino acid *phenylalanine*. This was a turning point in biochemical research, which stimulated researchers to discover the translation table of the genetic code. In fact, it was later discovered with similar methodologies that the codon AAA specified the amino acid *lysine*, and the codon CCC specified the amino acid *proline*. In 1965, therefore, the code was completely deciphered and a table like the one shown in Figure 2[2] was identified.

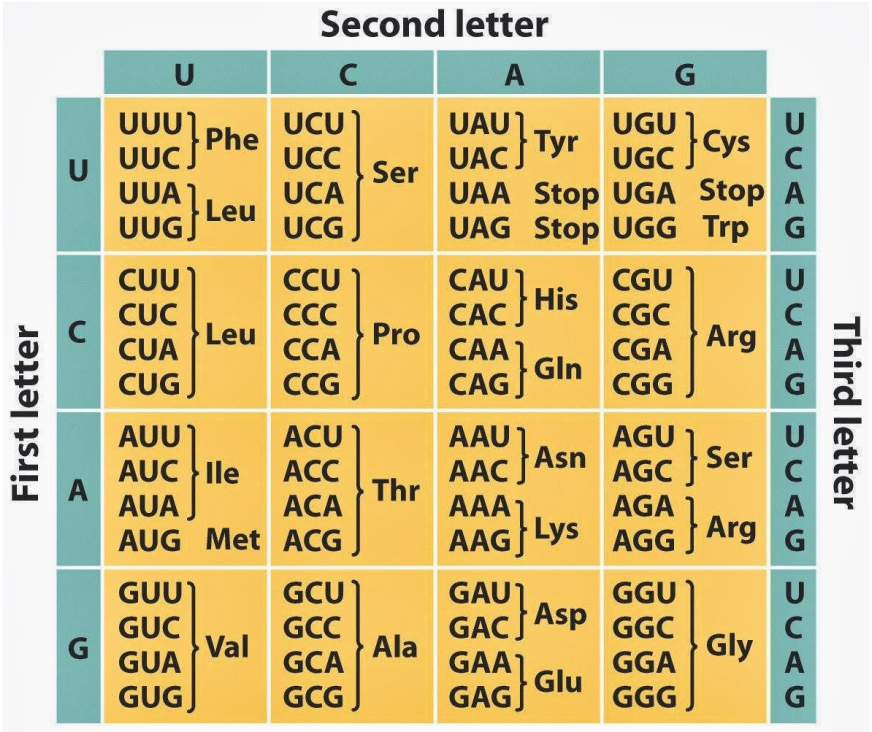


Figure 2: Translation table of the genetic code.

[2] http://biology-pictures.blogspot.com/2013/10/table-of-genetic-code.html

We can, therefore, describe some basic properties of the genetic code:

- it is a *triplet code*: each amino acid is encoded by a triplet of nucleotides (codons);
- it is *degenerate*: each amino acid can be encoded by more than one codon;
- it is *not-overlapping*: each nucleotide in the sequences is part of one and only one codon;
- it is *universal*: it is the same in almost all organisms, from bacteria to viruses to humans. With a few exceptions, such as mitochondrial DNA, it is the universal language of life.

Further important concepts related to the genetic code that are worth mentioning are:

- *start* and *stop codons*, i.e. those particular codons that signal the start and end of the sequence to be encoded during the translation process. The most common start codon is AUG (ATG), which is read as *methionine* or, in bacteria, as *formylmethionine*. The stop codons, on the other hand, are UAG, UGA and UAA and they mark the end of the translation because there are no complementary anticodons to these stop signals, so they allow a *release factor* (which actually releases the protein) to attach to the ribosome.
- the *codon bias*, which is the phenomenon where *synonymous* codons (i.e. those coding for the same amino acid) are not used uniformly, but there is a preference in the use of certain codons over others. This particular phenomenon, which is due to the degeneracy of the genetic code, has been and is still being studied in order to identify a theoretical context that regulates the translation mechanism.

Finally, it is useful to present the notion of *reading frame*, i.e. the way of dividing nucleotide sequences into a group of consecutive, non-overlapping codons. Each sequence, in fact, can be read in three different ways depending on whether you choose to start at the first, second or third position. Consider, for example, the sequence $\{AAATGAACG\}$. If read from the first position, it contains the codons AAA, TGA, and ACG; if read from the second position, it contains the codons AAT and GAA; if read from the third position, it contains the codons ATG and AAC. It is therefore crucial that the translation is done considering the correct reading frame, as errors (called *frame shifts*) can lead to the creation of a completely wrong protein.

1.2 Circular codes, symmetries and transformations

1.2.1 Comma free codes and circular codes

To ensure that protein synthesis is efficient and error-free, two conditions are necessary: that the points where translation begins and ends are correctly recognised and that the ribosome is synchronised in the correct reading frame. The ability to avoid reading errors due to frame shift is called *reading frame maintenance* and is essential because an error in frame synchronisation could lead to the creation of a completely incorrect protein.

While the mechanism for recognising the start and end points of the translation is clear, the dynamics of *reading frame maintenance* are still quite unknown. The first answer was given in Crick et al. (1957) and was based on **comma free codes**. A comma-free code is a special set of codons that allows the correct reading frame to be retrieved at any point in the sequence, provided it is composed of codons that are all part of a comma-free code. Figure 3 shows an example (from Giannerini et al. (2021)), which which clarifies the comprehension.

Example 1. The comma free code X has two codons

$$X = \{\text{CTG}, \text{AAT}\}$$

1. Build a sequence with the codons of X (in green), for instance

   AAT CTG AAT AAT

2. Read it in the 3 possible frames:

   frame 0: AAT CTG AAT AAT
   frame 1: ~~A~~ ATC TGA ATA ~~AT~~
   frame 2: ~~AA~~ TCT GAA TAA ~~T~~

3. There is only one frame (frame 0) where all the codons belong to X: the *correct reading frame*. **None** of the codons (in red) read in frames +1 and +2 belong to X.

Figure 3: Comma free codes - example (reproduced from Giannerini et al., 2021).

Thus, in a sequence composed of codons that are part of a comma free code, a shift in the reading frame immediately leads to a codon that is not part of the code. Despite this desirable property, it has been proven that comma free codes are not adequate to explain the mechanism of reading frame maintenance (Nirenberg & Matthaei 1961). This is due to the fact that, for theoretical

reasons, some codons could not be part of any comma free code, but since all 64 codons are used in protein synthesis, none should be disregarded.

Forty years after Crick's theory of comma free codes, Arquès & Michel (1996) found a less stringent version of comma free codes that allows the correct reading frame to be retrieved: **circular codes**. Again, they can be explained through a simple example in Figure 4, from Giannerini et al. (2021).

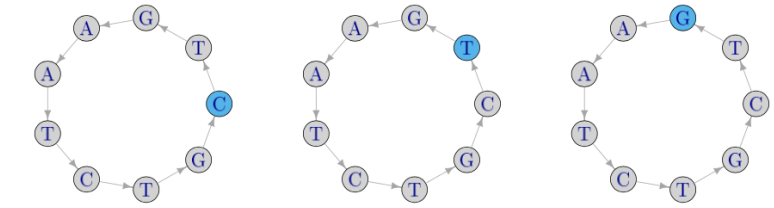
Example 2. Assume that the circular code X has 3 codons

$$X = \{\text{CTG}, \text{AAT}, \text{TGA}\}.$$

1. Form an arbitrary sequence with the codons of X, for instance:

   CTG AAT CTG

2. Put it in a circle and read it in the 3 possible frames (the starting nucleotide is coloured in blue):



3. There is only one frame (frame 0) where all the codons belong to X: the *correct reading frame*, even if some of the codons read in frames +1 and +2 can belong to X.

Figure 4: Circular codes - example (reproduced from Giannerini et al., 2021).

Therefore, the difference between comma free and circular codes is quite clear: in the former a frame shift immediately leads to a codon that is not part of the code, while in the latter it is possible to find *valid* codons even when the sequence is read out of frame.

The codes presented in Arquès & Michel (1996) satisfy three main properties:

- they are *maximal*: they are made up of the maximum number of codons they can contain (i.e. 20, by construction);
- they are *self complementary*: if a codon belongs to a code, then also its reverse complement belongs to the code;

- they are C^3: the circular permutations (Figure 5) of the codons of a circular code also form a maximal circular code.

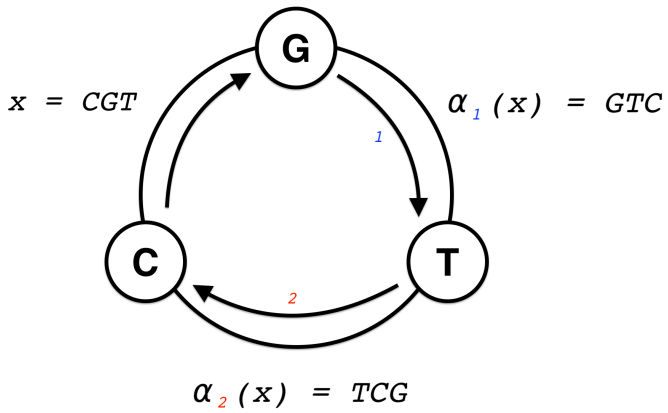


Figure 5: Circular permutation - example.

There are exactly **216** codes that satisfy these three fundamental properties, provided in Michel et al. (2008). It has been shown, in Fimmel et al. (2015), that these 216 codes have special symmetries linked to nucleotide transformations, i.e. those rules that map the set of 4 nucleotides onto one of its 24 possible permutations. Among these, there are 8 special transformations that are related to the *dihedral symmetry group*, i.e. they represent the 8 symmetries of a square (in Figure 6). The first four transformations in Figure 6 constitute a particular symmetry group (called *Klein V group*) that contains the identity and three chemical transformations of nucleotides (Gonzalez et al. 2008).

1.	(A)(T)(C)(G)	: A ↦ A; T ↦ T; C ↦ C; G ↦ G	Identity	(I)
2.	(AT)(CG)	: A ↦ T; T ↦ A; C ↦ G; G ↦ C	Strong/Weak	(SW)
3.	(AG)(CT)	: A ↦ G; G ↦ A; C ↦ T; T ↦ C	Purine/Pyrimidine	(YR)
4.	(AC)(GT)	: A ↦ C; C ↦ A; G ↦ T; T ↦ G	Keto/Amino	(KM)
5.	(A)(T)(CG)	: A ↦ A; T ↦ T; C ↦ G; G ↦ C		
6.	(AT)(C)(G)	: A ↦ T; T ↦ A; C ↦ C; G ↦ G		
7.	(ACTG)	: A ↦ C; C ↦ T; T ↦ G; G ↦ A		
8.	(AGTC)	: A ↦ G; G ↦ T; T ↦ C; C ↦ A		

Figure 6: Transformations of the nucleotides forming the dihedral group (reproduced from Giannerini et al., 2021).

It has been shown that through these 8 transformations it is possible to split the 216 circular codes into 27 equivalence classes, each containing 8 circular codes linked by the transformations in Figure 6 (Fimmel et al. 2015).

Table 1 shows the 216 circular codes grouped in the 27 equivalence classes (each number corresponds to the index of one of the 216 codes). In Table 2, on the other hand, it is possible to observe the codons that are part of eight circular codes in one of the 27 equivalence classes (the first row in Table 1).

Among the 27 equivalence classes, there are 16 for which the *identity* (*I*, first column in Table 1) and *Keto-Amino* transformation (*KM*, last column in Table 1) codes have no codons in common. These 16 classes are highlighted in bold in Table 1. We can see, in fact, that the codons in the first and last columns of Table 2 (that refers to the first equivalence class) are all distinct.

Lastly, it is useful to note that during the analysis, reference will often be made to a so-called **remainder** code. By remainder code, it is meant that group of codons composed by the total 64 minus the ones that compose the *best* and the *worst* codes within the single classes of equivalence (we will have, therefore, 27 remainder codes). The size of this group will be, therefore, equal to 24 codons for the 16 equivalence classes in which best code and worst code are disjoint ($64-20-20=24$), while it will be greater in the remaining 11 equivalence classes.

Table 1: The 216 circular codes grouped in 27 equivalence classes according to the 8 transformations of the dihedral symmetry group. The rows in bold refers to the 16 classes for which the codes corresponding to the *identity* (*I*, first column) and to the *Keto-Amino* transformation (*KM*, last column) have no common codons.

I	AU	CG	SW	YR	ACUG	AGUC	KM
173	**176**	**203**	**206**	**183**	**193**	**182**	**192**
23	**33**	**77**	**81**	**13**	**65**	**37**	**87**
98	**10**	**96**	**8**	**52**	**55**	**45**	**53**
25	35	76	85	50	59	47	56
20	34	75	80	17	69	40	89
166	**216**	**164**	**213**	**186**	**189**	**187**	**191**
4	**104**	**6**	**102**	**16**	**61**	**42**	**86**
30	27	84	72	12	64	38	88
117	160	118	157	130	133	131	135
111	159	116	151	119	138	126	145
22	29	71	79	2	100	1	**99**
172	**175**	**202**	**205**	**181**	**196**	**184**	**195**
21	**31**	**74**	**78**	**11**	**68**	**39**	**91**
24	**32**	**73**	**83**	**49**	**60**	**48**	**57**
97	**9**	**95**	**7**	**51**	**58**	**46**	**54**
171	**174**	**201**	**204**	**167**	**200**	**178**	**208**
3	**103**	**5**	**101**	**15**	**62**	**43**	**90**
165	**215**	**163**	**212**	**185**	**190**	**188**	**194**
26	28	70	82	36	92	14	**66**
123	124	141	143	105	150	106	147
115	**158**	**113**	**155**	**129**	**134**	**132**	**136**
161	**214**	**162**	**211**	**168**	**197**	**179**	**207**
122	125	140	142	110	152	108	**149**
41	**94**	**18**	**67**	**19**	**63**	**44**	**93**
107	**156**	**112**	**148**	**120**	**139**	**127**	**146**
198	170	209	180	169	199	177	210
137	121	144	128	114	153	109	154

Table 2: Equivalence class formed by eight circular codes. Each column contains codons in 8 of the 216 circular codons, related to each other by transformations of the dihedral group.

I	*AU*	*CG*	*SW*	*YR*	*ACUG*	*AGUC*	*KM*
173	176	203	206	183	193	182	192
AAC	AAC	AAG	AAG	AAT	ACA	AAT	ACA
GTT	GTT	CTT	CTT	ATT	TGT	ATT	TGT
AAT	ATC	AAT	ATG	ACA	ACT	ACA	ACT
ATT	GAT	ATT	CAT	TGT	AGT	TGT	AGT
ATC	CAC	ATG	CAA	ACC	AGA	ACC	AGA
GAT	GTG	CAT	TTG	GGT	TCT	GGT	TCT
CAC	CAG	CAA	CAC	ACG	CCA	ACG	CCA
GTG	CTG	TTG	GTG	CGT	TGG	CGT	TGG
CAG	CTC	CAC	CAG	ACT	CGA	ACT	CCG
CTG	GAG	GTG	CTG	AGT	TCG	AGT	CGG
CTC	GAA	CAG	CCG	AGA	GCA	AGA	CGA
GAG	TTC	CTG	CGG	TCT	TGC	TCT	TCG
GAA	GAC	CCG	CTA	AGC	GCC	AGC	GCA
TTC	GTC	CGG	TAG	GCT	GGC	GCT	TGC
GAC	GCC	CTA	CTC	AGG	GGA	AGG	GGA
GTC	GGC	TAG	GAG	CCT	TCC	CCT	TCC
GCC	GTA	CTC	GAC	GCC	TAA	CCG	TAA
GGC	TAC	GAG	GTC	GGC	TTA	CGG	TTA
GTA	TAA	GAC	TAA	TCA	TCA	TCA	TCA
TAC	TTA	GTC	TTA	TGA	TGA	TGA	TGA

1.2.2 Codon usage and code coverage

Codon usage and **code coverage** are two quantities that will play a key role in the analysis presented. The term codon usage simply refers to the frequency of occurrence of each of the 64 codons in one (or more) DNA sequence. For example, in the sequence composed by the group of codons $\{ATG, AGC, GTT, ACA, ATG, GTT, ATG, GTT\}$, the codon usage for ATG and GTT would be $3/8 = 0.375$, for ACG and ACA would be $1/8 = 0.125$ and for the remaining 60 codons would be zero.

Instead, by code coverage over one sequence or organisms, we mean the sum of codon usage that are part of a code. This particular quantity can be considered as a measure of the *goodness* of a code, as it measures how much a code is present in a sequence or organism, thus how much

it contributes to its translation (Gonzalez et al. 2009). Obviously, in our case, we will analyse and discuss the coverage of circular codes. Figure 7 shows a simple example of code coverage calculation, extracted from Giannerini et al. (2021), where a rigorous mathematical definition of this quantity is also given.

Example 3. Consider the sequence CAT CTG AAT GGA CTG and the two codes $X_1 = \{\mathrm{CTG}, \mathrm{AAT}\}$, $X_2 = \{\mathrm{GGA}, \mathrm{TGT}\}$. The codon usage of the sequence is

Codons	CAT	CTG	AAT	GGA
Usage	1/5	2/5	1/5	1/5

The coverage of X_1 results $2/5 + 1/5 = 3/5 = 0.60$, and that of X_2 results $1/5 = 0.20$.

Figure 7: Codon usage and code coverage - example (reproduced from Giannerini et al., 2021).

From now on, in order to simplify the analytical formulae, the following notation will be used: cu_i indicates the codon usage of the generic codon i; C_j indicates the code coverage of the generic code j.

1.3 Motivation of the study

In Giannerini et al. (2021), a study was conducted on the entire Codon Usage Database, i.e. on all organisms for which codon usage values are available, which discovered several universal results on circular codes coverage. It has been observed that, although the values for coverage are obviously more or less variable depending on the taxonomy of the organisms under consideration, there are universal recurring properties related to the symmetries of the circular codes. In particular, considering the 8 codes present in each of the 27 equivalence classes sorted according to code coverage in the different organisms, it is possible to observe a recurrent order (which follows the order of the columns in Table 1). This property applies to all 27 equivalence classes. Even more surprisingly, it was also proved that the code with the lowest coverage within each class (the *worst* one) corresponds to the chemical *Keto-Amino* transformation of the code with the highest coverage (the *best*).

Figure 8 (extracted from Giannerini et al. (2021)) shows some of the results obtained by considering

the codes in Table 2 to get an idea of the universal properties mentioned. The Figure presents the coverage (top panel), absolute ranks (middle panel) and relative ranks (bottom panel) for the equivalence class of the 8 circular codes. The universality of the results becomes clear when ranks within classes are considered. For example, although the coverage of the code *173* (46.4%) is not the highest among the 216 codes (it is the second), it is the highest within its class. This universal behaviour applies to the whole set of 216 codes divided into 27 equivalence classes.

coverage	X_{173}	X_{176}	X_{203}	X_{206}	X_{183}	X_{182}	X_{193}	X_{192}
bacteria	46.4	43.9	36.0	33.6	26.8	22.8	22.1	18.1
animals	42.0	38.8	35.9	32.8	28.6	26.2	25.8	23.4
viral	43.2	40.3	35.9	33.0	28.4	26.1	24.7	22.4
plants	39.7	36.7	34.8	31.7	29.3	27.5	25.3	23.5
absolute rank	X_{173}	X_{176}	X_{203}	X_{206}	X_{183}	X_{182}	X_{193}	X_{192}
bacteria	2	11	58	81	155	189	195	212
animals	2	19	43	84	148	180	187	208
viral	2	18	53	84	148	176	190	209
plants	16	35	55	98	140	165	190	208
relative rank	X_{173}	X_{176}	X_{203}	X_{206}	X_{183}	X_{182}	X_{193}	X_{192}
bacteria	1	2	3	4	5	6	7	8
animals	1	2	3	4	5	6	7	8
viral	1	2	3	4	5	6	7	8
plants	1	2	3	4	5	6	7	8

Figure 8: Circular code coverage - universal properties (reproduced from Giannerini et al., 2021).

These results, therefore, suggest the existence of a universal order structure and that this can be linked to the theory of circular codes, with particular attention to the *Keto-Amino* chemical transformation. The aim of this study, therefore, will be to understand more about this scheme, focusing in particular on the dualism between the identity code (which will henceforth be referred to as the *best* code) and the transformed *KM* code (which will be referred to as the *worst* code). For this reason, the first and last columns of Tables 1 and 2, which refer to the best and worst codes, are coloured in blue and red. Furthermore, the order of the classes in Table 1 by row is not random. The equivalence classes, in fact, are ordered according to how strong the properties discussed within the eight codes in each class are.

Another interesting result pointed out in Giannerini et al. (2021) is related to the beginning and end of the sequences. It was observed, in fact, that a particular behaviour for the coverage of the

best (*173*) and worst (*192*) codes in the first and last about 30 codons of the DNA sequences. In particular, it was observed that in these areas the coverage for the best code tends to be lower than the average over the whole sequence. For the worst code, on the other hand, an opposite pattern is observed. These considerations are also common to all organisms under investigation. In Figure 9 (Giannerini et al. 2021) this behaviour can be observed on the results for code coverage calculated by rolling windows on *E.coli*.

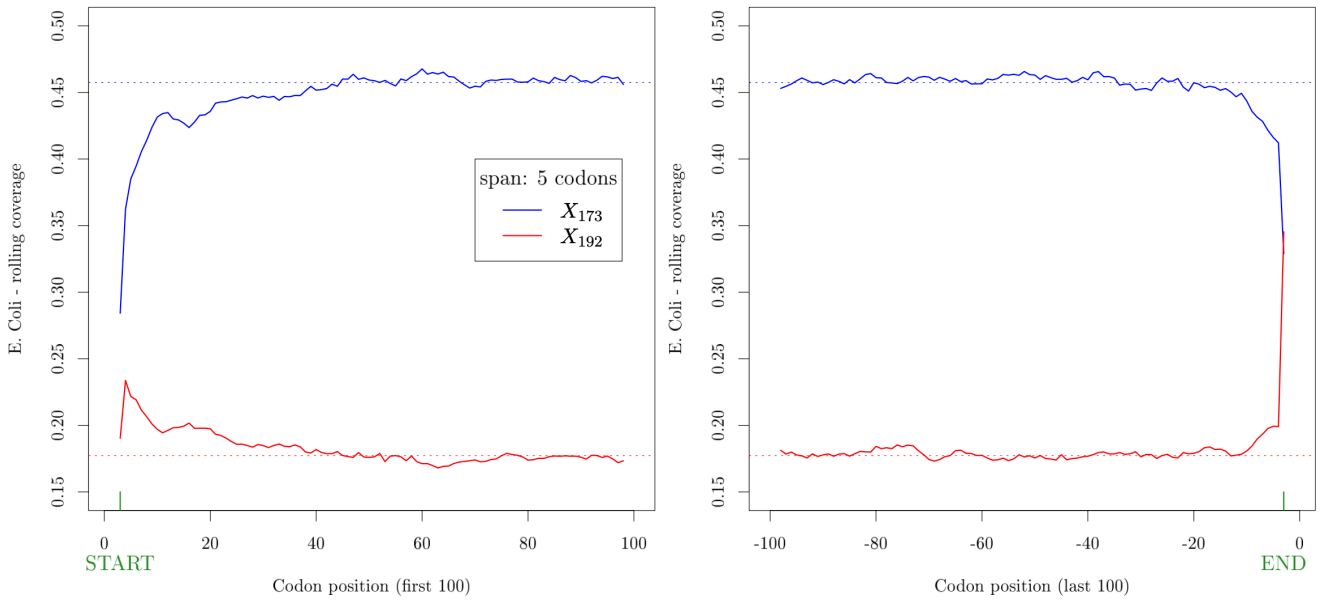


Figure 9: Coverage of the best code (173, in blue) and of the worst code (192, in red) at the beginning and at the end of the sequences in *E.coli* (reproduced from Giannerini et al., 2021).

A further aim of this analysis, therefore, will also be to investigate this effect at the beginning and end of the sequence by considering different approaches to calculating code coverage and the different codes in the 27 equivalence classes.

With the introduction of these new concepts, it is useful to update the previous notation:

$cu_i^{ent} \implies$ codon usage of the generic codon i, considering the entire sequences

$cu_i^{cut} \implies$ codon usage of the generic codon i, considering the sequences without initial and final parts

$C_b^{ent}, C_w^{ent}, C_r^{ent} \implies$ code coverages of the best, worst and remainder codes respectively, considering the entire sequences

$C_b^{cut}, C_w^{cut}, C_r^{cut} \implies$ code coverages of the best, worst and remainder codes respectively, considering the sequences without initial and final parts

1.3.1 Bootstrap test

It was observed that the code with the highest coverage is (almost) unique and that the code with the worst result corresponds to the *KM* transformation applied to the *best* code. However, it is reasonable to expect that the more recurrent a set of codons is, the less recurrent are the codons that do not belong to that set. It makes sense, then, to investigate whether the particular relationship linking *best* and *worst* codes is simply due to chance.

To answer this question, a statistical test was defined and applied in Giannerini et al. (2021). This bootstrap test led to proof that the inverse relationship linking the code coverage of the code pairs *173-192* and *23-87* (in the first and second rows of Table 1) is not due to chance with a p-value smaller or to 0.0001.

The aim of this test is to understand whether the difference between the code coverage of the *best* and *worst* codes is compatible with that which would be produced by a random choice of codons. Taking, for example, the *best* and *worst* sets in general, i.e. the pair *173-192*, it was tested whether coverage of code *192* is significantly lower than coverage of a random group of 20 codons taken from those not belonging to code *173* (which will be 44, i.e. 64 total minus the 20 from *173*). The null hypothesis, therefore, is that the coverage of the worst group is compatible with the coverage obtained by considering a random group of 20 codons. The alternative hypothesis, instead, is that the results are not compatible and that, therefore, the relation between the *best* and the *worst* codes is not due to chance.

$$\begin{cases} H_0 : C_w \text{ compatible with } C_{RAN} & \Longrightarrow \text{ relationship due to chance} \\ H_1 : C_w \text{ not compatible with } C_{RAN} & \Longrightarrow \text{ relationship not due to chance} \end{cases}$$

where C_w is the coverage of the *worst* code and C_{RAN} is the random variable representing the coverage of a random set of 20 codons taken from the subset of 44 codons complementary to the ones in the *best* code.

To perform the test, the results of the code coverage calculated on the whole genome are considered and 10,000 sets of 20 random codons are generated, on which the coverage on the genome under consideration is calculated. The set of resampled codons was made homogeneous with respect to *CG content* by imposing that their *CG content* be equal to that of the *worst* code considered.

In addition, resampling of the random sets of codons was carried out under two different assumptions: extracting the 20 codons according to a uniform distribution over the 44 codons complementary to the *best* code or respecting the distribution of these codons in the genome under consideration. The first hypothesis therefore assumes that all 216 codons have the same probability of occurrence and exist independently of the codon usage of each genome. The second, on the other hand, assumes that the occurrence of circular codes is linked to the codon usage of the genome under consideration.

The test, performed on 291 genomes, rejected with a p-value < 0.0001 the null hypothesis that the negative relationship linking codes *173-192* and *23-87* is due to random fluctuations, considering both hypotheses that the occurrence of the codes is uniform or linked to the code coverage of the genome.

During this study, therefore, this same test will be applied to verify that the results on the relationships between the coverages obtained are not due to chance.

1.3.2 Overview of further results of interest

Since the discovery of a common circular code in the genomes of eukaryotes and prokaryotes in 1996 (Arquès & Michel 1996), the theory of circular codes has aroused great interest and underwent a rapid development. Several academics from different fields, from statistics to mathematics

to bionformatics, have in fact studied this theory in search of connections with the process of translation. Initially, the code identified by Arquès and Michel was only one, the so-called *set X*. This *set X* contains the following 20 trinucleotides:

X = {AAC, AAT, ACC, ATC, ATT, CAG, CTC, CTG, GAA, GAC, GAG, GAT, GCC, GGC, GGT, GTA, GTC, GTT, TAC, TTC}

This set X is associated with two other sets X_1 and X_2 of 20 nucleotides. These sets result from frame shifting on the set X of one and two positions respectively. Furthermore, it has been shown that these three codons are linked by circular permutations.

In 2011, the coverage of the entire class of 216 codes with respect to a large set of coding sequences was studied using a statistical approach (Gonzalez et al. 2011). The results of this study suggested that, on average, the code proposed by Arquès and Michel had the best coverage capacity and identified the existence of a sort of optimisation mechanism that relates the function of circular codes to the synchronisation of reading frames. In 2015, again using a statistical approach, the presence of the circular code X in prokaryotes and eukaryotes was studied in greater depth, and it was also identified in the genes of bacteria, plasmids and viruses (Michel 2015). The study also identified several variants of code X associated with different types of organisms.

Recently, the *maximality* property of the three circular codes X, X_1 and X_2 (all made up of 20 codons) has been statistically verified (Michel 2020). In another study, a necessary condition for the *self-complementarity* of an arbitrary code is demonstrated in terms of graph theory (Fimmel et al. 2018). In this paper it is also shown that circular codes allow the (correct) reading frame to be recovered in any arbitrary trinucleotide sequence after a maximum of 15 nucleotides, i.e. after 5 consecutive codons.

As regards the identification of the symmetries linking the circular codes, following some primordial hypothesis (Koch & Lehman 1997, Lacan & Michel 2001), in 2015 the particular transformations that allow the identification of the 8 equivalence classes described above were identified (Fimmel et al. 2015).

In addition, some research suggests that there is a relationship between circular codes and the origin of the genetic code, which is still an enigmatic topic. Recently, a study concerning the mathematical properties of the code X and its presence in the main actors involved in translation

suggested that it is an ancestor of the standard genetic code that was used to encode amino acids and simultaneously to identify and maintain the reading frame (Dila et al. 2019).

Moreover, in recent years efforts have been made to extend and generalise the theory of circular codes. In Fayazi et al. (2021), in fact, codes have been studied not only in the triletter case over the genetic alphabet with four letters, but generalizing to l-letter codes over larger alphabets. This study was motivated by some previous findings that suggested that nature may encode not only one set of information in DNA but 8 or even 24 sets at the same time (Demongeot & Seligmann 2020, Michel & Seligmann 2014, Seligmann 2016).

Finally, in Fimmel et al. (2020) the definition of circular codes was extended and so-called *k-circular* codes were introduced. A code C is said to be *k-circular* if any concatenation of at most k words from C, when read on a circle, admits exactly one partition into words from C. When a code is *k-circular* for each integer k, then it is circular. The results of this study suggest that this type of code may represent an important evolutionary step between the circular codes and the genetic code.

2 Data and algorithms description

This section discusses the data under analysis and describes as clearly and discursively as possible the operations that were carried out to obtain the results presented in the next section.

More detailed information on the processes can be deduced by directly reading the R code used to obtain the results, which can be found in **Appendix C**. In particular, in this chapter, objects created during the analysis will be described and the name of the corresponding R object in the code will be shown (in *italics*) to facilitate a complete understanding. In addition, together with the explanation of each process, a small (numeric) preview of the results obtained will be provided to give an example of how the objects created appear, since in the chapter describing the results priority will be given to graphic representations.

2.1 Data: 24 different organisms

The data evaluated in this research are 24 genomes of 24 different organisms, listed in the first column of Table 3.

Each genome is stored in an `RData` file and, once loaded into R, it is a list of sequences. Each sequence is a vector of single characters made up of different combination of the 4 nitrogenous basis (A, T, G, C). Each genome is composed by a different number of sequences (that are displayed in the third column of Table 3) and each sequence can consist of a different number of bases and encodes for a protein. Each sequence is *coding*, i.e. without introns (the non-coding regions of the genes), and *complete*, i.e. including the start and the stop codons.

The data come from *GenBank* (NCBI Resource Coordinators 2016), the NIH (*National Institutes of Health*) genetic sequence database that contains a collection of all publicly available DNA sequences, and were extracted using the R package `seqinr` (Charif & Lobry 2007).

Table 3 offers an overview of lengths and sizes of the genomes that will be analyzed. In the second and third columns, in fact, are respectively displayed the total number of basis composing all the sequences in the genomes and the number of sequences available for each genome.

Often during the analysis the single characters in the sequences (nitrogenous basis) are merged in groups of three sequential units, in order to obtain vectors of codons (three nitrogenous basis). Furthermore, since we have genomes of variable sizes, data have been frequently split into groups of genomes of similar sizes (in terms of number of basis) in order to make computations optimized and less time-consuming.

For every approach discussed in this section, will be also provided a small part of the created objects in order to make explanations clearer, while in the Results part of the dissertation the obtained results will be displayed and evaluated through graphical and more comprehensive tools. However, since we are dealing with huge datasets and, consequently, large results, information on six recurrent genomes that have been chosen as representatives of all the others will be presented and discussed. This choice is linked to the need for representativeness of the different sizes of genomes. Furthermore, *model* genomes, i.e. organisms for which similar types of analysis have been performed, were preferred over others in order to allow reproducibility and comparison with other studies. These six genomes of interest are colored in red in Table 3.

Table 3: Size of genomes considered in the study, *model* genomes in red.

Genome	Total number of bases	Total number of sequences
AeropyrumPernix	686,592	713
Thermoplasma.acidophilum	1,137,609	1,150
P.Horikoshii	1,377,468	1,583
Pyrococcus	1,388,490	1,441
Staphylococcus.aureus	1,946,109	1,977
Helicobacter.pylori	2,545,650	2,392
Methanosarcina	3,119,499	2,963
Archaeoglobus	3,541,029	3,757
Escherichia.coli	4,040,190	3,983
Streptomyces.coelicolorA3	5,485,755	5,202
M.Xanthus	6,130,989	5,037
Caenorhabditis.elegans	6,331,206	3,347
Sulfolobus.solfataricus	8,907,675	9,674
Schizosaccharomyces.Pombe	11,363,931	7,711
Plasmodiumfalciparum3D7	12,364,287	5,259
Leishmania.major	15,855,987	8,239
Drosophila.melanogaster	26,836,692	12,606
DanioRerio	31,672,806	24,118
ZeaMays	75,506,157	70,650
OryzaSativa	77,859,006	65,554
Bacillus.subtilis	104,637,978	104,992
MusMusculus	124,647,810	92,857
Homo.Sapiens	175,433,904	140,450
Arabidopsis.Thaliana	198,486,924	151,245

2.2 mathDNA R package

The analyses are conducted in the R environment and part of this work is dedicated to assist the development of the existing R package called `mathDNA` (Giennerini & Dalena 2021), that implements some of the functions that will be needed for the analysis.

Figure 10: `mathDNA` package logo

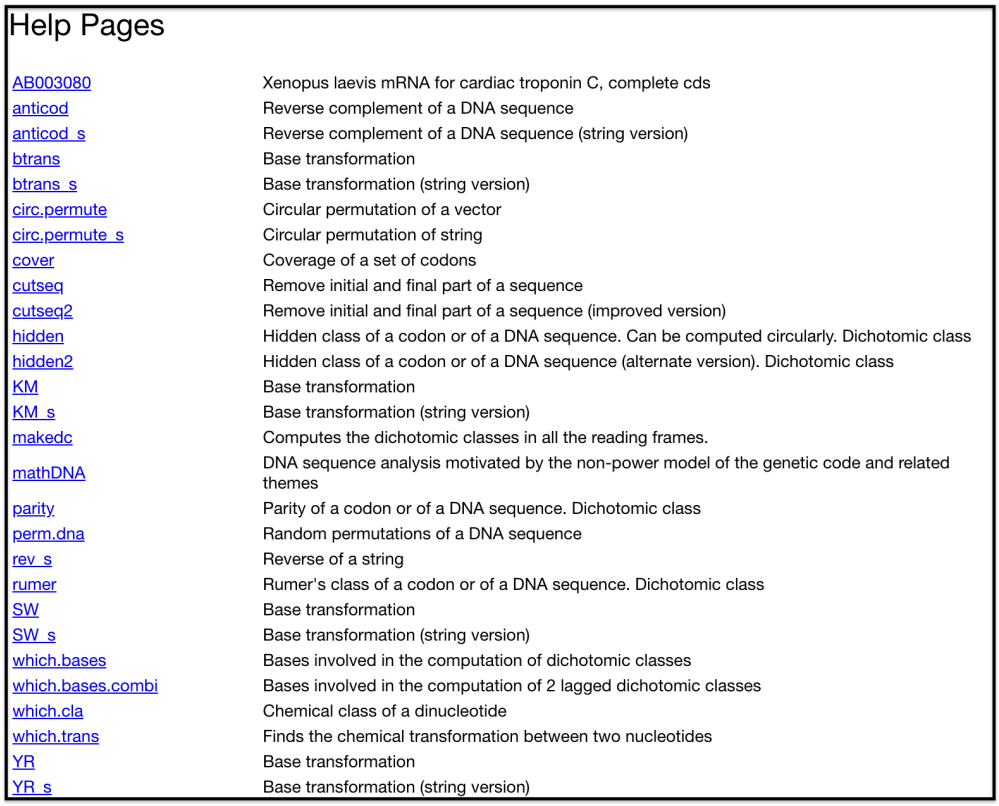
The package contains functions for managing strings in general, as well as DNA sequences. It implements different transformations on the sequences (*base transformations, circular permutation, reverse complement, random permutation*) and it can compute the coverage of different groups of codons in genomes (*codon/code usage*). In addition, the package implements functions for computing the so called *dichotomic classes*, binary variables motivated by the non-power model of the genetic code (Giannerini et al. 2012, Gonzalez et al. 2009).

The developing version of the R package is available on GitHub[3] and it is stable and ready to be used. In fact, the package already passes the `R CMD check` with zero notes, zero warnings and zero errors. Theoretically, therefore, it is ready to be released on CRAN, but there are still plans to add more features. One of the most important is related to the recognition of comma-free and circular codes, that are functions already present in another R package called `GCATR`[4] (Starman 2018), developed in `C++` and adapted into the `R` environment thanks to the `Rcpp` R package (Eddelbuettel & François 2011).

In Figure 11 there is the list of the help pages provided in R that displays the function present in the `mathDNA` R package with brief descriptions.

[3] At this link: https://github.com/PaoloDalena/mathDNA . At the time of writing the repository is private, but things may change. Email me if you have any problems.

[4] Developement version available at: https://github.com/StarmanMartin/GCATR



Help Pages

AB003080	Xenopus laevis mRNA for cardiac troponin C, complete cds
anticod	Reverse complement of a DNA sequence
anticod_s	Reverse complement of a DNA sequence (string version)
btrans	Base transformation
btrans_s	Base transformation (string version)
circ.permute	Circular permutation of a vector
circ.permute_s	Circular permutation of string
cover	Coverage of a set of codons
cutseq	Remove initial and final part of a sequence
cutseq2	Remove initial and final part of a sequence (improved version)
hidden	Hidden class of a codon or of a DNA sequence. Can be computed circularly. Dichotomic class
hidden2	Hidden class of a codon or of a DNA sequence (alternate version). Dichotomic class
KM	Base transformation
KM_s	Base transformation (string version)
makedc	Computes the dichotomic classes in all the reading frames.
mathDNA	DNA sequence analysis motivated by the non-power model of the genetic code and related themes
parity	Parity of a codon or of a DNA sequence. Dichotomic class
perm.dna	Random permutations of a DNA sequence
rev_s	Reverse of a string
rumer	Rumer's class of a codon or of a DNA sequence. Dichotomic class
SW	Base transformation
SW_s	Base transformation (string version)
which.bases	Bases involved in the computation of dichotomic classes
which.bases.combi	Bases involved in the computation of 2 lagged dichotomic classes
which.cla	Chemical class of a dinucleotide
which.trans	Finds the chemical transformation between two nucleotides
YR	Base transformation
YR_s	Base transformation (string version)

Figure 11: Help pages for mathDNA

2.3 Removal of beginning and ending parts of the sequences

As already mentioned in the chapter on circular codes, the analyses will also focus on the different behaviours found in the initial and final parts of the sequences. Figure 13, which provides a summary of the results obtained by considering the sequences of the *model* genomes, provides further insight into the behaviour of interest. In particular, in part (a) are shown the plots of the results for the code coverage in the first 1000 codons, where the three lines represent the coverage for the best (*173*, in blue), the worst (*192*, in red) and the *remainder* (in green) codes. Furthermore, in part (b) it is possible to observe the results on the best and worst codes but for the first 50 codons only, with the comparison with the population mean (dashed lines of the same colours). Without focusing on how the results were obtained (the specific methods will be described in the next paragraphs), it is clear that at the beginning there is an unusual behaviour compared to the rest of the sequences and that, thus, it makes sense to consider the sequences by removing the initial and final parts.

The procedure for cutting sequences is quite simple and relies on the `cutseq2` function from `mathDNA`. As it is explained in the R help page for this sequences (available in Figure 12), it subsets the character vector according to the parameters `head` and `tail`, that allows the user to specify how many character must be removed. This function is the optimized version of the function `cutseq` (also present in `mathDNA`, as we can see in Figure 11), that works both with vectors of single characters or unique strings of characters and returns an output of the same format of the input (but it is more time-consuming).

cutseq2 {mathDNA} R Documentation

Remove initial and final part of a sequence (improved version)

Description

This function removes the initial and/or final part of a sequence, according to two splitting parameters `head` and `tail`.
For a more general (but slower) version of this function, please refer to `cutseq`.

Usage

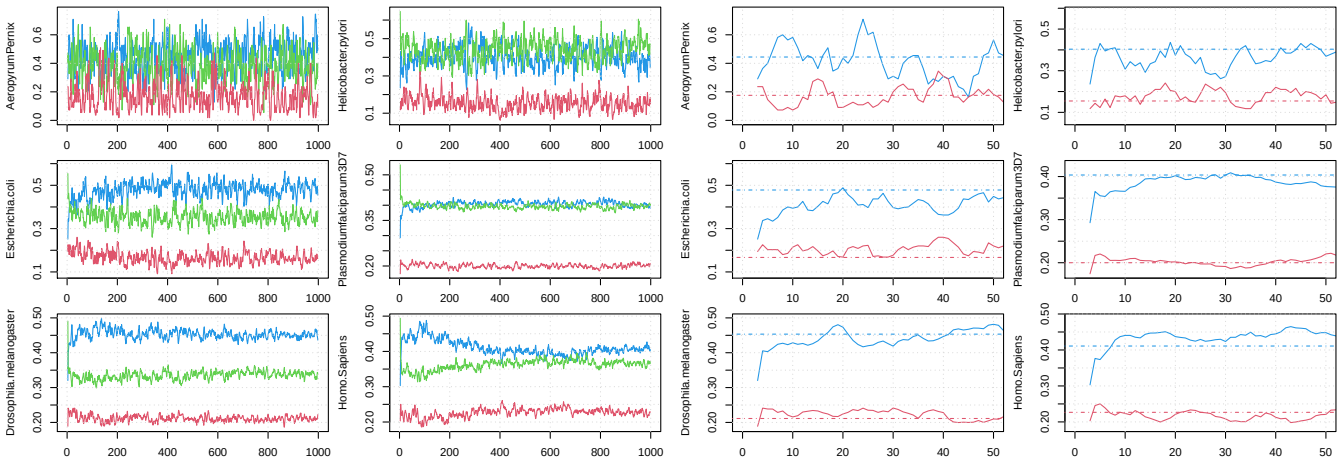
```
cutseq2(seq, head = 0, tail = 0)
```

Arguments

`seq`	The sequence of interest. A vector of individual characters (e.g. `c("A", "B", "C", "D", "E", "F")`).
`head`	An integer defining how many characters to remove from the initial part of the sequence.
`tail`	An integer defining how many characters to remove from the final part of the sequence.

Figure 12: R help page for `cutseq2` from `mathDNA`

In order to remove the initial and final part of the sequences during computations, we apply recursively `cutseq2` on every sequence in the genomes and, for each iteration, we also check if the new dimensions of the sequences are coherent with the specification provided (so, if $length_{cut}$ + `head` + `tail` = $length_{entire}$). For the analysis described in the following sections, the values for `head` and `tail` have been set respectively equal to 39 (13 codons) and 30 (10 codons).



(a) First 1000 codons (b) First 50 codons, with population mean

Figure 13: Code coverage distribution of best (173, in blue), worst (192, in red) and remainder (in green) codes - *model* genomes.

2.4 Codon usage on whole genomes

As described in the introductory chapter on circular codons, codon usage refers to the frequency of occurrence of each of the 64 codons in a sequence. In this section we will illustrate the procedures for calculating codon usage over the whole genomes, thus considering all the sequences of the genomes as if they were one very long sequence.

The main operations for computing the codon usage on whole genomes are:

- generating a vector with all the 64 codons in the genetic code (the possible permutations of three-letter sequences that can be made from the four nitrogenous bases) $\Longrightarrow \underset{[1\times 64]}{tre.s}$
- unlisting the information present in the genome data, in order to have a unique long vector made up of all the different sequences in each genome list one after another. The individual characters corresponding to the different nitrogenous bases are brought together every third so that the resulting vector consists of a sequence of codons $\Longrightarrow \underset{[1\times codons]}{xx0}$
- in order to compute the single codon usage on the whole genome, the proportions of every codon present in *tre.s* are inspected in $xx0$. The result vector, then, contains the proportions of usage of every possible codon in the whole genome taken into account. The results for the codon usage will be stored in a matrix with 64 rows (one for each codon in the genetic code) and 24 columns (one for each genome under analysis) $\Longrightarrow \underset{[64\times 24]}{cu0}$

An example of the obtained results for *AeropyrumPernix* and *Homo.Sapiens* for the first 20 codons can be found in Table 4. The values in the table are frequencies that sum to 1. For example, therefore, AAG in *Homo.sapiens* represents 3.2% of the entire genome (which corresponds to a value of 0.032 in the table).

Table 4: Preview of a small part of the results: codon usage of only 20 codons - *AeropyrumPernix* and *Homo.Sapiens*.

	AeropyrumPernix	Homo.Sapiens
AAA	0.009	0.015
AAC	0.018	0.020
AAG	0.037	0.032
AAT	0.004	0.015
ACA	0.010	0.011
ACC	0.014	0.015
ACG	0.011	0.012
ACT	0.009	0.010
AGA	0.010	0.010
AGC	0.025	0.017
AGG	0.050	0.016
AGT	0.005	0.008
ATA	0.041	0.009
ATC	0.010	0.021
ATG	0.024	0.024
ATT	0.009	0.014
CAA	0.002	0.012
CAC	0.011	0.014
CAG	0.017	0.021
CAT	0.004	0.011
[...]		

2.5 Code coverage

As previously explained, the coverage of a code is the cumulative codon usage of the set of codons belonging to that code. In the following paragraphs will be discussed the different approaches evaluated for computing the code coverage of the different circular code groups on the DNA sequences.

2.5.1 Considering whole genomes

The objects needed for this analysis are:

- the matrix containing all the 216 groups of circular codes. This matrix has 216 columns, one for each circular code, and 20 rows, one for each codon in every group. $\implies \underset{[20\times216]}{ccod}$
- the vector with all the 64 codons discussed in the previous section $\implies \underset{[1\times64]}{tre.s}$
- the results of the code usage on whole genomes from the previous section $\implies \underset{[64\times24]}{cu0}$

In order to compute the different coverage of the different circular codes, we apply on the results of the codon usage for the whole genomes the function `cover` of the `mathDNA` package.

As we can see from the R help page of this function (Figure 6), we just have to provide to this function:

- the set of codons of which we are interested to compute the coverage in, that in our case will be the different columns in $\underset{[20\times216]}{ccod}$;
- the vector containing the set of codons used for computing the codon usage of the sequence, that in our case will be all the 64 codons, that are stored in $\underset{[1\times64]}{tre.s}$;

Coverage of a set of codons

Description

Given a set of codons with its codon usage, this function computes the coverage of the (sub)set x.

Usage

```
cover(x, codons, usage)
```

Arguments

`x`	A vector containing the (sub)set of codons of interest that will be used to compute the coverage.
`codons`	A vector containing the set of codons used for computing the codon usage of the sequence.
`usage`	A matrix containing the codon usage for each codon (in the rows) and for each DNA sequence (in the columns).

Figure 14: R help page for `cutseq2` from `mathDNA`

- the matrix with the results of the codon

  usage $\underset{[64\times 24]}{cu0}$

The result matrix (an object called $\underset{[24\times 216]}{RES0}$), then, will be a matrix with 24 rows (one for each genome) and 216 columns (one for each circular code). Table 5 offers an example of the results, for two genomes and the code groups in the first row of Table 1 only.

Table 5: Preview of a small part of the results: code coverage of sets in the first equivalence class only - *AeropyrumPernix* and *Homo.Sapiens*.

	173	176	203	206	183	193	182	192
AeropyrumPernix	46.13	45.33	37.63	36.83	28.54	19.32	23.14	13.93
Homo.Sapiens	43.79	41.55	37.57	35.32	27.40	24.64	24.25	21.49

2.5.2 Rolling means

A *rolling mean* (*moving mean* or *running mean*) is a calculation based on the analysis of values by creating a series of averages of several subsets of complete dataset. Given a series of numbers and a fixed subset size (*span*), the first element of the rolling mean is obtained by taking the average of the initial subset of the number series. Then the subset is modified by shifting forward, so excluding the first number of the series and including the next value in the subset. The rolling mean approach is commonly used to smooth out short-term fluctuations and highlight longer-term trends in time-series analysis.

For the purposes of this research, this particular mean will be applied on the code coverage results on different spans of codons in the DNA sequences. In particular, 15 different spans will be evaluated, equal to the numbers from 3 to 31 taken every 2 (i. e. 3, 5, 7, 9 and so on…) in order to have odd amplitudes and facilitate the understanding of the results. Since we are dealing with large datasets with many basess (and therefore many codons), to optimize the computations the R package *data.table* will be used , which "provides a high-performance version of base R's *data.frame* with syntax and feature enhancements for ease of use, convenience and programming speed." (Dowle & Srinivasan 2020)

In particular, as we can see in the corresponding code in Appendix C, the function *frollmean* has been applied, providing the vector of spans (the object *bw*) and by fixing the alignment to 'center'.

This means that the rolling means will be centered, so the results will be placed at the center of the range. For example, if the span is equal to 3 and the length of the sequence is 4, in the second position there will be the average of the code coverage of the first three codons, in the third one will be present the average of the coverage considering the second, third and fourth codons, while in the first and in the last position of the results a NaN will be provided.

Before proceeding with these analyses, as we are dealing with genomes of heterogeneous lengths, it is necessary to set a value for the minimum number of codons for the length of the sequences (*thr* parameter). In these processes, this value has been set equal to 1000. This means that only sequences in the genome lists that are at least 1000 codons long will be taken into account and, in particular, only the first 1000 codons in these sequences will be considered.

The main steps for computing the code coverage with this approach are:

- removing the sequences lower than *thr* codons;
- taking from these sequences the first *thr* codons, in order to have results of fixed length;
- computing on each sequence (with fixed length equal to 1000) of every genome the rolling mean of the coverage according to the 15 selected spans for the best code (*173*), the worst code (*192*) and the remainder code. In this way we will create (for each genome) three different 3-dimensional arrays, with dimensions equal to: $thr(1000)$, the number of considered *spans* (15) and the number of suitable *sequences* in the considered genome (that changes for every list) $\implies \underset{[thr\times spans\times sequences]}{re1(best)} \quad \underset{[thr\times spans\times sequences]}{re8(worst)} \quad \underset{[thr\times spans\times sequences]}{re3(remainder)}$
- starting from these intermediate results, computing the means considering all the available results for each sequence in the 24 genomes. In this way we will have 15 (one for each span) uni-dimensional vectors (long 1000) of mean results for every genome. The results will be stored in three different 3-dimensional arrays with dimensions: 1000 (*thr*), 15 (*spans*) and 24 (*genomes*). $\implies \underset{[1000\times 15\times 24]}{RE1(best)} \quad \underset{[1000\times 15\times 24]}{RE8(worst)} \quad \underset{[1000\times 15\times 24]}{RE3(remainder)}$

Of course, when considering different spans, different NaN will be created due to the construction of the rolling means. In fact, in Table 6, where it is displayed a small part of results obtained considering the best code (*173*) in two genomes with different spans, it is possible to check that we do not have results for the first rows depending on the amplitude of the rolling windows.

Table 6: Preview of a small part of the results: code coverage with the rolling mean approach with different spans - *AeropyrumPernix* and *Homo.Sapiens*.

Span = 3		*Span = 5*		*Span = 7*		*[...]*	*Span = 15*	
Aero.P	Homo.S	Aero.P	Homo.S	Aero.P	Homo.S		Aero.P	Homo.S
0.242	0.268							
0.303	0.388	0.291	0.303					
0.333	0.361	0.364	0.376	0.351	0.328			
0.364	0.359	0.400	0.373	0.455	0.393			
0.515	0.379	0.491	0.390	0.481	0.394			
0.576	0.413	0.582	0.406	0.532	0.407			
0.667	0.437	0.600	0.428	0.532	0.415		0.448	0.388
0.667	0.451	0.564	0.437	0.558	0.427		0.455	0.419
0.485	0.438	0.582	0.440	0.558	0.439		0.448	0.420

A limit of this approach is the fact that we are artificially selecting only a subset of the available sequences (in particular the ones longer than thr codons) and that we are obtaining results only for the first thr codons. Even if 1000 is a quite big number that allows us to have statistically significant estimates, this could be a problem since we are dealing with genomes with an average sequence length that is around 1000 (for example, 962.96 for *AeropyrumPernix* or 1249.08 for *Homo.Sapiens*) but also with genomes with a larger average sequence length (for example, 2128.88 for *Drosophila.Melanogaster*). These non-representative issues will be solved in the next approaches for computing the codon usage.

2.5.3 Considering every sequence

In order to discuss how to obtain the results for code coverage taking into account all available DNA sequences for the genomes under consideration, it is necessary to briefly focus on how codon usage is calculated for this approach.

2.5.3.1 Codon usage for every sequence A solution the non-representative issue of the rolling mean approach with *thr* is offered by considering the codon usage results for every sequence among all the available for genomes in analysis. This means that, as we can see from

the third column in Table 3, we will have 713 vectors of results for *AeropyrumPernix*, 1150 for *Thermoplasma.acidophilum* and so on until 151245 for *Arabidopsis.Thaliana*.

The main steps of this computation are:

- extracting the maximum number of sequences among the genomes taken into account. Looking back at third column of Table 3, the general value would be 151245 (of *Arabidopsis.Thaliana*), but during the analysis genomes have been split for making the computations less time-consuming, hence there is a part of code that computes this value to avoid any issue $\implies maxlenseq$
- computing the codon usage on every sequence of the genomes taken into account and storing the results in a 3-dimensional array with dimensions equal to: the number of genomes of interest (24), the length of the longest sequence among all the sequences of the genomes of interest previously described and the 64 codons of the genetic code $\implies \underset{[24\times maxlenseq\times 64]}{cu\_each}$

Since all the available sequences in data are taken into account, the shortest sequences (the very small ones) could create problems in computations, because they present too many values equal to zero for the usage of whole codon blocks. This problem has been solved in two different ways:

A - by considering for the computations all the available sequences, recognizing the problematic sequences (by simply checking the dimensions of the results) and substituting them with NAs. This method also allows to save the length of the problematic sequences, in order to understand if these are actually short and how many sequences have been excluded from the analysis;

B - by filtering before the analysis the shortest sequences according to an arbitrary threshold (chosen equal to 300 codons) and removing them a priori from computations.

Both the methods solve the problem and lead to almost equal results, therefore in the following there will be only presented and discussed the values obtained by considering the solving method A, as it allows us to evaluate all the information available in our dataset.

In Table 7 is displayed a small part of the *cu_each* object. In particular, the results for the first 5 sequences of the genomes *AeropyrumPernix* and *Homo.Sapiens* are present, but only the first 10 (out of 64) codons are displayed. We can see that there are *NAs* for the second sequence of

Homo.Sapiens, hence this is an example of a problematic sequence. In fact, this sequence is only 363 basis (therefore, 131 codons) long.

Table 7: Preview of a small part of the results: codon usage for the first 5 sequences and 10 codons only - *AeropyrumPernix* and *Homo.Sapiens*.

	AAA	AAC	AAG	AAT	ACA	ACC	ACG	ACT	AGA	AGC
AeropyrumPernix										
1	0.065	0.000	0.020	0.025	0.005	0.005	0.005	0.005	0.035	0.020
2	0.005	0.068	0.068	0.000	0.010	0.005	0.016	0.000	0.000	0.016
3	0.017	0.023	0.048	0.009	0.012	0.009	0.006	0.010	0.027	0.032
4	0.003	0.012	0.063	0.003	0.003	0.009	0.012	0.007	0.009	0.028
5	0.014	0.014	0.033	0.014	0.014	0.014	0.009	0.024	0.005	0.038
Homo.Sapiens										
1	0.022	0.011	0.022	0.011	0.000	0.011	0.055	0.000	0.000	0.022
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	0.028	0.012	0.030	0.022	0.010	0.010	0.008	0.010	0.016	0.012
4	0.000	0.027	0.045	0.000	0.003	0.024	0.024	0.000	0.000	0.013
5	0.023	0.010	0.035	0.012	0.017	0.004	0.004	0.023	0.004	0.006

2.5.3.2 Code coverage for every sequence Starting from the codon usage for every sequence described in the previous section, calculating the coverage of the different circular codes is straightforward. In fact, the function `cover` from `mathDNA` will be used, similarly for what has been done for code coverage on whole genomes, but considering only one sequence at a time instead of all the sequences of one genomes merged into a very long one.

Since we are dealing with many iterations, instead of computing the coverages for all the 216 circular code groups, we will focus on only the 27 best and 27 worst codes (hence, only the ones in the first and last column in Table 1). In this way, we can easily obtain the result for the remainder group of codon as $1 - (coverage_{best} + coverage_{worst})$.

The results will be stored in a 3-dimensional array with dimensions equal to:

- 24, the number of genomes taken into account;
- 151245, the maximum number of sequences among all the genomes in analysis (that corresponds to the number of sequences for *Arabidopsis.Thaliana*);

- 54 $(27 + 27)$, that is the number of best and worst code groups for which the coverage are computed.

$$\Longrightarrow \underset{[24\times 151245\times 54]}{res\_allseqs}$$

In Table 8 it is displayed a small part of the results of this process.

Table 8: Preview of a small part of the results: code coverages of the first three pairs of best and worst codes in the first 5 sequences only - *AeropyrumPernix* and *Drosophila.melanogaster*.

	Best codes			Worst codes		
	173	23	98	192	87	53
AeropyrumPernix						
1	0.335	0.325	0.330	0.225	0.245	0.235
2	0.599	0.536	0.516	0.062	0.052	0.042
3	0.459	0.443	0.424	0.166	0.174	0.165
4	0.495	0.464	0.463	0.103	0.101	0.093
5	0.443	0.448	0.415	0.146	0.137	0.165
Drosophila.melanogaster						
1	0.481	0.480	0.483	0.179	0.182	0.187
2	0.606	0.587	0.552	0.082	0.085	0.110
3	0.615	0.604	0.581	0.099	0.099	0.094
4	0.484	0.457	0.471	0.199	0.208	0.202
5	0.502	0.503	0.484	0.178	0.190	0.202

This approach makes it possible to understand how much variability there is in the coverage of different sequences of the same genome, and allows us to look more closely at the relationships between different codes (*best*, *worst* and *remainder*) on the same sequence.

2.5.4 By position

In order to inspect more in depth if there are particular results related to the beginning and the end of the sequences, it could be useful also to consider the code coverage results for every position in the genomes. Therefore, instead of considering global results for the entire sequences in the genomes, we will focus on the results for the code coverages for the codons that are placed in the same position in the different sequences of every genome.

Let's consider, as an example, an imaginary and simple genome that is made up of only three sequences, with lengths equal to 3, 5 and 4 codons (displayed in Table 9). Let's imagine to compute the coverage for a code group that is made up only of AAA. Following the reasoning in the previous sections, in order to compute the code coverage for every sequence we have to check how many times AAA is present in every sequence (hence, *in every row*) and we will have a result for every sequence in the genome. Instead, for computing the codon usage by position, we have to study how many times AAA is present in every position of the sequences (hence, *in every column*) and we will have a result as long as the length of the longest sequence in the genome.

Table 9: Example of positional approach

	1	2	3	4	5		By sequence
Sequence 1	AAA	TTT	GGG			→	1/3
Sequence 2	AAA	CCC	GGG	CCC	AAA	→	2/5
Sequence 3	TTT	AAA	CCC	AAA		→	2/4
	↓	↓	↓	↓	↓		
By position	2/3	1/3	0/3	1/2	1/1		

Since we are interested in the behavior of the code coverage at the beginning and at the end of the sequences, for this computations we will evaluate the entire sequences instead of the cut ones.

The algorithm for obtaining the code coverage results by position is similar to the one described for the *rolling means* approach, but in this case we apply the base function `mean` (instead of `frollmean`) by columns (instead of rows). Furthermore, to store the results we need to extract the length of the longest sequence among all the ones of the genomes of interest. This value, that will be called *maxlencod*, is different from *maxlenseq*, which is the maximum value for the number of sequences in the considered genomes (and it has been computed for the codon usage for every

sequence).

We will do the computations only for the best and worst code groups in general, hence, for the code group 173 and 192, and for the remainder of these two. The results, then, will be three different matrix with dimensions equal to the genomes under analysis (24) and *maxlencod*, that refers to the three group of codes considered. $\implies$ $\underset{[24\times maxlencod]}{RESpos1(best)}$ $\underset{[24\times maxlencod]}{RESpos8(worst)}$ $\underset{[24\times maxlencod]}{RESpos3(remainder)}$

As a consequence, if in genome A the longest sequence is shorter than the longest sequence in genome B, the results for genome A will present NAs. In fact, in Table 10, which offers an example of the results related to the genomes *AeropyrumPernix* and *Helicobacter.pylori*, it is possible to see that after the position 1332 there are no results for the first genome under consideration. This is due to the fact that the longest sequence in *AeropyrumPernix* is 1332 codons long. Furthermore, from the last three results for this genome (in positions 1330-1331-1332) it is easy to infer that there is only one sequence with that length, since there are only zeros and ones.

Table 10: Preview of a small part of the results: code coverages computed on the first and last positions - *AeropyrumPernix* and *Helicobacter.pylori*.

	1	2	3	4	5	[...]	1330	1331	1332	1333	1334	1335
AeropyrumPernix												
173 (Best)	0	0.41	0.41	0.45	0.39		0.00	0.00	0.00			
Remainder	1	0.40	0.39	0.42	0.44		1.00	1.00	0.00			
192 (Worst)	0	0.20	0.20	0.12	0.17		0.00	0.00	1.00			
Helicobacter.pylori												
173 (Best)	0	0.28	0.36	0.34	0.38		0.36	0.46	0.46	0.27	0.64	0.64
Remainder	1	0.52	0.48	0.45	0.42		0.46	0.36	0.36	0.36	0.18	0.18
192 (Worst)	0	0.20	0.16	0.21	0.20		0.18	0.18	0.18	0.36	0.18	0.18

2.6 Bootstrap test

The bootstrap test previously presented will be applied considering 10,000 bootstrap replications. Moreover, in order to provide universality to the results, only the most general hypothesis that the occurrence of circular codes is uniform will be considered and all 16 disjoint pairs of *best* and *worst* codes (the bold rows in Table 1) will be taken into account.

The `codtest` function (in **Appendix C**) will be used to run the test, with the following parameters:

- `B = 10000`, the bootstrap replications;
- `quant = (0.0001, 0.9999)`, the quantile corresponding to the number of replications;
- `replace = FALSE`, to create subsets without repeating codons;
- `weight = FALSE`, for the codons to be extracted from a uniform distribution;
- the parameters `cod` and `xf`, which correspond respectively to the vector of codons constituting the *best* code and to the codon usage of the genome under consideration, vary according to the pair of codes and the genome under consideration.

3 Results

3.1 Differences in codon usage between entire and cut sequences

In this section, the results of **codon usage calculated on whole genomes** will be presented and discussed.

Table 11 contains the values of the differences between the codon usage calculated on the entire sequences and on the sequences without initial and final parts for the *model* genomes. In particular, the codons that are part of the best (*173*) and worst (*192*) codons and the *ATG* codon, which encodes *methionine* but also indicates the start of the protein coding region, are highlighted in this table. In addition, in Table 19 in **Appendix A**, it is possible to observe the individual values for entire and cut sequences, as well as the difference between them, for all the 64 codons in the genetic code.

Looking at Table 19, it can be seen that, with the exception of the *ATG* codon, for which we expect it to be systematically more present in whole sequences as it is always present at the beginning of the sequence, there does not seem to be any particular association between individual codons and differences in code coverage, as the values are almost all very close to zero. However, looking carefully at Table 11, which groups the codons according to their code group *173* and *192* membership, a particular feature of the results emerges. In fact, it is not difficult to notice that the vast majority of differences for codons belonging to group *173* correspond to a negative value, while for codons belonging to group *192* the positive and negative values are balanced. It can be deduced, therefore, that the initial part of the sequences is characterised by the absence of codons from group *173*, since they systematically appear more frequently in the cut sequences. This particular observation, which is in line with expected results from previous studies on code coverage at the beginning of sequences, will also be confirmed by forthcoming results.

Table 11: Differences in codon usage between entire and cut sequences for the codons in the best (*173*) and worst (*192*) codes and for the start codon - *model* genomes. These differences have been calculated using the formula below.

	AePe	HePy	EsCo	PlFa	DrMe	HoSa	Mean
Best code (173)							
AAC	-0.650	-0.441	-0.219	-0.016	-0.053	-0.277	-0.276
GTT	0.129	0.167	0.001	0.031	-0.043	-0.246	0.007
AAT	-0.089	-0.168	0.174	-1.605	-0.017	-0.266	-0.329
ATT	0.045	0.084	0.050	-0.096	-0.011	-0.277	-0.034
ATC	-0.098	-0.429	-0.252	0.027	-0.135	-0.250	-0.189
GAT	-0.008	-0.651	-0.656	-0.628	-0.131	-0.376	-0.408
CAC	-0.482	-0.050	-0.096	0.036	-0.107	-0.065	-0.127
GTG	-0.482	-0.749	-0.708	0.024	-0.179	-0.098	-0.365
CAG	0.043	-0.090	-0.436	0.020	-0.304	-0.136	-0.151
CTG	-0.025	-0.012	-0.922	0.056	-0.098	-0.119	-0.187
CTC	-0.036	-0.011	0.016	0.063	0.013	0.007	0.009
GAG	-0.405	-0.210	-0.114	0.026	-0.401	-0.268	-0.229
GAA	0.201	-0.311	-0.517	-0.410	-0.111	-0.307	-0.243
TTC	-0.704	-0.080	-0.178	0.114	-0.044	-0.325	-0.203
GAC	-0.658	-0.278	-0.377	0.021	-0.114	-0.427	-0.306
GTC	-0.409	-0.169	-0.232	0.039	-0.046	-0.197	-0.169
GCC	-0.887	-0.387	-0.571	0.033	-0.120	0.262	-0.278
GGC	-0.165	-0.696	-0.874	0.008	-0.135	-0.050	-0.319
GTA	-0.016	0.121	0.053	-0.036	-0.015	-0.062	0.007
TAC	-0.902	-0.233	-0.221	0.129	-0.071	-0.311	-0.268
Worst code (192)							
ACA	0.286	0.095	0.336	-0.106	-0.002	-0.150	0.077
TGT	0.051	0.061	-0.013	-0.096	-0.018	-0.028	-0.007
ACT	-0.245	-0.162	0.125	-0.066	-0.055	-0.142	-0.091
AGT	0.244	0.054	0.098	-0.094	-0.010	-0.061	0.038
AGA	0.309	0.277	0.295	0.064	0.043	0.025	0.169
TCT	0.181	-0.011	0.048	-0.078	-0.031	0.004	0.019
CCA	0.258	0.124	0.052	-0.026	-0.109	-0.043	0.043
TGG	0.233	0.021	-0.169	0.039	0.045	-0.076	0.016
CCG	-0.182	-0.124	-0.620	0.007	-0.083	0.182	-0.137
CGG	0.101	-0.030	0.016	0.008	-0.018	0.070	0.025
CGA	0.090	0.099	0.179	0.005	0.004	0.048	0.071
TCG	0.089	-0.010	-0.082	0.041	-0.003	0.253	0.048
GCA	-0.354	0.074	0.046	0.017	0.022	-0.045	-0.040
TGC	0.049	-0.127	-0.087	0.035	-0.082	0.001	-0.035
GGA	0.232	0.069	0.074	-0.041	-0.154	-0.042	0.023
TCC	-0.238	-0.061	-0.014	0.056	-0.106	0.201	-0.027
TAA	0.603	1.594	1.784	0.875	0.577	0.574	1.001
TTA	0.042	0.290	0.460	-0.063	0.089	-0.093	0.121
TCA	0.004	0.068	0.181	-0.017	0.027	-0.063	0.033
TGA	0.612	0.790	0.964	0.268	0.349	1.108	0.682
ATG	2.598	2.451	2.580	1.156	1.348	2.210	2.057

$$\text{difference} = (cu_i^{ent} * 1000) - (cu_i^{cut} * 1000)$$

3.2 Differences in code coverage between entire and cut sequences

In this section, the results of **code coverage calculated on whole genomes** will be presented and discussed.

Tables 12, 13 and 14 show the differences between the results obtained considering the whole sequences and the sequences without the first and last codons. In particular, Table 12 shows the differences in results for the 27 best codons, Table 13 for the 27 worst codons and Table 14 for the 27 remainder codons. In **Appendix A** are displayed the results for all genomes under analysis (in Tables 22, 23 and 24).

Observing the values in the tables, it is clear that the differences for the coverage of the best codes (i.e. those in the first column of Table 1) are all negative, while those for the coverage of the worst codes (last column in Table 1) and the remainder codes are almost all positive. It can be stated with certainty, therefore, that in the initial and final parts of the sequences there are generally fewer codons present than in the 27 best groups of circular codes. In addition, it can be observed that the values for the differences in the code coverage of the worst groups are, in almost all cases, greater (both on average and taken individually) than those for the code coverage of the remainder groups. This leads us to conclude that in the initial and final parts of the sequences the systematic lower presence of codons from the best groups corresponds to a systematic higher presence of codons from the worst groups, and that this trend is not only due to the fact that we consider disjointed groups of codons, but suggests an actual grouping of codons on the basis of properties related to circular codes theory. It should also be emphasised that this interesting result suggests a universal property, as it applies to **all** genomes analysed (Tables 22, 23 and 24).

Table 12: Differences in code coverage between entire and cut sequences for the 27 best codes - *model* genomes. These differences have been calculated using the formula below.

	A.Pernix	H.pylori	E.coli	P.Falcip	D.melano	H.Sapiens	Mean
173	-0.560	-0.460	-0.608	-0.217	-0.212	-0.379	**-0.406**
23	-0.433	-0.413	-0.644	-0.231	-0.217	-0.380	**-0.386**
98	-0.479	-0.453	-0.545	-0.214	-0.203	-0.345	**-0.373**
25	-0.468	-0.459	-0.569	-0.242	-0.210	-0.362	**-0.385**
20	-0.342	-0.413	-0.605	-0.256	-0.215	-0.363	**-0.366**
166	-0.549	-0.388	-0.476	-0.154	-0.194	-0.360	**-0.354**
4	-0.423	-0.342	-0.513	-0.168	-0.198	-0.362	**-0.334**
30	-0.364	-0.216	-0.442	0.021	-0.147	-0.278	**-0.238**
117	-0.458	-0.388	-0.438	-0.179	-0.192	-0.343	**-0.333**
111	-0.331	-0.342	-0.474	-0.193	-0.196	-0.345	**-0.314**
22	-0.390	-0.390	-0.628	-0.238	-0.204	-0.338	**-0.365**
172	-0.626	-0.551	-0.494	-0.219	-0.179	-0.345	**-0.402**
21	-0.500	-0.505	-0.531	-0.233	-0.183	-0.347	**-0.383**
24	-0.535	-0.551	-0.455	-0.244	-0.177	-0.328	**-0.382**
97	-0.546	-0.545	-0.432	-0.216	-0.169	-0.312	**-0.370**
171	-0.408	-0.505	-0.492	-0.258	-0.181	-0.329	**-0.362**
3	-0.490	-0.434	-0.399	-0.170	-0.165	-0.328	**-0.331**
165	-0.616	-0.480	-0.363	-0.156	-0.160	-0.327	**-0.350**
26	-0.583	-0.528	-0.478	-0.226	-0.166	-0.303	**-0.381**
123	-0.456	-0.482	-0.515	-0.240	-0.171	-0.304	**-0.361**
115	-0.524	-0.480	-0.324	-0.181	-0.158	-0.310	**-0.330**
161	-0.398	-0.434	-0.360	-0.196	-0.163	-0.311	**-0.310**
122	-0.365	-0.482	-0.476	-0.266	-0.168	-0.287	**-0.341**
41	-0.312	-0.390	-0.306	-0.217	-0.155	-0.277	**-0.276**
107	-0.325	-0.396	-0.328	-0.201	-0.147	-0.249	**-0.274**
198	-0.311	-0.168	-0.257	-0.003	-0.105	-0.222	**-0.178**
137	-0.279	-0.439	-0.422	-0.287	-0.161	-0.253	**-0.307**

$$\text{difference} = (C_b^{ent} * 100) - (C_b^{cut} * 100)$$

Table 13: Differences in code coverage between entire and cut sequences for the 27 worst codes - *model* genomes. These differences have been calculated using the formula below.

	A.Pernix	H.pylori	E.coli	P.Falcip	D.melano	H.Sapiens	Mean
192	0.236	0.309	0.367	0.083	0.049	0.172	**0.203**
87	0.231	0.272	0.376	0.108	0.051	0.172	**0.202**
53	0.217	0.261	0.262	0.074	0.022	0.167	**0.167**
56	0.269	0.304	0.236	0.085	0.014	0.151	**0.176**
89	0.264	0.267	0.244	0.110	0.017	0.150	**0.175**
191	0.280	0.268	0.381	0.086	0.067	0.160	**0.207**
86	0.275	0.231	0.390	0.110	0.069	0.159	**0.206**
88	0.134	0.179	0.292	0.110	0.035	0.167	**0.153**
135	0.312	0.263	0.250	0.088	0.033	0.139	**0.181**
145	0.307	0.226	0.258	0.113	0.035	0.138	**0.180**
99	0.132	0.206	0.272	0.079	0.034	0.105	**0.138**
195	0.417	0.372	0.374	0.117	0.108	0.257	**0.274**
91	0.412	0.335	0.383	0.142	0.110	0.256	**0.273**
57	0.450	0.367	0.243	0.119	0.074	0.236	**0.248**
54	0.398	0.324	0.269	0.108	0.082	0.251	**0.239**
208	0.445	0.331	0.251	0.144	0.076	0.235	**0.247**
90	0.456	0.294	0.397	0.144	0.128	0.244	**0.277**
194	0.461	0.331	0.388	0.120	0.126	0.245	**0.278**
66	0.318	0.306	0.270	0.089	0.092	0.191	**0.211**
147	0.313	0.270	0.279	0.113	0.094	0.190	**0.210**
136	0.493	0.326	0.257	0.122	0.092	0.224	**0.252**
207	0.488	0.289	0.265	0.147	0.094	0.223	**0.251**
149	0.345	0.265	0.147	0.116	0.059	0.169	**0.184**
93	0.343	0.195	0.196	0.151	0.072	0.218	**0.196**
146	0.679	0.437	0.409	0.216	0.185	0.328	**0.376**
210	-0.103	0.085	0.184	0.140	0.011	0.146	**0.077**
154	0.200	0.170	0.079	0.120	0.037	0.165	**0.128**

$$\text{difference} = (C_w^{ent} * 100) - (C_w^{cut} * 100)$$

Table 14: Differences in code coverage between entire and cut sequences for the 27 *remainder* sets - *model* genomes. These differences have been calculated using the formula below.

	A.Pernix	H.pylori	E.coli	P.Falcip	D.melano	H.Sapiens	Mean
r_1	0.323	0.151	0.241	0.134	0.164	0.206	**0.203**
r_2	0.202	0.141	0.269	0.123	0.166	0.209	**0.185**
r_3	0.262	0.192	0.283	0.140	0.181	0.179	**0.206**
r_4	0.199	0.155	0.333	0.157	0.196	0.210	**0.208**
r_5	0.078	0.146	0.361	0.146	0.198	0.213	**0.190**
r_6	0.269	0.121	0.095	0.068	0.127	0.200	**0.147**
r_7	0.148	0.111	0.123	0.058	0.129	0.203	**0.129**
r_8	0.230	0.037	0.151	-0.131	0.112	0.111	**0.085**
r_9	0.145	0.125	0.188	0.091	0.159	0.204	**0.152**
r_10	0.024	0.116	0.216	0.081	0.161	0.207	**0.134**
r_11	0.258	0.184	0.357	0.159	0.170	0.233	**0.227**
r_12	0.209	0.179	0.120	0.102	0.071	0.088	**0.128**
r_13	0.088	0.170	0.148	0.091	0.073	0.090	**0.110**
r_14	0.085	0.184	0.213	0.125	0.103	0.092	**0.134**
r_15	0.147	0.221	0.163	0.108	0.088	0.060	**0.131**
r_16	-0.036	0.174	0.240	0.114	0.105	0.094	**0.115**
r_17	0.034	0.140	0.002	0.026	0.036	0.084	**0.054**
r_18	0.155	0.149	-0.026	0.036	0.034	0.082	**0.072**
r_19	0.265	0.222	0.208	0.137	0.075	0.112	**0.170**
r_20	0.143	0.213	0.236	0.127	0.077	0.114	**0.152**
r_21	0.031	0.154	0.067	0.059	0.066	0.086	**0.077**
r_22	-0.090	0.144	0.095	0.049	0.068	0.088	**0.059**
r_23	0.019	0.217	0.328	0.150	0.109	0.118	**0.157**
r_24	-0.031	0.196	0.110	0.066	0.083	0.058	**0.080**
r_25	-0.355	-0.041	-0.081	-0.015	-0.038	-0.079	**-0.101**
r_26	0.414	0.084	0.073	-0.137	0.094	0.076	**0.101**
r_27	0.079	0.268	0.343	0.167	0.124	0.088	**0.178**

$$\text{difference} = (C_r^{ent} * 100) - (C_r^{cut} * 100)$$

3.3 A critical investigation on the rolling means approach

In this section, the **code coverage** results obtained considering the **rolling means approach** will be presented and discussed.

In Figures 15 and 16 are displayed the distributions of the results for the code coverage of the best code group (*173*) obtained through the *rolling means* approach. There are two different boxplots for each span of the rolling windows: the one in green refers to the code coverage calculated on the sequences without initial and final part, while the one in blue refers to the results obtained considering the entire sequences. The blue and the red line, on the other hand, indicate the value for the code coverage calculated on the whole genome taking into account the cut sequences and the entire sequences, respectively. So, to be clear, the blue line for the graphs referring to the code coverage for the best code will correspond to the sum of the codon usage results corresponding to the 20 codons that constitute the code group *173* (listed in the first column of the Table 2).

Figures 17 and 18 show the results for the worst code (*192*). Also in this case we have two boxplots referring to the cut and entire sequences and there are the two lines for the code coverage calculated on the whole genome.

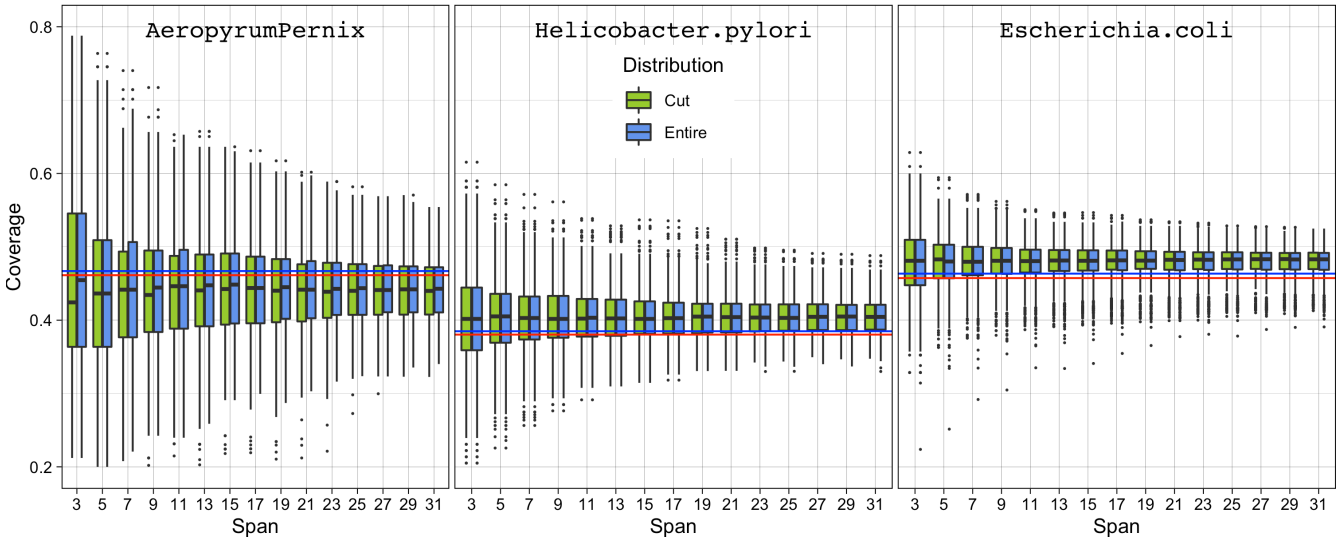


Figure 15: Coverages of the best code (*173*) calculated using the rolling means approach: cut (green box) vs entire (blue box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli.*

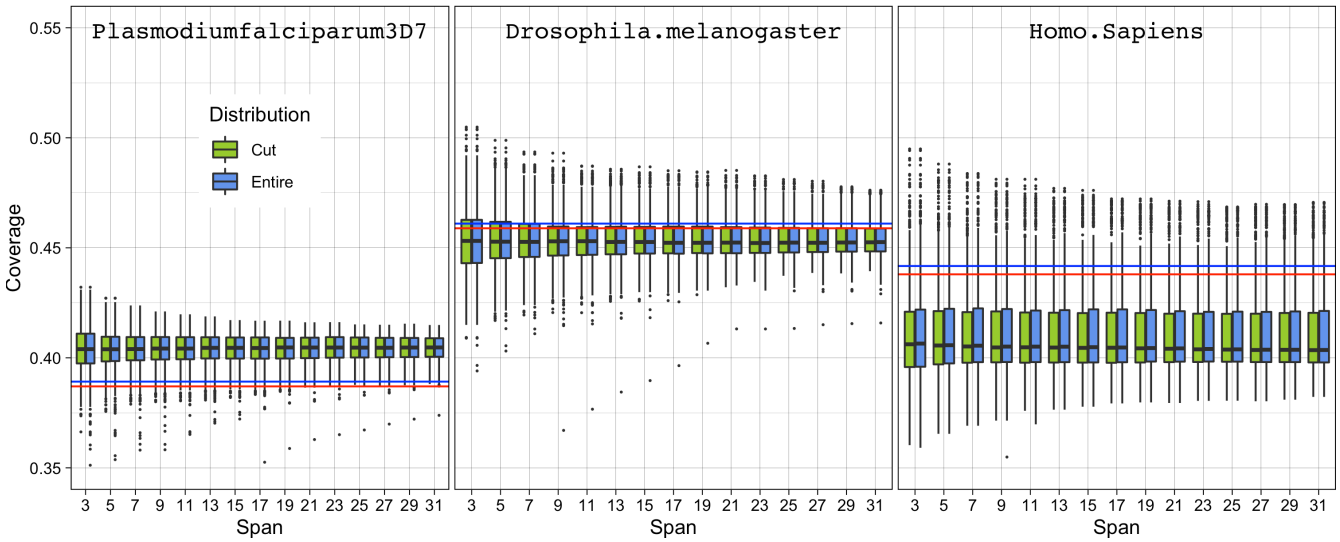


Figure 16: Coverages of the best code (*173*) calculated using the rolling means approach: cut (green box) vs entire (blue box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens.*

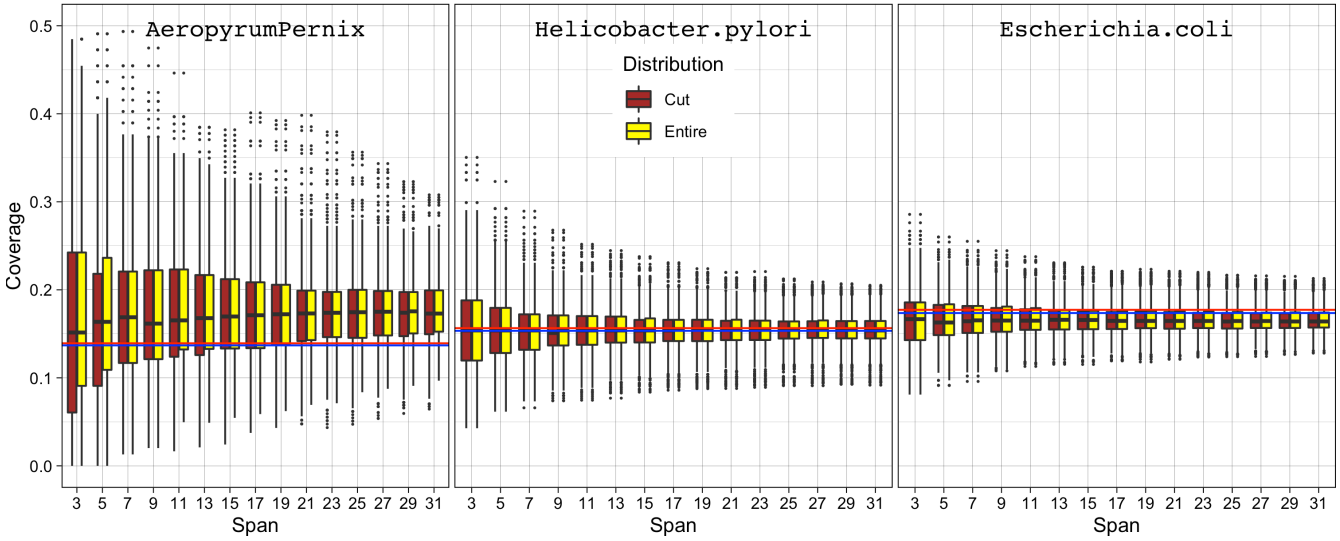


Figure 17: Coverages of the worst code (*192*) calculated using the rolling means approach: cut (red box) vs entire (yellow box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli.*

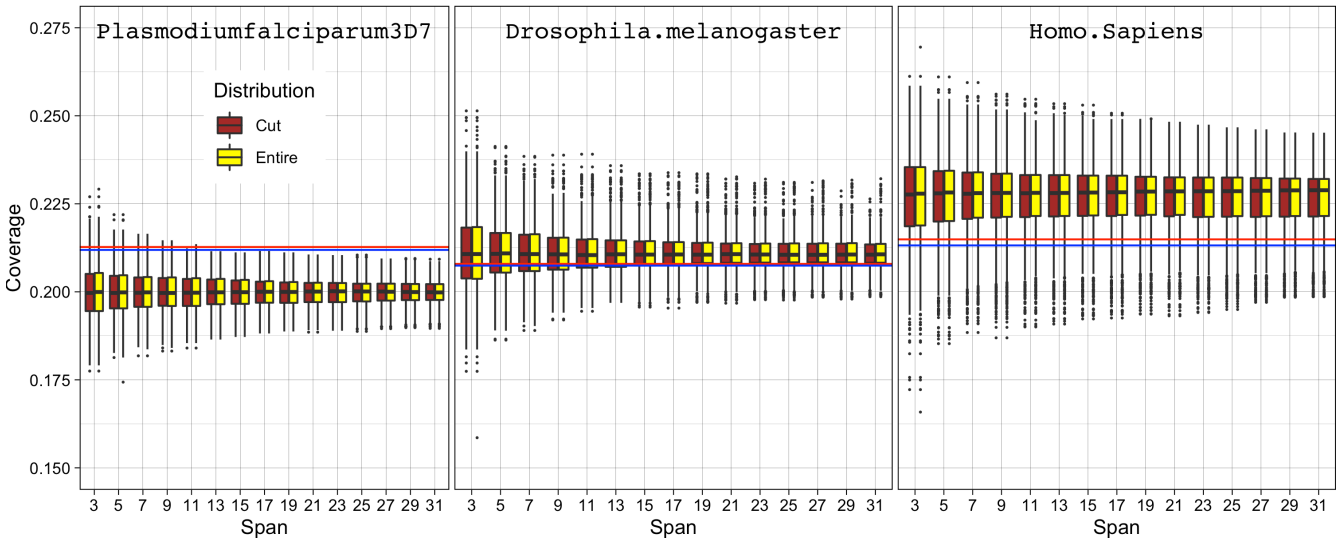


Figure 18: Coverages of the best code (*192*) calculated using the rolling means approach: cut (red box) vs entire (yellow box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens.*

Looking at the graphs there are several comments that can be made:

- as the span of the rolling windows increases, the distributions tend to be less and less wide, which is an expected result given the construction of the *rolling means*. In particular, however, it can be observed that the results tend to stabilise already around spans 9-11.
- the distributions of the results considering entire sequences or those without initial and final parts are almost identical, especially when larger genomes are taken into account.
- the width of the distribution tends to be more limited as the number of sequences in the genome increases (it should be noted that the limit on the y-axis varies between the graphs of the *AeropyrumPernix*, *Helicobacter.pylori*, *Escherichia.coli* group and those of the *Plasmodiumfalciparum3D7*, *Drosophila.melanogaster*, *Homo.Sapiens* group), which can be explained by the fact that the larger the sample, the less variable the results.
- the boxplots are not centred on what can be considered as the benchmark for these results, i.e. the code coverage calculated over the whole genome (red and blue lines). This is due to the fact that, as previously explained, when calculating code coverage with the *rolling means* approach, only sequences longer than 1000 codons are artificially selected and only the first 1000 codons of these are analysed. This particular result may suggest that the code

coverage results may be correlated with the length of the sequences under analysis, as will be discussed in the following paragraphs.

3.4 Relationships between best and worst codes coverage on individual sequences

In this section, the **code coverage** results obtained by **considering all available sequences** in the genomes of interest are evaluated.

In Figures from 19 to 24 are displayed the scatterplots of the results for the code coverage obtained considering every sequence in the *model genomes*. Thus, each point in the scatterplots corresponds to a particular sequence in the genome under consideration; in fact, it can easily be observed that the number of points in the graphs grows as the sequences in the genome increase. In particular, every figure is made up of three scatterplots:

- in the first one, on the left, there are the code coverage results for the best code (*173*) on the x-axis and for the worst code (*192*) on the y-axis;
- in the second one, in the middle, there are the code coverage results for the best code (*173*) on the x-axis and for the remainder code on the y-axis;
- in the third one, on the right, there are the the code coverage results for the worst code (*192*) on the x-axis and for the remainder code on the y-axis.

In this part only the results for the first best (*173*), worst (*192*) and remainder codes are presented, but the results have been obtained for all the 27 best, worst and remainder codes (in the first and last columns in Table 1). In **Appendix B** it is possible to observe the $81(27 \times 3)$ scatterplots for all the combinations of best vs worst, best vs remainder and worst vs remainder for the genomes *Drosophila.melanogaster* and *Homo.Sapiens* (in Figures 35 and 36).

Moreover, in the plots it is also possible to observe the quadratic curve fitted on the points using functions in the `ggplot2` package (Wickham 2016) with the coefficient of determination (R^2),

i.e. the proportion of the variation in the dependent variable that is predictable from the independent variable taking into account the quadratic model $y = ax + bx^2 + c$. In addition, Tables 15 (below), 20 and 21 (in **Appendix B**) present the coefficients of determination (again for a quadratic model) for the three possible combinations (best vs worst, best vs remainder, worst vs remainder) for the first three best and worst codes, i.e. pairs *173-192*, *23-87* and *98-53* respectively.

It should be noted that in Figures from 19 to 24 and 35 - 36, the term *complement* is intended as a synonym for *remainder* code. It should not be confused, then, with the *complementary DNA sequence*, i.e. the sequence obtained by substituting the nitrogenous bases on a string with the complementary ones.

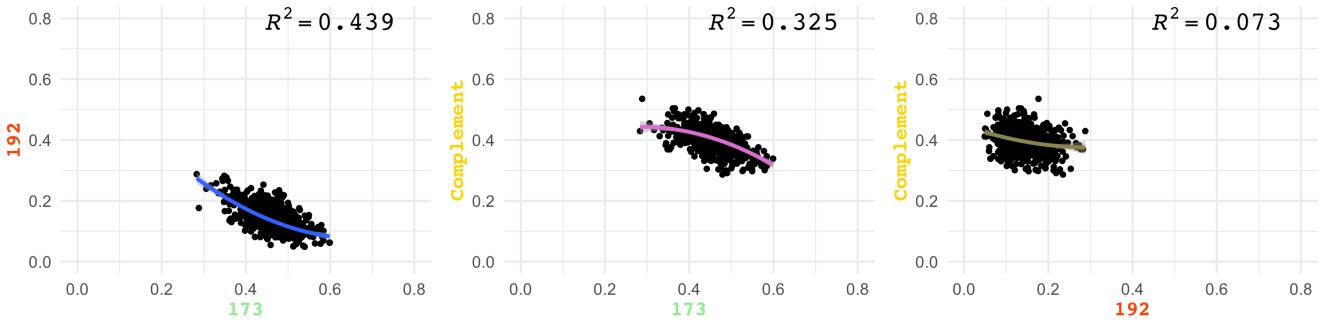


Figure 19: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *AeropyrumPernix*.

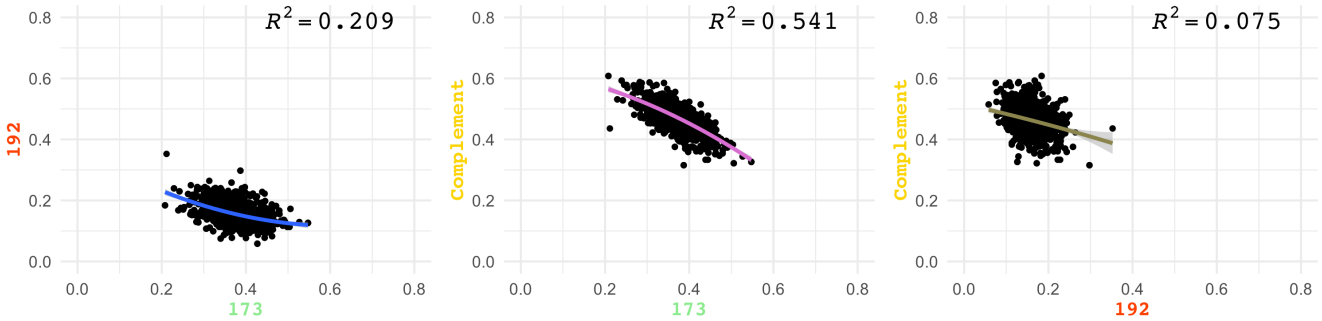


Figure 20: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Helicobacter.pylori*.

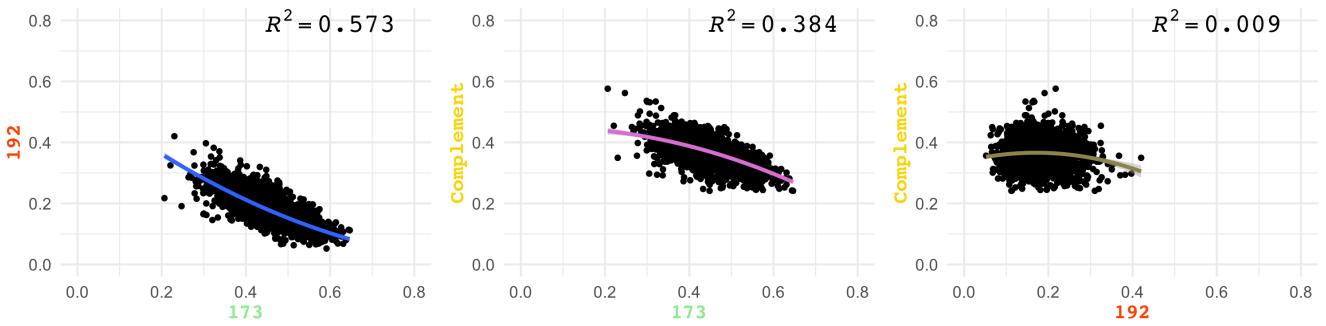


Figure 21: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Escherichia.coli*.

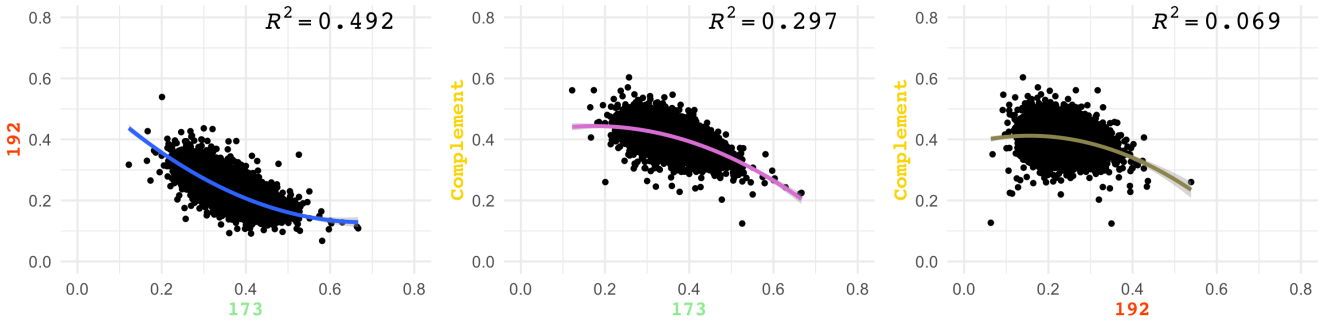


Figure 22: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Plasmodiumfalciparum3D7*.

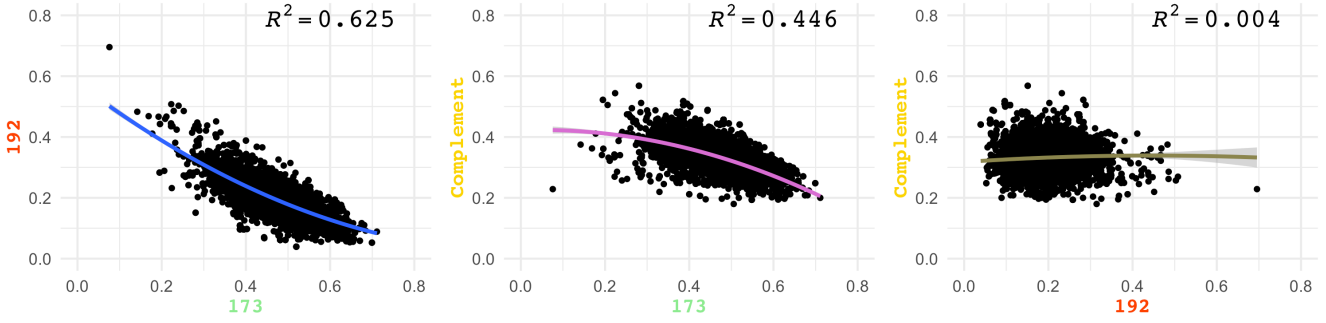


Figure 23: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Drosophila.melanogaster*.

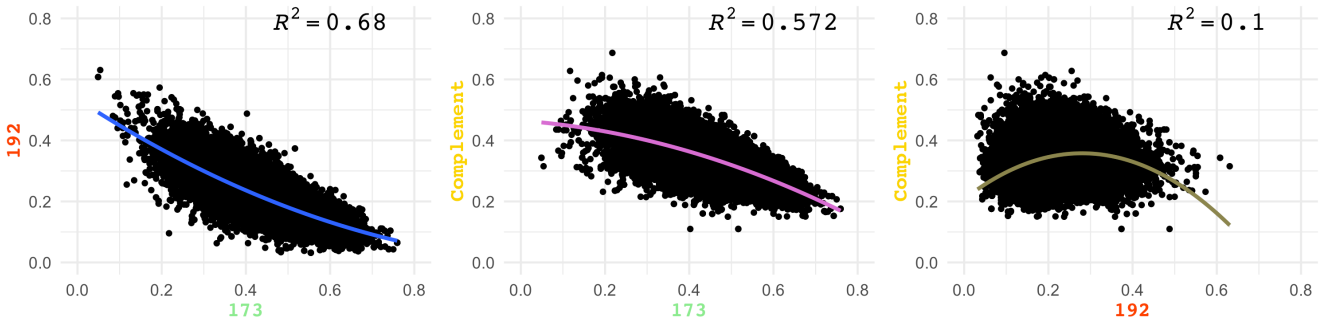


Figure 24: Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Homo.Sapiens*.

Looking at the scatterplots in Figures from 19 to 24 and the different values for R^2 in Tables 15, 20 and 21, it is clear that the relationship between the best with the worst code is generally grater than the ones between the best and worst code with the remainder one. In particular, it can be observed that the values for R^2 between the best and worst codes are always significantly higher than those linking the worst code to the remainder. This observation holds for almost all cases also when comparing the values obtained considering the *best-worst* and the *best-remainder* code pairs. Moreover, it is of great interest that the worst and remainder groups are almost uncorrelated (R^2 values are very close to zero).

These observations make it possible to generalise the interpretation given above by examining the different results for code coverage. In fact, if before we observed a correspondence between the groups obtained by evaluating the theory of circular codes and differences in the results referring only to the initial and final parts of the sequences, now it is possible to draw general conclusions regarding the whole sequences. It can be observed, in fact, that the codons present in the best codons are systematically more present and that those that constitute the worst groups are systematically less present. The significant finding is that this inverse relationship is not due merely to the fact that disjointed subsets of the total group of 64 codons are considered, since the correlation between the considered codes and the remainder one are not equally strong (indeed, one of them is almost null).

It can therefore be concluded that there is a general grouping of codons present in DNA coding sequences according to the properties associated with the theory of circular codes. As in the previous discussions, these results apply to **all** organisms considered in the analyses.

Table 15: R-squared of quadratic regression: codes *173*, *192* and remainder- all the genomes under analysis.

	173-192	173-rem	192-rem
AeropyrumPernix	0.439	0.325	0.073
Thermoplasma.acidophilum	0.451	0.319	0.054
P.Horikoshii	0.618	0.099	0.221
Pyrococcus	0.375	0.269	0.133
Staphylococcus.aureus	0.268	0.316	0.187
Helicobacter.pylori	0.209	0.541	0.075
Methanosarcina	0.379	0.432	0.038
Archaeoglobus	0.350	0.356	0.105
Escherichia.coli	0.573	0.384	0.009
Streptomyces.coelicolorA3	0.559	0.543	0.010
M.Xanthus	0.561	0.505	0.008
Caenorhabditis.elegans	0.640	0.150	0.146
Sulfolobus.solfataricus	0.511	0.163	0.128
Schizosaccharomyces.Pombe	0.424	0.046	0.373
Plasmodiumfalciparum3D7	0.492	0.297	0.069
Leishmania.major	0.700	0.315	0.011
Drosophila.melanogaster	0.625	0.446	0.004
DanioRerio	0.545	0.330	0.016
ZeaMays	0.683	0.583	0.073
OryzaSativa	0.663	0.541	0.085
Bacillus.subtilis	0.351	0.444	0.048
MusMusculus	0.632	0.404	0.002
Homo.Sapiens	0.680	0.572	0.100
Arabidopsis.Thaliana	0.403	0.197	0.172
MEAN	0.505	0.357	0.089

Lastly, the distributions of the code coverage results considering every sequences for all the *model* genomes are displayed through boxplots in Figure 25. For space reasons, only the results for the overall best (*173*) and worst (*192*) code are presented. In **Appendix B** it is possible to find the distributions of the code coverage results considering all the 27 best and worst code groups for the six *model genomes* (Figure 32).

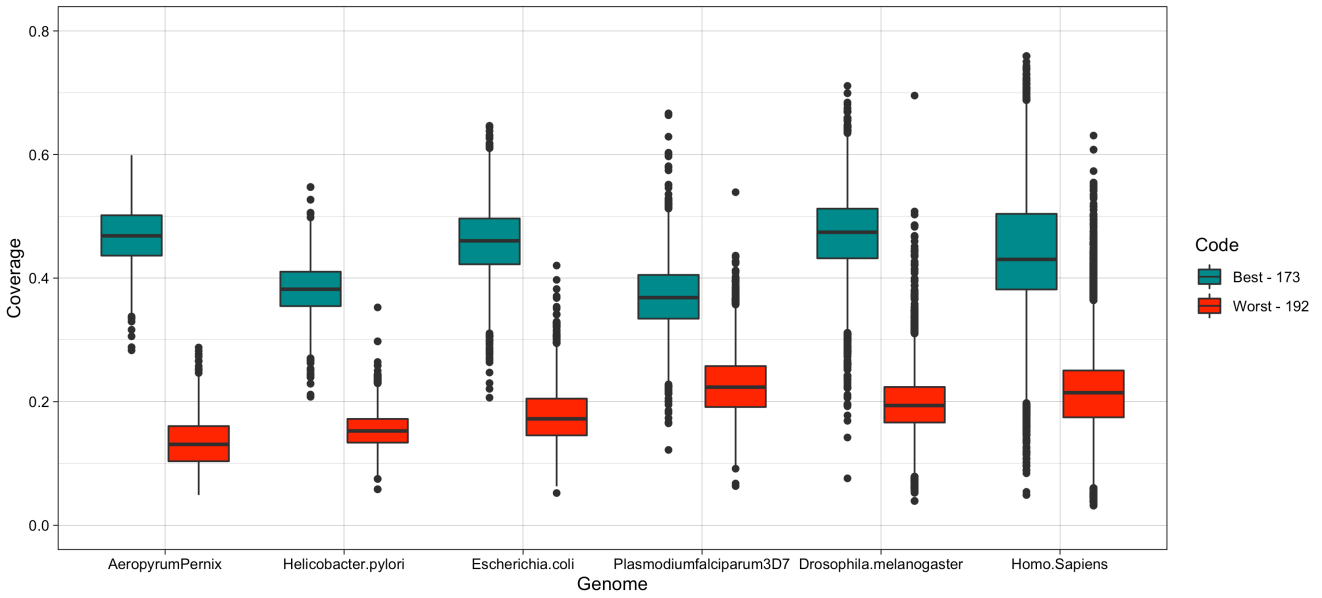


Figure 25: Distributions of coverages of best (*173*, in blue) and worst (*192*, in red) codes computed considering every sequence - *model* genomes.

Studying the distributions in Figures 25 and 32, it is clear that the coverage of the best codes is systematically and significantly higher than the one of the worst codons, for all the genomes under analysis. Looking at Figure 32, however, it can be seen that the difference between the best and worst codes coverages becomes gradually less marked as the *importance* of the pair of codes under investigation decreases (i.e. when scrolling down the first and last columns of Table 1). Nevertheless, this particular behaviour is due to the fact that different pairs of code groups correspond to different overall values for coverage. In fact, if we investigate the distributions of the weighted percentage differences between the best and worst codes (shown in Figures 33 and 34 in **Appendix B**) we can see that they are all very similar for the same genome regardless of which pair of groups is considered. The weighted percentage differences in Figure 33 and 34 were

calculated considering the difference

$$\frac{C_b - C_w}{C_b} \times 100$$

for every sequence in the *model* genomes.

The two figures present the same results but in a different order. Figure 33, in fact, presents 27 blocks of 6 boxplots, which correspond to the distributions of the differences in the *model* genomes considering the 27 best and worst code pairs. Figure 34, on the other hand, consists of 6 blocks of 27 boxplots, which correspond to the distributions of the 27 best and worst code pairs (sorted according to importance, i.e. by scrolling down from the top to the bottom of the first and last columns of Table 1) grouped by *model* genome.

3.5 Sequence length effect

The above findings, especially those obtained considering the *rolling means* approach, lead us to think that there might be a relationship between the code coverage and the length of the sequence. Considering the code coverages of all the individual sequences, it is easy to calculate the correlation between these quantities for the different codes and the sequence lengths. Table 16 shows the *Spearman's rank correlation coefficients* between the coverage for the three best and worst groups calculated on the individual sequences with the length of the latter, with reference to the *model* genomes. Although these values tend to be low (often very close to zero), it is interesting to note that very similar results are obtained when considering the best and worst groups for the same genome. This might suggest that there is indeed a relationship between the *sequence length effect* and the circular code groups under analysis. In particular, among the organisms taken as a model, *Plasmodiumfalciparum3D7* shows more extreme results (positive and negative correlations of around 0.39 in absolute value). Figures 26 and 27 shows a graphic representation of the length effect, based on the results obtained for this genome. The curve across the points in Figure 26 was calculated using the *LOESS* (locally estimated scatterplot smoothing) method, a local regression that combines the simplicity of linear least squares with the flexibility of non-linear regression. In Figure 27, on the other hand, since the relationship between the variables seems decidedly less complex, the function through the points of the scatterplot has been calculated using a simple linear regression.

As can be seen from Figure 26, the relationship between sequence length and code coverage is asymmetrical and non-linear. This is the reason why the correlations in Table 16 have been calculated considering *Spearman's rank correlation coefficient*, which is useful to study monotonic relationships (whether linear or not).

Table 16: Spearman's rank correlation cofficients between code coverage on every sequence of the first three pairs of best and worst codes and length of the sequence - *model* genomes.

	Best codes			Worst codes		
	173	23	98	192	87	53
AeropyrumPernix	0.051	0.090	0.075	0.009	0.012	0.056
Helicobacter.pylori	0.090	0.108	0.063	-0.057	-0.060	-0.051
Escherichia.coli	0.154	0.166	0.145	-0.144	-0.107	-0.088
Plasmodiumfalciparum3D7	0.385	0.386	0.391	-0.352	-0.391	-0.311
Drosophila.melanogaster	-0.150	-0.148	-0.133	0.193	0.210	0.205
Homo.Sapiens	0.077	0.101	0.073	-0.084	-0.064	-0.092

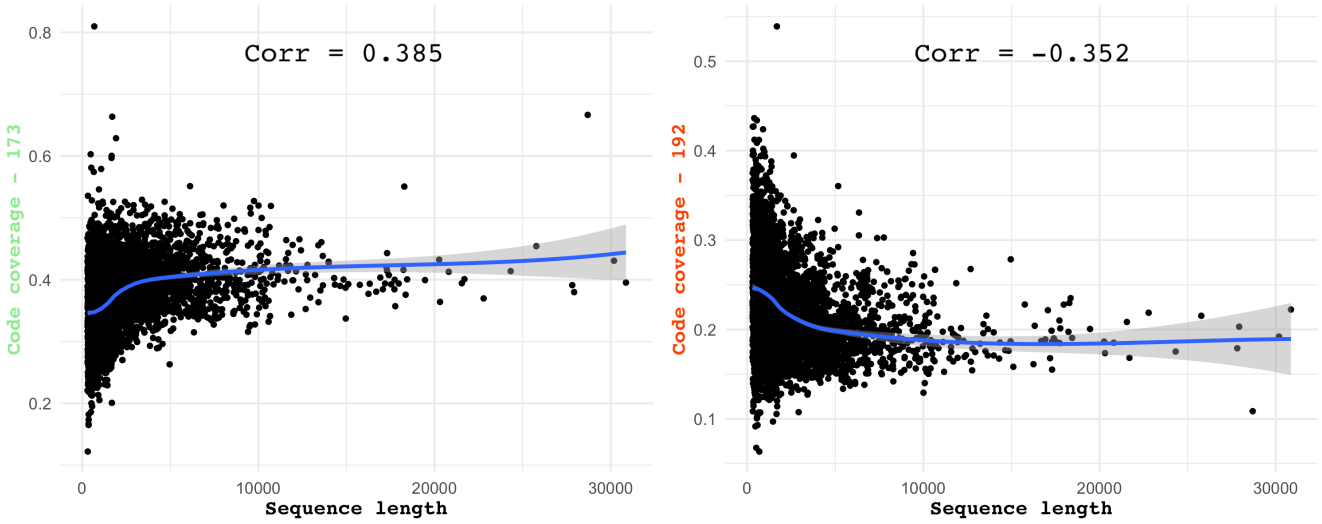


Figure 26: Relationship between coverage on every sequence of best (*173*) and worst (*192*) codes and length of the sequence. In blue the *LOESS* regression curve. - *Plasmodiumfalciparum3D7*

Looking at the scatterplots in Figure 26, a complex asymmetric dependency can be observed. The results for code coverage, in fact, tend to stabilise in a higher (for *173*) and lower (for *192*) range as the sequence length increases. Figure 27 shows scatterplots of the coverage of codons *173* and *192* using the base 10 logarithm of the sequence length for the *Plasmodiumfalciparum3D7*. From the graphs in the figure, the positive and negative effect of sequence length can be seen more clearly.

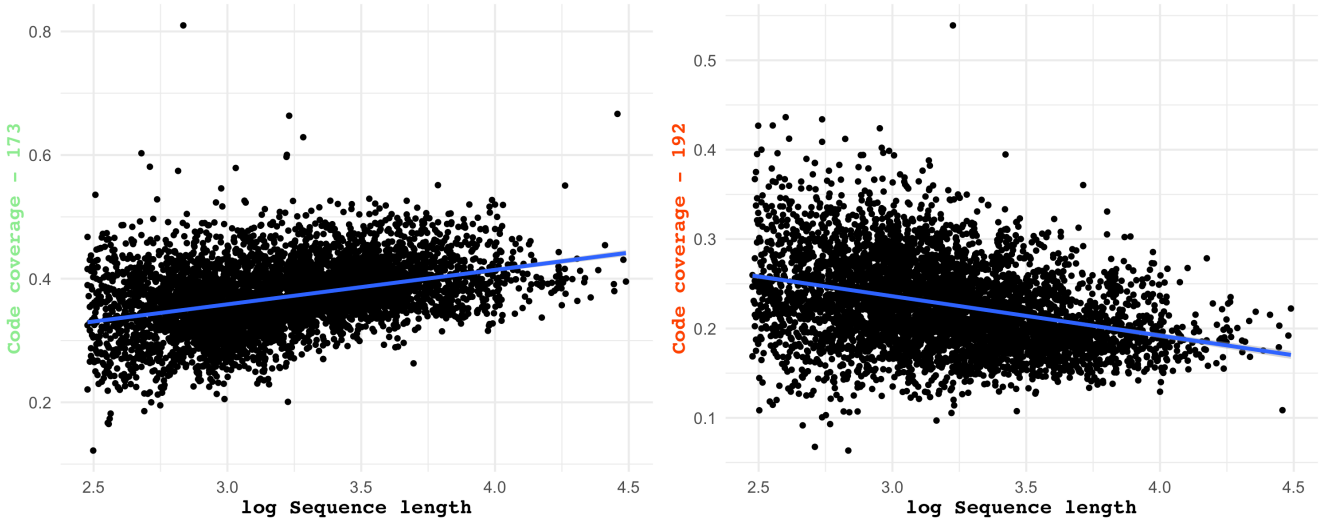


Figure 27: Relationship between coverage on every sequence of best (*173*) and worst (*192*) codes and logarithm of sequence length. In blue the simple linear regression curve. - *Plasmodiumfalciparum3D7*

Figures 42 and 43 in **Appendix B** show the scatterplots considering the actual sequence length and the sequence length to which the base-10 logarithm transformation was applied for all *model* genomes. It can be seen, therefore, that the above considerations also apply to all *model* genomes and that, consequently, it is possible to assume the existence of a *sequence length effect* as a universal property. In addition, the reasons for these particular results can be extracted from biological theory. In fact, it can be assumed that longer sequences have more need to be optimised, i.e. to contain codons whose synthesis is faster (present in the *best* code groups), in order to avoid slowing down during protein synthesis.

Thus, it would not be inaccurate to assume that codon usage and, consequently, code coverage values differ when considering *long* and *short* sequences (in terms of codon number). This suggests an internal separation of the individual sequences available for each genome, taking 1000 as the threshold value for length. Table 17 shows the different sizes of the two groups. Obviously, the sum of the sizes of the two subgroups corresponds to the total number of sequences in each genome, shown in the third column of Table 3.

Table 17: Number of sequences longer and shorter than 1000 codons - *model* genomes.

	> 1000	< 1000	Sum
AeropyrumPernix	286	427	713
Helicobacter.pylori	1,040	1,352	2,392
Escherichia.coli	1,568	2,415	3,983
Plasmodiumfalciparum3D7	3,525	1,734	5,259
Drosophila.melanogaster	8,590	4,016	12,606
Homo.Sapiens	73,856	66,594	140,450

Table 18, therefore, shows the values for code usage for the best (*173*) and worst (*192*) code in general obtained considering all sequences, the only sequences longer than 1000 codons and the only sequences shorter than 1000 codons, plus the difference between the latter two values. Observing the table, it can be seen that the overall value is always within the range limited by the results obtained on the *long* and *short* sequences only. This is consistent with the fact that the global value can be considered effectively as a weighted mean of the other two values (weighing for the different numerosities of the subgroups of *long* and *short* sequences). Studying the values of the differences, moreover, it is possible to note that in many cases they are not negligible at all. It can be assumed, therefore, that the results for codon usage and code usage are somehow related to the length of the sequences being examined, thus, that an actual *sequence length effect* may exists.

Table 18: Comparison of code coverage results for best (*173*) and worst (*192*) codes considering the sequences split according to their length - *model* genomes.

	Best code - 173				*Worst code - 192*			
	Global	> 1000	< 1000	Δ	Global	> 1000	< 1000	Δ
AeropyrumPernix	46.69	46.64	46.77	-0.13	13.69	13.97	13.27	0.70
Helicobacter.pylori	38.49	38.87	37.72	1.15	15.34	15.15	15.72	-0.57
Escherichia.coli	46.34	46.88	45.39	1.48	17.36	16.93	18.14	-1.21
Plasmodiumfalciparum3D7	38.92	39.30	34.83	4.47	21.19	20.88	24.44	-3.56
Drosophila.melanogaster	46.10	45.93	47.89	-1.97	20.74	20.91	19.01	1.90
Homo.Sapiens	44.17	44.18	44.15	0.03	21.32	21.25	21.56	-0.31

This separation also allows a new interpretation of the results obtained with the *rolling means* approach. Figures 28 and 29, in fact, are equivalent to those in the previous Figures 15 and 16,

with the difference that in this case the two red and blue lines now refer to the results obtained by considering *long* and *short* sequences respectively.

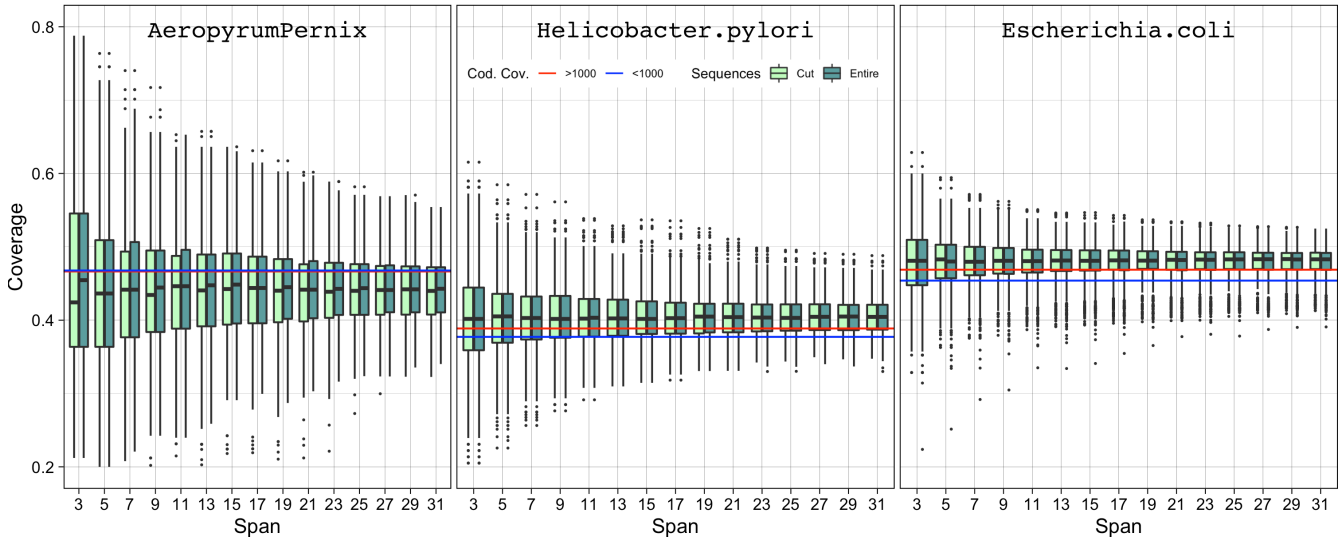


Figure 28: Coverages of the best code (*173*) calculated using the rolling means approach: cut (light green box) vs entire (dark green box) sequences. The benchmarks this time are the values for global code coverage considering the group of sequences longer than 1000 codons (in red) and the group of sequences shorter than 1000 codons (in blue) - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli*.

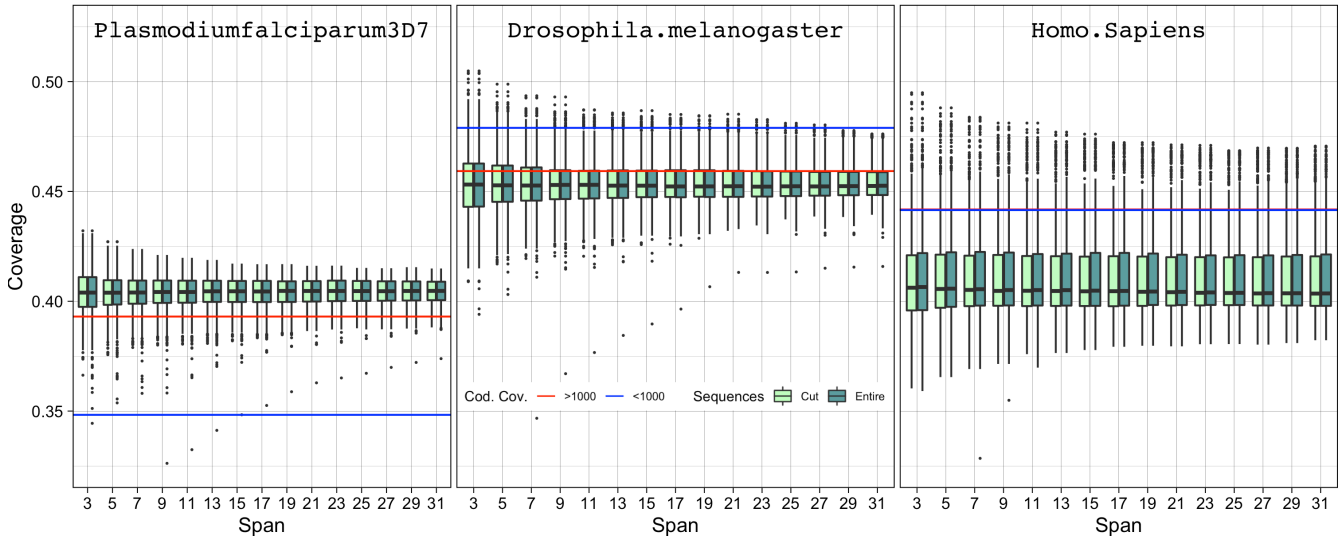


Figure 29: Coverages of the best code (*173*) calculated using the rolling means approach: cut (light green box) vs entire (dark green box) sequences. The benchmarks this time are the values for global code coverage considering the group of sequences longer than 1000 codons (in red) and the group of sequences shorter than 1000 codons (in blue) - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens*.

It is clear from the graphs that the results calculated on the *long* sequences only (red line) are systematically and often significantly closer to the values on which the distributions tend to stabilise.

This, therefore, leads us to understand that the distance between the results obtained with this approach and those on the whole genome (taken as a reference previously) is actually due to the fact that in the *rolling means* approach we select only the sequences with at least 1000 codons, as hypothesised in the relative paragraph. It is also legitimate to think that the distance that is still recorded between the values on which the distributions tend to stabilise and the code coverage on the *long* sequences only is due to the fact that in the *rolling means* approach we consider only the first 1000 codons of the *long* sequences, which introduces a further bias.

3.6 Transient effect in the first positions of the sequences

In this section the results for **code coverage considering the positional approach** are presented and discussed.

Figures 30 and 44 (in **Appendix B**) offer a visual summary of the results obtained by calculating the code coverage by position. In particular, the figures present the trend of the results for the best (*173*) and worst (*192*) code coverage. The limits on the x-axis are, therefore, *position 1* and the length of the longest sequence in the genome under consideration. The results for *position 0* have been deliberately removed, as the first codon in each sequence is always the start codon (ATG) which is not part of any code. It is useful to note that these analyses were carried out on entire sequences, so that it is possible to study whether there are particular patterns in the early part of the sequences.

Figure 44 shows the results for the whole range taken into consideration. In Figure 30, instead, the code coverage on the first 50 codons can be observed in detail and compared with the results for code coverage obtained considering the whole genome (without the initial and final part of the sequences).

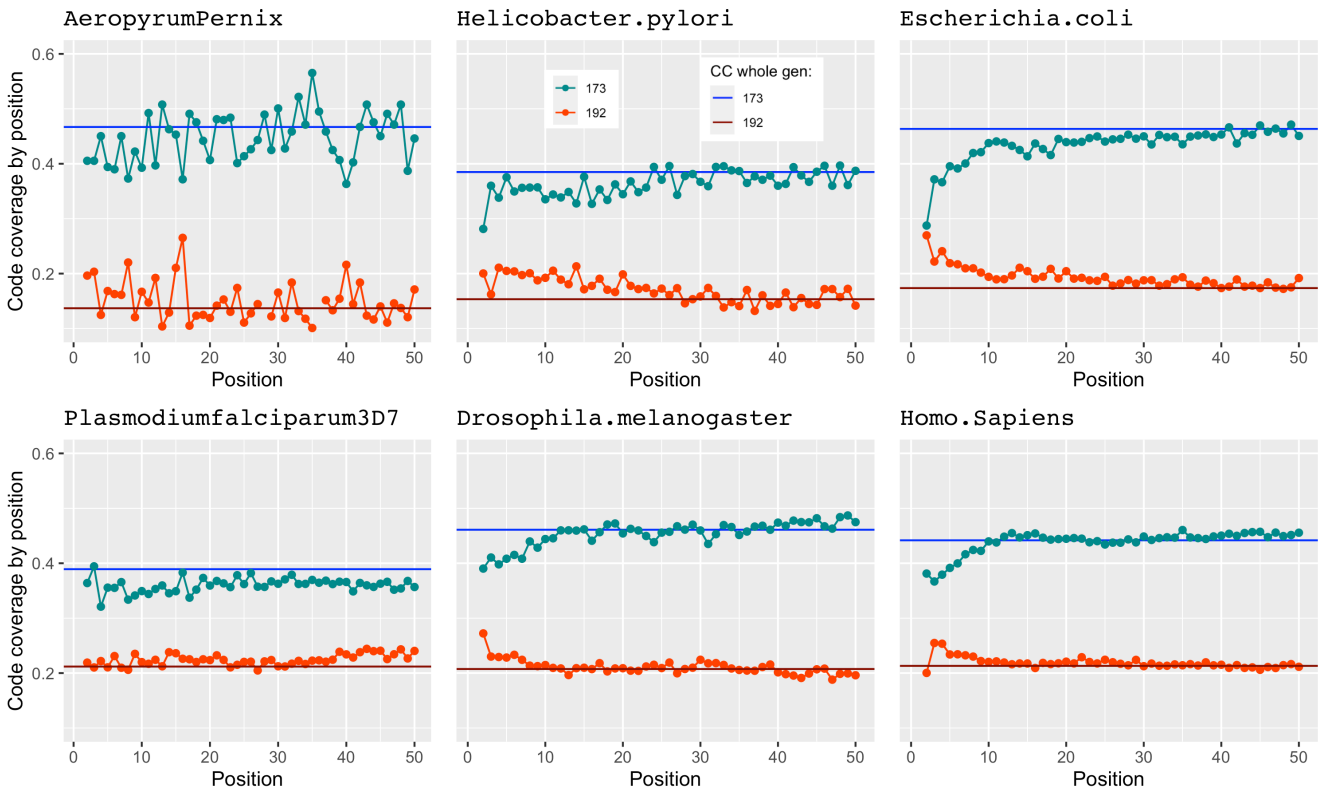


Figure 30: Coverages of the codes *173* (in green) and *192* (in red) by position, focus on the first 50 positions only. The coverage values obtained by considering the whole genome for code *173* (blue line) and *192* (dark red line) are also shown - *model* genomes.

Figures 37 to 41 (in **Appendix B**), in addition, show the results for the coverage of all 27 *best* and *worst* groups for the first 50 codons for five *model* genomes.

Several general observations can be drawn:

- from the graphs over the whole range, it can be seen that the results become extremely variable as the position taken into consideration increases. This particular behaviour is due to the fact that as the position increases, the amount of sequences in the genome with at least that number of codons decreases. This means that there are fewer observations on which to calculate the average, hence more variable results. Towards the last positions, in fact, the results are only 0 or 1, as they are calculated on a single sequence (the longest in the genome).
- Although when the variability increases the results tend to overlap, it can be observed that the coverage of code *173* tends to be higher than that of code *192*.

- Looking at the trend in the top 50 positions and the overall value of the whole genome, an effect is quite clear. In fact, it can be observed that the results tend to stabilise on the benchmark value only after the first positions, both for the best and the worst code. In particular, at the beginning of the sequences the coverage of *173* tends to be lower than the global value, while that of *192* is higher. This particular behaviour confirms the conclusions reached by analysing the results in the previous paragraphs: in the first positions of the sequences the codons that are part of the best and worst code are respectively less and more frequent than they are in the whole sequences.

- Results obtained on larger genomes, i.e. with more sequences, tend to be generally less variable. The coverage on the first 50 codons of *AeropyrumPernix*, for example, is much less stable than that of *Homo.Sapiens*. This is due to the fact that we average over larger sequence samples.

- In general, the values on which the results tend to stabilise are very close to the results obtained considering the whole genome. This suggests a consistency between the results for code coverage obtained by the different approaches under analysis.

3.7 Evidence of a non-random relationship between the coverage of *best* and *worst* codes

Figure 31 contains the results of the bootstrap test previously presented for all 16 *best* and *worst* disjoint code pairs and all 24 organisms under analysis.

It is useful to recall the hypotheses of the test:

$$\begin{cases} H_0 : C_w \text{ compatible with } C_{RAN} & \Longrightarrow \text{ relationship due to chance} \\ H_1 : C_w \text{ not compatible with } C_{RAN} & \Longrightarrow \text{ relationship not due to chance} \end{cases}$$

where C_w is the coverage of the *worst* code and C_{RAN} is the random variable representing the coverage of a random set of 20 codons taken from the subset of 44 codons complementary to the ones in the *best* code. In addition, it is worth mentioning for this test we derive the bootstrap rejection bands at a significance level 0.0001 by generating 10,000 bootstrap resamples.

Figure 31 shows 16 different graphs for all the best and worst disjoint code pairs. For each graph, it is possible to observe the *non-rejection region* (in green), that is the zone in which the code coverage would be considered compatible with that of the simulated random sets. The limits of this area correspond to the quantiles 0.0001 and 0.9999, output of the `codtest` function. If the coverage of a code lies within the *non-rejection region*, then it is not possible to reject the null hypothesis H_0 that the relation linking it to the *best* code is simply due to chance. In each graph it is possible to observe the results of the coverage of the *worst* code (in red) and of the *remainder* code (in blue) obtained considering the whole genome (in particular, considering the cut sequences, taking into account the fact that the differences with the results considering the entire sequences are almost zero). The test was performed on all the genomes under analysis, present on the *x-axis*.

Looking at the graphs in Figure 31, we can conclude that almost all the coverages of the *worst* codes lie below the *non-rejection zone*. It is possible, therefore, to reject with a 99.99% confidence in almost all cases the hypothesis that the fact of the marked difference between the coverage of the *best* and *worst* codes is simply due to chance.

Furthermore, it can be observed that the coverage for the *remainder* code always lies in the *non-rejection region*. When considering the *remainder* codes, therefore, we do not reject the null

hypothesis that the relationship with the *best* code is random. This puts even more emphasis on the previous result. It can, therefore, be concluded that the codes that correspond to the *Keto-Amino* transformation of the *best* codes are highly selected to be, indeed, the *worst* codes.

This conclusive result, together with the various consistent observations made in the previous paragraphs, provides significant evidence that there is a clear connection between the theory of circular codes and the translation process.

Figure 31: Bootstrap test results for the 16 disjoint pairs of *best* and *worst* codes and all the 24 genomes under analysis, with 10000 bootstrap replications and $\alpha = 0.0001$. In every plot are displayed the bootstrap rejection bands under the null hypothesis at $\alpha = 0.0001$ that the relation is produced by chance (in green) and the results of the coverage of the *worst* (in red) and the *remainder* (in blue) codes.

4 Conclusion

The results presented and described in the previous chapter provide veracity to the observations from which this study was based. With regard to the different behaviour at the ends of the sequences, in fact, all the results lead to the conclusion that the codons in the 27 *best* codes tend to be less present at the beginning and at the end of the sequences, in favour of those that are contained in the the *worst* codes. This conclusion, in fact, can be drawn both by considering the different values of codon usage and code coverage calculated on the whole genome taking into account the whole and cut sequences, and by observing the distribution of the code coverage by position in the first positions. These results, therefore, suggest the existence of a transient effect in the first 10-15 positions, implying lower values for the *best* codes coverage and higher values for the *worst* codes coverage compared to those recorded in the elongation phase. This observation is consistent with previous studies (Boël et al. 2016) and also allows a biological interpretation. It is reasonable to assume that the role played by circular codes affects the central parts of the sequences, i.e. the *elongation* phase. This is consistent with the fact that there are other predominant factors acting at the beginning of the sequence and connected to the *initiation* phase, so that circular codes and optimization for speed are not needed.

Also when considering the relationship linking the *best* codes to those on which the *Keto-Amino* transformation is applied (the *worst* codes), the results lead to an explicit conclusion in line with previous studies. In fact, regardless of the different calculation approach adopted, the code coverage of the *best* codes is always significantly higher than that of the *worst* codes. In particular, it was also seen that the coverage of the *best* and *worst* groups are correlated: as one increases, the other tends to decrease. The bootstrap test proved that this particular relationship is not due to chance, but implies a systematic codon separation. It is reasonable to conclude, therefore, that there is a connection between the theory of circular codes and the dynamics underlying the translation process. This relationship leads to the identification of two classes of codes, *best* and *worst*, that are optimised to contain codons that ensure a respectively higher and lower efficiency of the entire translation process. It is useful to recall that these observations are **universal** both in terms of genomes, i.e. they are common to all 24 genomes under analysis, and in terms of equivalence classes, i.e. they are valid for all 27 best and worst code pairs in each class.

Finally, the code coverage calculated on the individual sequences made it possible to study and recognise a new result. In fact, it was seen that there is a correlation between the length of the sequences under examination and the code coverage of the *best* and *worst* groups. In particular, it was observed that beyond a certain threshold for the sequence length, the results tend to stabilise on higher values for the *best* codes and lower values for the *worst* codes. It is therefore possible to define a *sequence length effect*. A biological interpretation is also possible here. It can be assumed that longer sequences, the translation of which is longer and more likely to be problematic by construction, have a greater need to be composed of more efficient codons, i.e. those contained in the *best* codes. In contrast, shorter sequences do not have this particular requirement. This interesting result deserves further investigation and could lay the foundations for future studies related to this topic.

5 Bibliography

Arquès, D. G. & Michel, C. J. (1996), 'A complementary circular code in the protein coding genes', *J. Theor. Biol.* **182**, 45–58.

Boël, G., Letso, R., Neely, H., Price, W., Wong, K.-H., Su, M., Luff, J., Valecha, M., Everett, J., Acton, T., Xiao, R., Montelione, G., Aalberts, D. & Hunt, J. (2016), 'Codon influence on protein expression in E. coli correlates with mRNA levels', *Nature* **529**, 358 – 376.

Charif, D. & Lobry, J. (2007), SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis., *in* U. Bastolla, M. Porto, H. Roman & M. Vendruscolo, eds, 'Structural approaches to sequence evolution: Molecules, networks, populations', Biological and Medical Physics, Biomedical Engineering, Springer Verlag, New York, pp. 207–232. ISBN : 978-3-540-35305-8.

Crick, F., Griffith, J. & Orgel, L. (1957), 'Codes without commas', *Proc. Nat. Acad. Sci. U. S. A.* **43**, 416–421.

Demongeot, J. & Seligmann, H. (2020), 'Pentamers with non-redundant frames: Bias for natural circular code codons', *Journal of Molecular Evolution* **88**(2), 194–201.
URL: *https://doi.org/10.1007/s00239-019-09925-0*

Dila, G., Ripp, R., Mayer, C., Poch, O., Michel, C. J. & Thompson, J. D. (2019), 'Circular code motifs in the ribosome: a missing link in the evolution of translation?', *RNA* .

Dowle, M. & Srinivasan, A. (2020), *data.table: Extension of 'data.frame'*. R package version 1.13.4.
URL: *https://CRAN.R-project.org/package=data.table*

Eddelbuettel, D. & François, R. (2011), 'Rcpp: Seamless R and C++ integration', *Journal of Statistical Software* **40**(8), 1–18.
URL: *https://www.jstatsoft.org/v40/i08/*

Fayazi, F., Fimmel, E. & Strüngmann, L. (2021), 'Equivalence classes of circular codes induced by permutation groups', *Theory in Biosciences* **140**.

Fimmel, E., Giannerini, S., Gonzalez, D. L. & Strüngmann, L. (2015), 'Circular codes, symmetries and transformations', *Journal of Mathematical Biology* **70**(7), 1623–1644.

Fimmel, E., Michel, C., Pirot, f., Sereni, J.-S., Starman, M. & Strüngmann, L. (2020), 'The relation between k-circularity and circularity of codes', *Bulletin of Mathematical Biology* **82**.

Fimmel, E., Michel, C., Starman, M. & Strüngmann, L. (2018), 'Self-complementary circular codes in coding theory', *Theory in Biosciences* **137**.

Gamow, G. (1954), 'Possible relation between deoxyribonucleic acid and protein structures', *Nature* **173**(4398), 318.

Giannerini, S., González, D., Goracci, G. & Danielli, A. (2021), 'A role for circular code properties in translation', *Scientific Reports* **11**(9218).
URL: *https://doi.org/10.1038/s41598-021-87534-y*

Giannerini, S., Gonzalez, D. & Rosa, R. (2012), 'DNA, frame synchronization and dichotomic classes: a quasicrystal framework', *Phil. Trans. R. Soc. A* **370**(1969), 2987–3006.

Giennerini, S. & Dalena, P. (2021), *mathDNA: DNA sequence analysis motivated by the non-power model of the genetic code and related themes*. R package version 0.4.0.

Gonzalez, D., Giannerini, S. & Rosa, R. (2011), 'Circular codes revisited: A statistical approach', *Journal of Theoretical Biology* **275**(1), 21–28.

Gonzalez, D. L., Giannerini, S. & Rosa, R. (2008), 'Strong short-range correlations and dichotomic codon classes in coding DNA sequences', *Physical Review E* **78**(5), 051918.

Gonzalez, D. L., Giannerini, S. & Rosa, R. (2009), 'The mathematical structure of the genetic code: a tool for inquiring on the origin of life.', *Statistica* **LXIX**(3–4), 143–157.

Koch, A. J. & Lehman, J. (1997), 'About a symmetry of the genetic code', *J. Theor. Biol.* **189**, 171–174.

Lacan, J. & Michel, C. J. (2001), 'Analysis of a circular code model', *J. Theor. Biol.* **213**, 159–170.

Michel, C. (2020), 'The maximality of circular codes in genes statistically verified', *Biosystems* **197**, 104201.

Michel, C. J. (2015), 'The maximal C^3 self-complementary trinucleotide circular code X in genes of bacteria, eukaryotes, plasmids and viruses', *Journal of Theoretical Biology* **380**, 156 – 177.

Michel, C. J. & Seligmann, H. (2014), 'Bijective transformation circular codes and nucleotide exchanging RNA transcription', *Biosystems* **118**, 39–50.
URL: *https://doi.org/10.1016/j.biosystems.2014.02.002*

Michel, C., Pirillo, G. & Pirillo, M. (2008), 'A relation between trinucleotide comma-free codes and trinucleotide circular codes', *Theoretical Computer Science* **401**(1-3), 17 – 26.

NCBI Resource Coordinators (2016), 'Database resources of the National Center for Biotechnology Information', *Nucleic acids research* **44**(D1), D7–D19. Edition: 2015/11/28 Publisher: Oxford University Press.
URL: *https://pubmed.ncbi.nlm.nih.gov/26615191*

Nirenberg, M. & Matthaei, J. (1961), 'The dependence of cell-free protein synthesis in E. coli upon naturally occurring or synthetic polyribonucleotides', *Proceedings of the National Academy of Sciences* **47**(10), 1588–1602.

Seligmann, H. (2016), 'Swinger RNA self-hybridization and mitochondrial non-canonical swinger transcription, transcription systematically exchanging nucleotides', *Journal of Theoretical Biology* **399**, 84–91.
URL: *https://doi.org/10.1016/j.jtbi.2016.04.007*

Starman, M. (2018), *GCATR: Genetic Code Analysis Toolkit (GCAT) in R -> GCATR*. R package version 0.1.

Watson, J. D. & Crick, F. H. C. (1953), 'Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid', *Nature* **171**(4356), 737–738.
URL: *https://doi.org/10.1038/171737a0*

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
URL: *https://ggplot2.tidyverse.org*

6 Appendix

This chapter is divided into 3 sub-chapters: Appendix A contains the additional tables, Appendix B the additional figures and Appendix C some of the code used for the analyses.

6.1 Appendix A: additional tables

For reasons of space, the captions of the tables are not as informative as those in the previous chapters. For a better understanding, please refer to the part of the Results where the values in the tables are discussed.

In Tables 22, 23 and 24, organisms are ordered as in the first column of Table 3. The names of the genomes have been replaced by letters for space reasons. Columns with bold values refer to *model* genomes.

Table 19: Different values in codon usage (x100) between entire and cut sequences - 'model' genomes

	A.Pernix			*H.pylori*			*E.coli*			*Plasmodium*			*D.melanogaster*			*H.Sapiens*		
	E	C	Δ	E	C	Δ	E	C	Δ	E	C	Δ	E	C	Δ	E	C	Δ
AAA	0.86	0.83	0.03	6.68	6.57	0.11	3.41	3.32	0.09	9.55	9.57	-0.02	1.75	1.72	0.03	1.48	1.49	-0.01
AAC	1.76	1.82	-0.06	2.54	2.59	-0.05	2.18	2.20	-0.02	2.00	2.00	0.00	2.63	2.64	-0.01	1.99	2.01	-0.02
AAG	3.72	3.75	-0.03	2.05	2.02	0.03	1.10	1.05	0.05	2.15	2.13	0.02	3.88	3.88	0.00	3.21	3.23	-0.02
AAT	0.41	0.42	-0.01	3.27	3.29	-0.02	1.90	1.89	0.01	12.35	12.51	-0.16	2.18	2.18	0.00	1.48	1.50	-0.02
ACA	1.05	1.02	0.03	0.66	0.65	0.01	0.80	0.76	0.04	2.17	2.18	-0.01	1.22	1.22	0.00	1.12	1.13	-0.01
ACC	1.44	1.49	-0.05	1.43	1.46	-0.03	2.28	2.33	-0.05	0.48	0.48	0.00	2.13	2.15	-0.02	1.55	1.55	0.00
ACG	1.13	1.17	-0.04	1.01	1.03	-0.02	1.49	1.51	-0.02	0.38	0.38	0.00	1.40	1.40	0.00	1.20	1.19	0.01
ACT	0.87	0.89	-0.02	1.36	1.38	-0.02	0.90	0.89	0.01	1.06	1.06	0.00	1.11	1.12	-0.01	1.02	1.03	-0.01
AGA	1.04	1.01	0.03	0.86	0.84	0.02	0.28	0.25	0.03	1.60	1.59	0.01	0.55	0.54	0.01	0.97	0.96	0.01
AGC	2.50	2.55	-0.05	2.74	2.80	-0.06	1.59	1.61	-0.02	0.39	0.39	0.00	2.00	2.00	0.00	1.66	1.65	0.01
AGG	4.99	4.97	0.02	0.85	0.85	0.00	0.18	0.16	0.02	0.43	0.43	0.00	0.61	0.61	0.00	1.57	1.56	0.01
AGT	0.54	0.51	0.03	0.98	0.97	0.01	0.93	0.92	0.01	2.04	2.05	-0.01	1.19	1.19	0.00	0.84	0.84	0.00
ATA	4.14	4.19	-0.05	0.87	0.85	0.02	0.54	0.51	0.03	5.02	5.03	-0.01	0.99	0.98	0.01	0.88	0.90	-0.02
ATC	1.04	1.05	-0.01	2.82	2.86	-0.04	2.39	2.42	-0.03	0.63	0.63	0.00	2.17	2.19	-0.02	2.09	2.12	-0.03
ATG	2.41	2.15	0.26	2.29	2.05	0.24	2.73	2.47	0.26	2.19	2.07	0.12	2.16	2.03	0.13	2.42	2.20	0.22
ATT	0.86	0.85	0.01	3.52	3.51	0.01	2.95	2.94	0.01	3.60	3.61	-0.01	1.74	1.74	0.00	1.42	1.45	-0.03
CAA	0.23	0.22	0.01	3.10	3.11	-0.01	1.45	1.42	0.03	2.39	2.40	-0.01	1.72	1.73	-0.01	1.23	1.24	-0.01
CAC	1.14	1.18	-0.04	0.66	0.67	-0.01	0.94	0.94	0.00	0.35	0.34	0.01	1.53	1.54	-0.01	1.38	1.39	-0.01
CAG	1.68	1.67	0.01	0.56	0.57	-0.01	2.97	3.01	-0.04	0.37	0.37	0.00	3.58	3.61	-0.03	2.10	2.12	-0.02
CAT	0.43	0.43	0.00	1.45	1.46	-0.01	1.28	1.28	0.00	2.07	2.09	-0.02	1.06	1.07	-0.01	1.06	1.07	-0.01
CCA	0.74	0.71	0.03	0.49	0.48	0.01	0.83	0.83	0.00	0.90	0.91	-0.01	1.52	1.53	-0.01	1.35	1.35	0.00
CCC	1.96	1.96	0.00	0.88	0.90	-0.02	0.55	0.54	0.01	0.21	0.20	0.01	1.87	1.89	-0.02	1.18	1.17	0.01
CCG	1.21	1.22	-0.01	0.34	0.35	-0.01	2.28	2.34	-0.06	0.10	0.09	0.01	1.55	1.56	-0.01	1.83	1.81	0.02
CCT	1.06	1.04	0.02	1.63	1.67	-0.04	0.72	0.72	0.00	0.78	0.79	-0.01	0.83	0.84	-0.01	1.27	1.28	-0.01
CGA	0.11	0.10	0.01	0.25	0.24	0.01	0.38	0.36	0.02	0.24	0.24	0.00	0.86	0.86	0.00	0.55	0.55	0.00
CGC	0.34	0.34	0.00	0.84	0.85	-0.01	2.09	2.12	-0.03	0.04	0.04	0.00	1.73	1.73	0.00	1.59	1.59	0.00
CGG	0.41	0.40	0.01	0.10	0.11	-0.01	0.62	0.62	0.00	0.03	0.03	0.00	0.76	0.76	0.00	1.28	1.27	0.01
CGT	0.26	0.26	0.00	0.48	0.47	0.01	2.04	2.06	-0.02	0.30	0.30	0.00	0.90	0.90	0.00	0.64	0.65	-0.01
CTA	2.01	2.02	-0.01	0.80	0.79	0.01	0.38	0.38	0.00	0.61	0.60	0.01	0.81	0.81	0.00	0.74	0.75	-0.01
CTC	3.38	3.38	0.00	1.00	1.00	0.00	1.04	1.04	0.00	0.18	0.17	0.01	1.28	1.28	0.00	2.79	2.79	0.00
CTG	2.82	2.82	0.00	0.43	0.44	-0.01	5.15	5.24	-0.09	0.15	0.14	0.01	3.53	3.54	-0.01	2.24	2.25	-0.01
CTT	1.61	1.62	-0.01	1.60	1.59	0.01	1.13	1.11	0.02	0.87	0.86	0.01	0.92	0.92	0.00	1.52	1.54	-0.02
GAA	1.12	1.10	0.02	5.07	5.10	-0.03	3.95	4.00	-0.05	6.09	6.14	-0.05	2.41	2.42	-0.01	1.99	2.02	-0.03
GAC	2.95	3.02	-0.07	1.36	1.39	-0.03	1.94	1.98	-0.04	0.87	0.87	0.00	2.40	2.41	-0.01	2.92	2.96	-0.04
GAG	6.15	6.19	-0.04	1.81	1.83	-0.02	1.89	1.90	-0.01	1.03	1.03	0.00	4.21	4.25	-0.04	3.81	3.83	-0.02
GAT	1.43	1.43	0.00	3.41	3.47	-0.06	3.29	3.36	-0.07	5.59	5.65	-0.06	2.80	2.81	-0.01	2.41	2.45	-0.04
GCA	1.50	1.53	-0.03	0.68	0.67	0.01	2.08	2.07	0.01	0.84	0.83	0.01	1.30	1.29	0.01	1.67	1.67	0.00
GCC	3.62	3.71	-0.09	1.47	1.51	-0.04	2.54	2.60	-0.06	0.21	0.21	0.00	3.18	3.20	-0.02	3.17	3.14	0.03
GCG	1.97	2.01	-0.04	2.11	2.16	-0.05	3.24	3.31	-0.07	0.11	0.11	0.00	1.29	1.28	0.01	2.77	2.69	0.08
GCT	2.38	2.40	-0.02	2.67	2.71	-0.04	1.53	1.54	-0.01	0.82	0.82	0.00	1.49	1.49	0.00	1.87	1.87	0.00
GGA	1.29	1.27	0.02	0.58	0.58	0.00	0.88	0.88	0.00	1.24	1.25	-0.01	1.85	1.87	-0.02	1.51	1.52	-0.01
GGC	3.40	3.41	-0.01	2.12	2.19	-0.07	2.82	2.90	-0.08	0.13	0.13	0.00	2.47	2.48	-0.01	3.09	3.10	-0.01
GGG	2.37	2.37	0.00	2.23	2.29	-0.06	1.18	1.20	-0.02	0.28	0.28	0.00	0.45	0.44	0.01	1.72	1.72	0.00
GGT	1.63	1.56	0.07	0.99	1.00	-0.01	2.45	2.52	-0.07	1.18	1.19	-0.01	1.33	1.35	-0.02	1.45	1.46	-0.01
GTA	1.50	1.50	0.00	0.60	0.59	0.01	1.09	1.08	0.01	1.56	1.57	-0.01	0.70	0.70	0.00	0.65	0.66	-0.01
GTC	2.33	2.37	-0.04	0.81	0.83	-0.02	1.47	1.49	-0.02	0.24	0.24	0.00	1.35	1.36	-0.01	2.10	2.12	-0.02
GTG	3.13	3.18	-0.05	2.80	2.87	-0.07	2.58	2.65	-0.07	0.48	0.48	0.00	2.66	2.68	-0.02	2.56	2.57	-0.01
GTT	2.44	2.42	0.02	1.52	1.50	0.02	1.80	1.80	0.00	1.52	1.52	0.00	1.24	1.25	-0.01	1.52	1.55	-0.03
TAA	0.06	0.00	0.06	0.16	0.00	0.16	0.18	0.00	0.18	0.10	0.01	0.09	0.06	0.00	0.06	0.06	0.00	0.06
TAC	2.46	2.55	-0.09	1.11	1.13	-0.02	1.24	1.26	-0.02	0.62	0.61	0.01	1.77	1.78	-0.01	1.65	1.69	-0.04
TAG	0.19	0.00	0.19	0.04	0.00	0.04	0.02	0.00	0.02	0.02	0.00	0.02	0.05	0.00	0.05	0.07	0.00	0.07
TAT	1.10	1.12	-0.02	2.47	2.48	-0.01	1.67	1.69	-0.02	5.07	5.10	-0.03	1.07	1.07	0.00	0.97	0.99	-0.02
TCA	0.51	0.51	0.00	0.58	0.57	0.01	0.82	0.80	0.02	1.66	1.66	0.00	0.89	0.89	0.00	1.21	1.22	-0.01
TCC	1.01	1.03	-0.02	0.57	0.57	0.00	0.90	0.90	0.00	0.51	0.51	0.00	1.95	1.96	-0.01	1.70	1.68	0.02
TCG	0.70	0.69	0.01	0.37	0.37	0.00	0.88	0.89	-0.01	0.30	0.30	0.00	1.64	1.64	0.00	1.28	1.26	0.02
TCT	0.64	0.63	0.01	1.56	1.56	0.00	0.87	0.87	0.00	1.47	1.48	-0.01	0.81	0.81	0.00	1.20	1.20	0.00
TGA	0.06	0.00	0.06	0.08	0.00	0.08	0.10	0.00	0.10	0.03	0.01	0.02	0.04	0.00	0.04	0.11	0.00	0.11
TGC	0.42	0.42	0.00	0.72	0.73	-0.01	0.62	0.63	-0.01	0.23	0.23	0.00	1.38	1.39	-0.01	1.28	1.28	0.00
TGG	1.23	1.20	0.03	0.68	0.67	0.01	1.53	1.54	-0.01	0.50	0.49	0.01	0.94	0.93	0.01	1.35	1.36	-0.01
TGT	0.14	0.13	0.01	0.35	0.34	0.01	0.51	0.51	0.00	1.54	1.55	-0.01	0.67	0.67	0.00	0.57	0.58	-0.01
TTA	0.41	0.41	0.00	4.29	4.26	0.03	1.35	1.31	0.04	4.72	4.73	-0.01	0.50	0.49	0.01	0.59	0.60	-0.01
TTC	2.55	2.62	-0.07	1.15	1.16	-0.01	1.60	1.62	-0.02	0.72	0.70	0.02	2.04	2.04	0.00	2.43	2.46	-0.03
TTG	0.59	0.57	0.02	3.00	3.01	-0.01	1.28	1.27	0.01	1.05	1.04	0.01	1.59	1.59	0.00	1.42	1.42	0.00
TTT	0.61	0.61	0.00	4.09	4.04	0.05	2.20	2.20	0.00	3.64	3.61	0.03	1.29	1.28	0.01	1.27	1.29	-0.02

Table 20: R-squared of quadratic regression - codes 23 and 87

	23-87	23-rem	87-rem
AeropyrumPernix	0.370	0.248	0.171
Thermoplasma.acidophilum	0.453	0.210	0.128
P.Horikoshii	0.583	0.096	0.250
Pyrococcus	0.362	0.267	0.140
Staphylococcus.aureus	0.264	0.409	0.116
Helicobacter.pylori	0.209	0.496	0.101
Methanosarcina	0.412	0.438	0.031
Archaeoglobus	0.335	0.362	0.127
Escherichia.coli	0.595	0.472	0.008
Streptomyces.coelicolorA3	0.598	0.650	0.057
M.Xanthus	0.506	0.495	0.001
Caenorhabditis.elegans	0.592	0.229	0.158
Sulfolobus.solfataricus	0.434	0.143	0.203
Schizosaccharomyces.Pombe	0.559	0.258	0.061
Plasmodiumfalciparum3D7	0.373	0.309	0.126
Leishmania.major	0.634	0.261	0.023
Drosophila.melanogaster	0.617	0.376	0.002
DanioRerio	0.561	0.331	0.014
ZeaMays	0.701	0.395	0.011
OryzaSativa	0.655	0.382	0.024
Bacillus.subtilis	0.375	0.385	0.060
MusMusculus	0.589	0.349	0.005
Homo.Sapiens	0.676	0.392	0.025
Arabidopsis.Thaliana	0.370	0.229	0.168
MEAN	0.493	0.341	0.084

Table 21: R-squared of quadratic regression - codes 98 and 53

	98-53	98-rem	53-rem
AeropyrumPernix	0.388	0.342	0.099
Thermoplasma.acidophilum	0.351	0.269	0.149
P.Horikoshii	0.560	0.080	0.351
Pyrococcus	0.366	0.226	0.174
Staphylococcus.aureus	0.357	0.227	0.177
Helicobacter.pylori	0.248	0.522	0.059
Methanosarcina	0.287	0.347	0.141
Archaeoglobus	0.288	0.297	0.177
Escherichia.coli	0.465	0.394	0.022
Streptomyces.coelicolorA3	0.499	0.569	0.009
M.Xanthus	0.503	0.522	0.014
Caenorhabditis.elegans	0.563	0.238	0.050
Sulfolobus.solfataricus	0.490	0.160	0.149
Schizosaccharomyces.Pombe	0.404	0.187	0.194
Plasmodiumfalciparum3D7	0.491	0.237	0.102
Leishmania.major	0.631	0.225	0.051
Drosophila.melanogaster	0.567	0.406	0.006
DanioRerio	0.496	0.336	0.031
ZeaMays	0.638	0.561	0.050
OryzaSativa	0.610	0.506	0.062
Bacillus.subtilis	0.340	0.422	0.060
MusMusculus	0.570	0.433	0.003
Homo.Sapiens	0.627	0.531	0.065
Arabidopsis.Thaliana	0.371	0.198	0.197
MEAN	0.463	0.343	0.100

Table 22: Difference in code coverages (x100) between entire and cut sequences (entire - cut), best 27 code groups - all genomes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
173	**-0.56**	-0.40	-0.44	-0.37	-0.32	**-0.46**	-0.50	-0.51	**-0.61**	-1.02	-0.74	-0.15	-0.30	-0.20	**-0.22**	-0.27	**-0.21**	-0.35	-0.48	-0.35	-0.58	-0.32	**-0.38**	-0.26
23	**-0.43**	-0.39	-0.47	-0.41	-0.43	**-0.41**	-0.53	-0.51	**-0.64**	-0.93	-0.58	-0.14	-0.33	-0.24	**-0.23**	-0.24	**-0.22**	-0.35	-0.46	-0.35	-0.58	-0.31	**-0.38**	-0.26
98	**-0.48**	-0.30	-0.31	-0.28	-0.29	**-0.45**	-0.40	-0.35	**-0.55**	-0.99	-0.71	-0.14	-0.24	-0.19	**-0.21**	-0.26	**-0.20**	-0.33	-0.44	-0.34	-0.51	-0.30	**-0.35**	-0.24
25	**-0.47**	-0.32	-0.38	-0.33	-0.32	**-0.46**	-0.46	-0.46	**-0.57**	-0.94	-0.67	-0.14	-0.27	-0.19	**-0.24**	-0.26	**-0.21**	-0.34	-0.45	-0.34	-0.55	-0.31	**-0.36**	-0.25
20	**-0.34**	-0.31	-0.41	-0.36	-0.43	**-0.41**	-0.49	-0.46	**-0.61**	-0.85	-0.52	-0.14	-0.29	-0.23	**-0.26**	-0.23	**-0.21**	-0.34	-0.43	-0.34	-0.55	-0.29	**-0.36**	-0.25
166	**-0.55**	-0.42	-0.34	-0.25	-0.23	**-0.39**	-0.50	-0.49	**-0.48**	-0.96	-0.67	-0.12	-0.25	-0.17	**-0.15**	-0.23	**-0.19**	-0.33	-0.43	-0.34	-0.52	-0.30	**-0.36**	-0.25
4	**-0.42**	-0.41	-0.37	-0.28	-0.34	**-0.34**	-0.52	-0.49	**-0.51**	-0.87	-0.51	-0.12	-0.27	-0.21	**-0.17**	-0.20	**-0.20**	-0.32	-0.41	-0.34	-0.52	-0.28	**-0.36**	-0.25
30	**-0.36**	-0.36	-0.37	-0.34	-0.17	**-0.22**	-0.39	-0.39	**-0.44**	-0.93	-0.58	-0.03	-0.13	-0.10	**0.02**	-0.21	**-0.15**	-0.23	-0.35	-0.25	-0.34	-0.21	**-0.28**	-0.14
117	**-0.46**	-0.34	-0.27	-0.20	-0.23	**-0.39**	-0.45	-0.44	**-0.44**	-0.88	-0.60	-0.12	-0.21	-0.17	**-0.18**	-0.22	**-0.19**	-0.32	-0.40	-0.33	-0.50	-0.28	**-0.34**	-0.23
111	**-0.33**	-0.33	-0.31	-0.24	-0.34	**-0.34**	-0.48	-0.44	**-0.47**	-0.79	-0.45	-0.12	-0.24	-0.21	**-0.19**	-0.19	**-0.20**	-0.31	-0.38	-0.33	-0.50	-0.26	**-0.34**	-0.23
22	**-0.39**	-0.39	-0.52	-0.49	-0.45	**-0.39**	-0.55	-0.46	**-0.63**	-0.75	-0.45	-0.19	-0.37	-0.23	**-0.24**	-0.20	**-0.20**	-0.32	-0.40	-0.31	-0.57	-0.29	**-0.34**	-0.26
172	**-0.63**	-0.38	-0.47	-0.43	-0.36	**-0.55**	-0.51	-0.61	**-0.49**	-0.78	-0.55	-0.14	-0.37	-0.20	**-0.22**	-0.23	**-0.18**	-0.28	-0.44	-0.32	-0.56	-0.30	**-0.35**	-0.25
21	**-0.50**	-0.37	-0.51	-0.46	-0.47	**-0.51**	-0.53	-0.61	**-0.53**	-0.69	-0.40	-0.14	-0.40	-0.24	**-0.23**	-0.21	**-0.18**	-0.28	-0.42	-0.32	-0.56	-0.28	**-0.35**	-0.25
24	**-0.53**	-0.30	-0.41	-0.38	-0.36	**-0.55**	-0.46	-0.56	**-0.46**	-0.70	-0.48	-0.14	-0.34	-0.20	**-0.24**	-0.22	**-0.18**	-0.27	-0.41	-0.31	-0.53	-0.28	**-0.33**	-0.23
97	**-0.55**	-0.27	-0.35	-0.33	-0.33	**-0.55**	-0.40	-0.45	**-0.43**	-0.75	-0.53	-0.13	-0.31	-0.20	**-0.22**	-0.22	**-0.17**	-0.26	-0.40	-0.31	-0.49	-0.28	**-0.31**	-0.23
171	**-0.41**	-0.29	-0.45	-0.41	-0.48	**-0.50**	-0.49	-0.56	**-0.49**	-0.61	-0.33	-0.14	-0.36	-0.24	**-0.26**	-0.20	**-0.18**	-0.27	-0.39	-0.31	-0.53	-0.26	**-0.33**	-0.23
3	**-0.49**	-0.38	-0.41	-0.34	-0.39	**-0.43**	-0.52	-0.59	**-0.40**	-0.63	-0.32	-0.12	-0.34	-0.22	**-0.17**	-0.17	**-0.16**	-0.25	-0.37	-0.31	-0.50	-0.25	**-0.33**	-0.23
165	**-0.62**	-0.39	-0.37	-0.30	-0.28	**-0.48**	-0.50	-0.59	**-0.36**	-0.72	-0.48	-0.12	-0.32	-0.18	**-0.16**	-0.19	**-0.16**	-0.26	-0.39	-0.31	-0.50	-0.27	**-0.33**	-0.23
26	**-0.58**	-0.37	-0.52	-0.51	-0.39	**-0.53**	-0.52	-0.56	**-0.48**	-0.60	-0.42	-0.19	-0.42	-0.20	**-0.23**	-0.18	**-0.17**	-0.26	-0.38	-0.28	-0.55	-0.28	**-0.30**	-0.25
123	**-0.46**	-0.36	-0.56	-0.54	-0.50	**-0.48**	-0.55	-0.56	**-0.51**	-0.51	-0.26	-0.19	-0.44	-0.24	**-0.24**	-0.16	**-0.17**	-0.25	-0.36	-0.28	-0.55	-0.27	**-0.30**	-0.25
115	**-0.52**	-0.32	-0.31	-0.25	-0.28	**-0.48**	-0.46	-0.54	**-0.32**	-0.64	-0.41	-0.12	-0.28	-0.18	**-0.18**	-0.18	**-0.16**	-0.25	-0.36	-0.30	-0.48	-0.25	**-0.31**	-0.22
161	**-0.40**	-0.31	-0.35	-0.29	-0.39	**-0.43**	-0.48	-0.54	**-0.36**	-0.55	-0.26	-0.12	-0.31	-0.21	**-0.20**	-0.16	**-0.16**	-0.24	-0.34	-0.30	-0.48	-0.23	**-0.31**	-0.22
122	**-0.36**	-0.28	-0.49	-0.50	-0.50	**-0.48**	-0.51	-0.51	**-0.48**	-0.43	-0.20	-0.18	-0.41	-0.23	**-0.27**	-0.15	**-0.17**	-0.24	-0.34	-0.27	-0.52	-0.25	**-0.29**	-0.24
41	**-0.31**	-0.24	-0.25	-0.21	-0.37	**-0.39**	-0.42	-0.45	**-0.31**	-0.52	-0.23	-0.12	-0.25	-0.20	**-0.22**	-0.15	**-0.15**	-0.24	-0.30	-0.27	-0.44	-0.23	**-0.28**	-0.20
107	**-0.32**	-0.28	-0.31	-0.24	-0.40	**-0.40**	-0.38	-0.45	**-0.33**	-0.26	-0.18	-0.12	-0.26	-0.21	**-0.20**	-0.12	**-0.15**	-0.23	-0.26	-0.24	-0.40	-0.21	**-0.25**	-0.19
198	**-0.31**	-0.29	-0.29	-0.29	-0.21	**-0.17**	-0.31	-0.36	**-0.26**	-0.68	-0.37	-0.05	-0.14	-0.08	**0.00**	-0.16	**-0.11**	-0.15	-0.26	-0.21	-0.25	-0.18	**-0.22**	-0.12
137	**-0.28**	-0.22	-0.40	-0.42	-0.48	**-0.44**	-0.45	-0.42	**-0.42**	-0.40	-0.17	-0.18	-0.35	-0.22	**-0.29**	-0.14	**-0.16**	-0.23	-0.29	-0.24	-0.49	-0.24	**-0.25**	-0.22

Table 23: Difference in code coverages (x100) between entire and cut sequences (entire - cut), worst 27 code groups - all genomes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
192	**0.24**	0.16	0.16	0.08	0.04	**0.31**	0.25	0.18	**0.37**	0.47	0.37	0.01	0.08	0.12	**0.08**	0.09	**0.05**	0.15	0.22	0.16	0.25	0.14	**0.17**	0.16
87	**0.23**	0.20	0.19	0.10	0.05	**0.27**	0.29	0.25	**0.38**	0.46	0.37	0.02	0.09	0.11	**0.11**	0.09	**0.05**	0.17	0.22	0.15	0.28	0.16	**0.17**	0.16
53	**0.22**	0.14	0.17	0.11	-0.02	**0.26**	0.19	0.15	**0.26**	0.49	0.35	0.03	0.09	0.09	**0.07**	0.07	**0.02**	0.14	0.24	0.17	0.21	0.12	**0.17**	0.17
56	**0.27**	0.15	0.18	0.11	0.10	**0.30**	0.23	0.24	**0.24**	0.25	0.19	0.03	0.13	0.12	**0.09**	0.05	**0.01**	0.08	0.17	0.15	0.19	0.12	**0.15**	0.16
89	**0.26**	0.19	0.21	0.13	0.11	**0.27**	0.27	0.31	**0.24**	0.24	0.19	0.04	0.15	0.11	**0.11**	0.05	**0.02**	0.09	0.17	0.14	0.23	0.14	**0.15**	0.16
191	**0.28**	0.19	0.23	0.20	0.03	**0.27**	0.28	0.26	**0.38**	0.48	0.38	0.06	0.11	0.11	**0.09**	0.09	**0.07**	0.14	0.20	0.15	0.28	0.13	**0.16**	0.15
86	**0.27**	0.23	0.26	0.22	0.04	**0.23**	0.33	0.32	**0.39**	0.47	0.38	0.07	0.12	0.10	**0.11**	0.09	**0.07**	0.16	0.20	0.13	0.31	0.15	**0.16**	0.16
88	**0.13**	0.11	0.13	0.04	0.03	**0.18**	0.21	0.16	**0.29**	0.15	0.22	0.03	0.06	0.09	**0.11**	0.06	**0.04**	0.13	0.22	0.15	0.21	0.13	**0.17**	0.16
135	**0.31**	0.18	0.25	0.22	0.09	**0.26**	0.26	0.32	**0.25**	0.27	0.20	0.08	0.16	0.11	**0.09**	0.05	**0.03**	0.07	0.15	0.13	0.22	0.11	**0.14**	0.15
145	**0.31**	0.22	0.28	0.24	0.10	**0.23**	0.31	0.39	**0.26**	0.26	0.20	0.09	0.17	0.10	**0.11**	0.05	**0.03**	0.09	0.15	0.12	0.26	0.13	**0.14**	0.15
99	**0.13**	0.17	0.08	0.00	0.00	**0.21**	0.23	0.22	**0.27**	0.39	0.32	-0.01	0.07	0.07	**0.08**	0.07	**0.03**	0.14	0.14	0.09	0.21	0.12	**0.11**	0.10
195	**0.42**	0.19	0.25	0.16	0.13	**0.37**	0.27	0.24	**0.37**	0.48	0.41	0.06	0.15	0.17	**0.12**	0.15	**0.11**	0.21	0.33	0.26	0.29	0.20	**0.26**	0.22
91	**0.41**	0.23	0.28	0.18	0.15	**0.34**	0.32	0.30	**0.38**	0.47	0.40	0.07	0.16	0.15	**0.14**	0.15	**0.11**	0.23	0.33	0.24	0.32	0.22	**0.26**	0.22
57	**0.45**	0.19	0.27	0.18	0.19	**0.37**	0.25	0.30	**0.24**	0.27	0.23	0.08	0.20	0.17	**0.12**	0.10	**0.07**	0.14	0.27	0.24	0.23	0.18	**0.24**	0.21
54	**0.40**	0.17	0.26	0.18	0.07	**0.32**	0.21	0.21	**0.27**	0.50	0.39	0.09	0.16	0.14	**0.11**	0.13	**0.08**	0.20	0.34	0.26	0.25	0.19	**0.25**	0.22
208	**0.44**	0.23	0.30	0.20	0.20	**0.33**	0.30	0.36	**0.25**	0.26	0.22	0.09	0.21	0.16	**0.14**	0.11	**0.08**	0.16	0.28	0.23	0.27	0.20	**0.24**	0.21
90	**0.46**	0.26	0.34	0.29	0.13	**0.29**	0.35	0.38	**0.40**	0.48	0.41	0.12	0.19	0.14	**0.14**	0.15	**0.13**	0.23	0.30	0.23	0.35	0.21	**0.24**	0.21
194	**0.46**	0.22	0.31	0.27	0.12	**0.33**	0.30	0.31	**0.39**	0.50	0.42	0.11	0.18	0.16	**0.12**	0.15	**0.13**	0.21	0.30	0.24	0.32	0.20	**0.24**	0.21
66	**0.32**	0.16	0.14	0.05	0.08	**0.31**	0.20	0.21	**0.27**	0.42	0.36	0.04	0.12	0.14	**0.09**	0.13	**0.09**	0.19	0.25	0.19	0.22	0.16	**0.19**	0.15
147	**0.31**	0.20	0.17	0.07	0.09	**0.27**	0.25	0.28	**0.28**	0.41	0.35	0.05	0.14	0.12	**0.11**	0.13	**0.09**	0.21	0.25	0.18	0.25	0.18	**0.19**	0.16
136	**0.49**	0.22	0.33	0.29	0.18	**0.33**	0.28	0.37	**0.26**	0.28	0.24	0.13	0.23	0.16	**0.12**	0.11	**0.09**	0.13	0.25	0.22	0.26	0.18	**0.22**	0.20
207	**0.49**	0.26	0.36	0.31	0.19	**0.29**	0.33	0.44	**0.27**	0.27	0.23	0.14	0.24	0.15	**0.15**	0.11	**0.09**	0.15	0.25	0.21	0.30	0.19	**0.22**	0.20
149	**0.35**	0.20	0.19	0.10	0.15	**0.26**	0.23	0.34	**0.15**	0.19	0.17	0.06	0.19	0.12	**0.12**	0.09	**0.06**	0.13	0.20	0.16	0.20	0.16	**0.17**	0.15
93	**0.34**	0.23	0.41	0.37	0.24	**0.19**	0.29	0.41	**0.20**	0.20	0.06	0.16	0.28	0.16	**0.15**	0.07	**0.07**	0.14	0.24	0.21	0.26	0.16	**0.22**	0.21
146	**0.68**	0.33	0.52	0.43	0.43	**0.44**	0.46	0.57	**0.41**	0.29	0.28	0.22	0.43	0.26	**0.22**	0.18	**0.19**	0.25	0.37	0.32	0.48	0.28	**0.33**	0.29
210	**-0.10**	0.00	0.12	0.05	0.08	**0.08**	0.13	0.08	**0.18**	0.00	-0.02	0.04	0.06	0.09	**0.14**	0.01	**0.01**	0.11	0.19	0.15	0.15	0.08	**0.15**	0.16
154	**0.20**	0.16	0.23	0.15	0.20	**0.17**	0.19	0.31	**0.08**	0.12	0.00	0.08	0.23	0.14	**0.12**	0.05	**0.04**	0.12	0.19	0.16	0.16	0.13	**0.16**	0.15

Table 24: Difference in code coverages (x100) between entire and cut sequences (entire - cut), 27 remainder codes - 'model' genomes

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
r_1	**0.32**	0.24	0.28	0.29	0.27	**0.15**	0.26	0.33	**0.24**	0.55	0.37	0.13	0.23	0.08	**0.13**	0.18	**0.16**	0.20	0.26	0.19	0.33	0.19	**0.21**	0.10
r_2	**0.20**	0.19	0.28	0.30	0.37	**0.14**	0.23	0.26	**0.27**	0.48	0.22	0.12	0.24	0.13	**0.12**	0.15	**0.17**	0.18	0.23	0.20	0.29	0.15	**0.21**	0.10
r_3	**0.26**	0.16	0.14	0.17	0.31	**0.19**	0.21	0.20	**0.28**	0.51	0.36	0.10	0.15	0.10	**0.14**	0.18	**0.18**	0.19	0.20	0.18	0.30	0.18	**0.18**	0.07
r_4	**0.20**	0.17	0.20	0.22	0.22	**0.16**	0.23	0.21	**0.33**	0.69	0.48	0.12	0.14	0.07	**0.16**	0.21	**0.20**	0.27	0.28	0.20	0.36	0.19	**0.21**	0.09
r_5	**0.08**	0.12	0.20	0.23	0.32	**0.15**	0.21	0.15	**0.36**	0.61	0.33	0.10	0.15	0.12	**0.15**	0.19	**0.20**	0.24	0.26	0.21	0.32	0.15	**0.21**	0.09
r_6	**0.27**	0.23	0.11	0.05	0.20	**0.12**	0.22	0.23	**0.10**	0.48	0.28	0.06	0.14	0.06	**0.07**	0.13	**0.13**	0.18	0.23	0.19	0.24	0.16	**0.20**	0.09
r_7	**0.15**	0.18	0.12	0.07	0.30	**0.11**	0.19	0.17	**0.12**	0.40	0.13	0.05	0.15	0.12	**0.06**	0.11	**0.13**	0.16	0.21	0.21	0.21	0.13	**0.20**	0.09
r_8	**0.23**	0.25	0.24	0.31	0.14	**0.04**	0.19	0.23	**0.15**	0.78	0.36	0.01	0.07	0.01	**-0.13**	0.16	**0.11**	0.10	0.12	0.09	0.13	0.08	**0.11**	-0.02
r_9	**0.15**	0.15	0.03	-0.02	0.14	**0.13**	0.19	0.12	**0.19**	0.61	0.40	0.04	0.05	0.05	**0.09**	0.17	**0.16**	0.25	0.25	0.20	0.28	0.16	**0.20**	0.09
r_10	**0.02**	0.11	0.03	0.00	0.24	**0.12**	0.17	0.06	**0.22**	0.54	0.25	0.03	0.06	0.11	**0.08**	0.14	**0.16**	0.22	0.23	0.21	0.24	0.13	**0.21**	0.09
r_11	**0.26**	0.22	0.44	0.49	0.45	**0.18**	0.32	0.24	**0.36**	0.36	0.13	0.19	0.31	0.16	**0.16**	0.12	**0.17**	0.18	0.26	0.22	0.35	0.18	**0.23**	0.16
r_12	**0.21**	0.19	0.23	0.27	0.23	**0.18**	0.23	0.37	**0.12**	0.30	0.14	0.08	0.22	0.04	**0.10**	0.08	**0.07**	0.07	0.11	0.07	0.27	0.10	**0.09**	0.03
r_13	**0.09**	0.14	0.23	0.28	0.33	**0.17**	0.21	0.31	**0.15**	0.22	-0.01	0.07	0.24	0.09	**0.09**	0.06	**0.07**	0.05	0.09	0.08	0.23	0.06	**0.09**	0.03
r_14	**0.08**	0.11	0.15	0.20	0.17	**0.18**	0.21	0.26	**0.21**	0.43	0.26	0.06	0.13	0.03	**0.12**	0.12	**0.10**	0.13	0.14	0.07	0.30	0.10	**0.09**	0.02
r_15	**0.15**	0.11	0.09	0.15	0.26	**0.22**	0.19	0.24	**0.16**	0.25	0.14	0.05	0.15	0.06	**0.11**	0.09	**0.09**	0.06	0.06	0.05	0.24	0.09	**0.06**	0.00
r_16	**-0.04**	0.06	0.15	0.21	0.27	**0.17**	0.19	0.19	**0.24**	0.35	0.11	0.05	0.15	0.08	**0.11**	0.09	**0.11**	0.11	0.11	0.09	0.27	0.06	**0.09**	0.02
r_17	**0.03**	0.12	0.07	0.05	0.25	**0.14**	0.17	0.21	**0.00**	0.15	-0.09	-0.01	0.15	0.08	**0.03**	0.02	**0.04**	0.03	0.06	0.08	0.15	0.04	**0.08**	0.02
r_18	**0.15**	0.17	0.06	0.03	0.15	**0.15**	0.19	0.28	**-0.03**	0.23	0.06	0.01	0.14	0.02	**0.04**	0.04	**0.03**	0.05	0.09	0.07	0.19	0.07	**0.08**	0.02
r_19	**0.26**	0.21	0.38	0.46	0.31	**0.22**	0.32	0.35	**0.21**	0.18	0.06	0.15	0.29	0.07	**0.14**	0.05	**0.07**	0.07	0.14	0.09	0.33	0.12	**0.11**	0.10
r_20	**0.14**	0.16	0.39	0.47	0.41	**0.21**	0.30	0.28	**0.24**	0.11	-0.09	0.14	0.31	0.12	**0.13**	0.03	**0.08**	0.05	0.12	0.10	0.30	0.09	**0.11**	0.09
r_21	**0.03**	0.10	-0.02	-0.04	0.10	**0.15**	0.17	0.17	**0.07**	0.36	0.17	-0.01	0.05	0.01	**0.06**	0.07	**0.07**	0.11	0.11	0.08	0.22	0.07	**0.09**	0.02
r_22	**-0.09**	0.05	-0.02	-0.02	0.20	**0.14**	0.15	0.10	**0.09**	0.28	0.02	-0.02	0.06	0.07	**0.05**	0.05	**0.07**	0.09	0.09	0.09	0.18	0.04	**0.09**	0.02
r_23	**0.02**	0.09	0.31	0.40	0.35	**0.22**	0.28	0.17	**0.33**	0.24	0.02	0.12	0.21	0.11	**0.15**	0.06	**0.11**	0.11	0.14	0.10	0.33	0.09	**0.12**	0.09
r_24	**-0.03**	0.02	-0.16	-0.16	0.13	**0.20**	0.13	0.04	**0.11**	0.31	0.17	-0.04	-0.03	0.04	**0.07**	0.08	**0.08**	0.10	0.06	0.06	0.18	0.07	**0.06**	-0.01
r_25	**-0.35**	-0.05	-0.21	-0.19	-0.03	**-0.04**	-0.08	-0.11	**-0.08**	-0.03	-0.10	-0.10	-0.16	-0.06	**-0.01**	-0.06	**-0.04**	-0.02	-0.11	-0.07	-0.08	-0.07	**-0.08**	-0.10
r_26	**0.41**	0.29	0.17	0.24	0.13	**0.08**	0.18	0.28	**0.07**	0.68	0.39	0.01	0.07	-0.01	**-0.14**	0.15	**0.09**	0.05	0.07	0.06	0.10	0.11	**0.08**	-0.04
r_27	**0.08**	0.06	0.16	0.27	0.28	**0.27**	0.25	0.11	**0.34**	0.27	0.17	0.10	0.12	0.08	**0.17**	0.09	**0.12**	0.12	0.11	0.08	0.33	0.11	**0.09**	0.07

6.2 Appendix B: additional figures

As with the tables, for reasons of space the figure captions are not as informative as those in the previous chapters. For a better understanding, please refer to the part of the Results where the graphs of interest are commented.

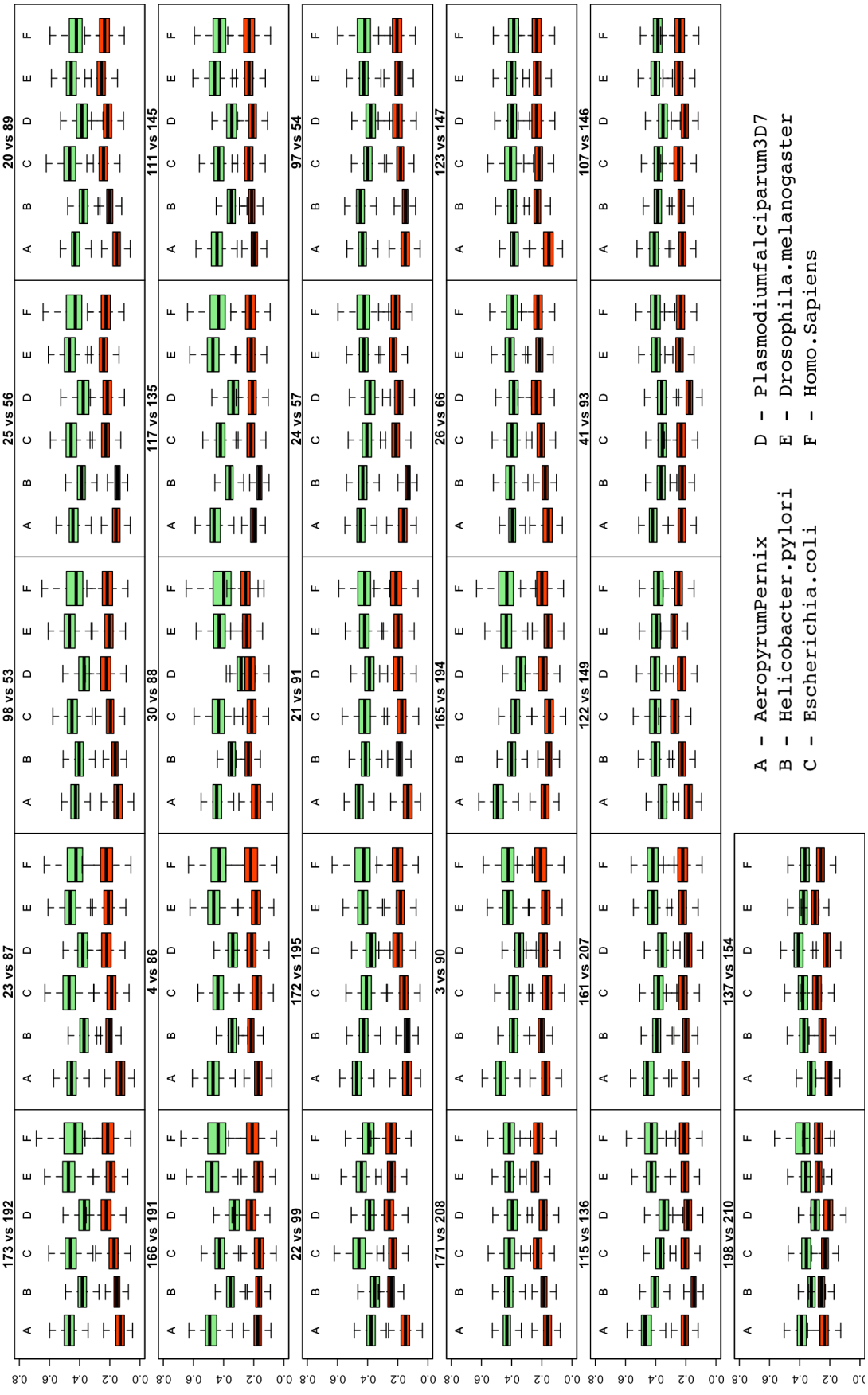


Figure 32: Code coverage distributions considering every sequence - all pairs of code groups, 'model' genomes

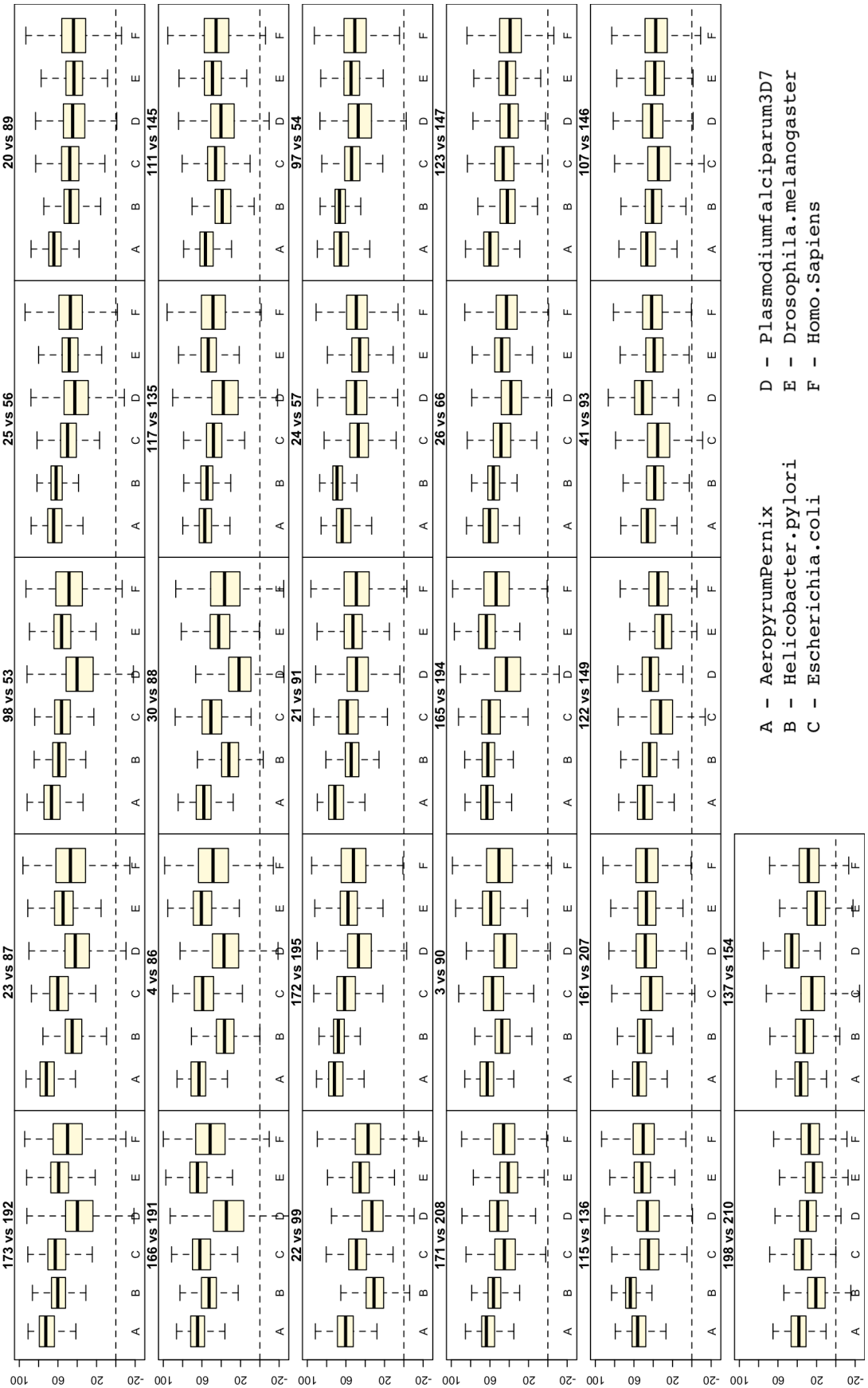


Figure 33: Code coverage considering every sequence: weighted percentage difference distributions - all pairs of code groups, 'model' genomes (1)

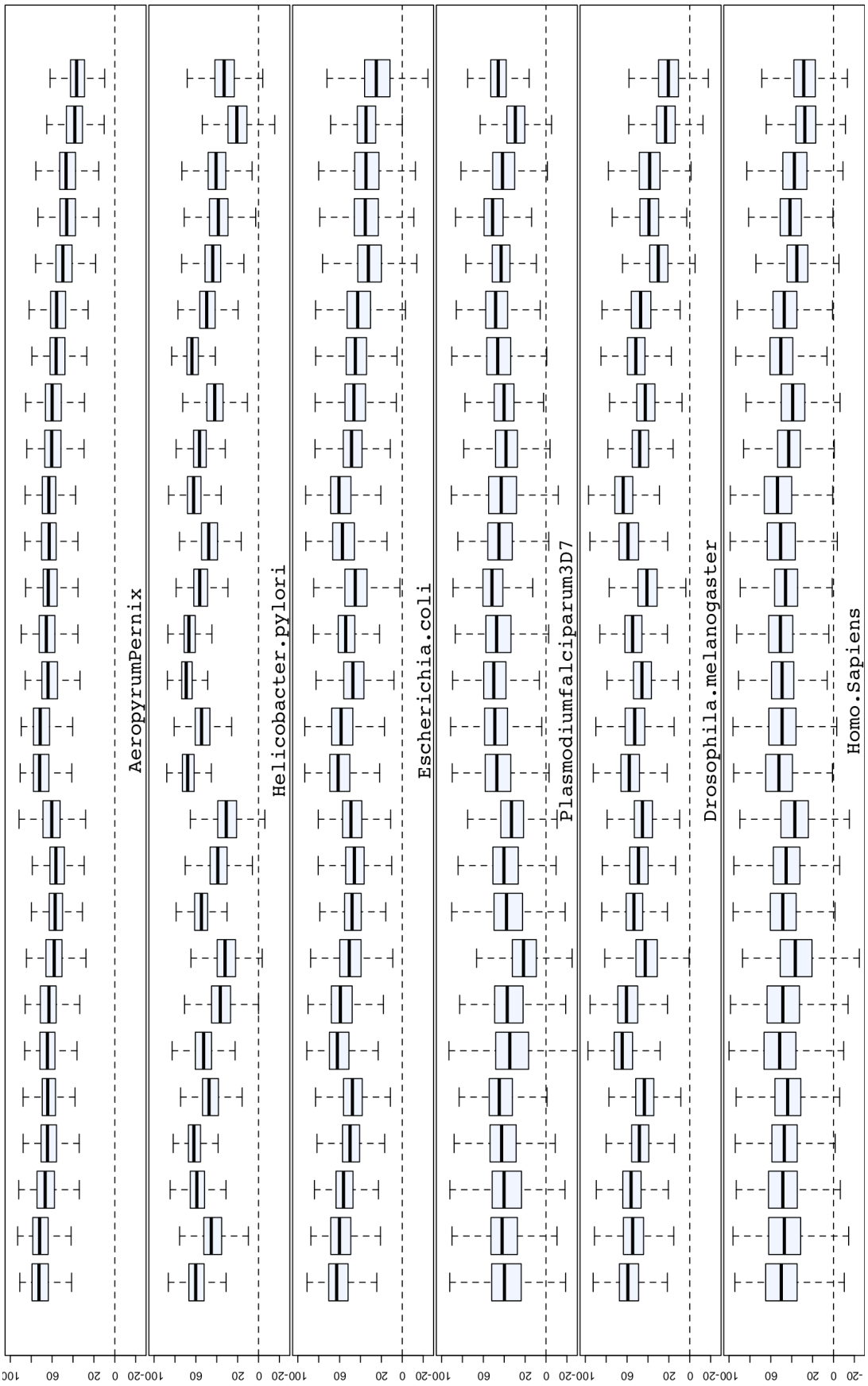


Figure 34: Code coverage considering every sequence: weighted percentage difference distributions - all pairs of code groups, 'model' genomes (2)

Figure 35: Code coverage considering every sequence: all the code groups - Drosophila.melanogaster

Figure 36: Code coverage considering every sequence: all the code groups - Homo.Sapiens

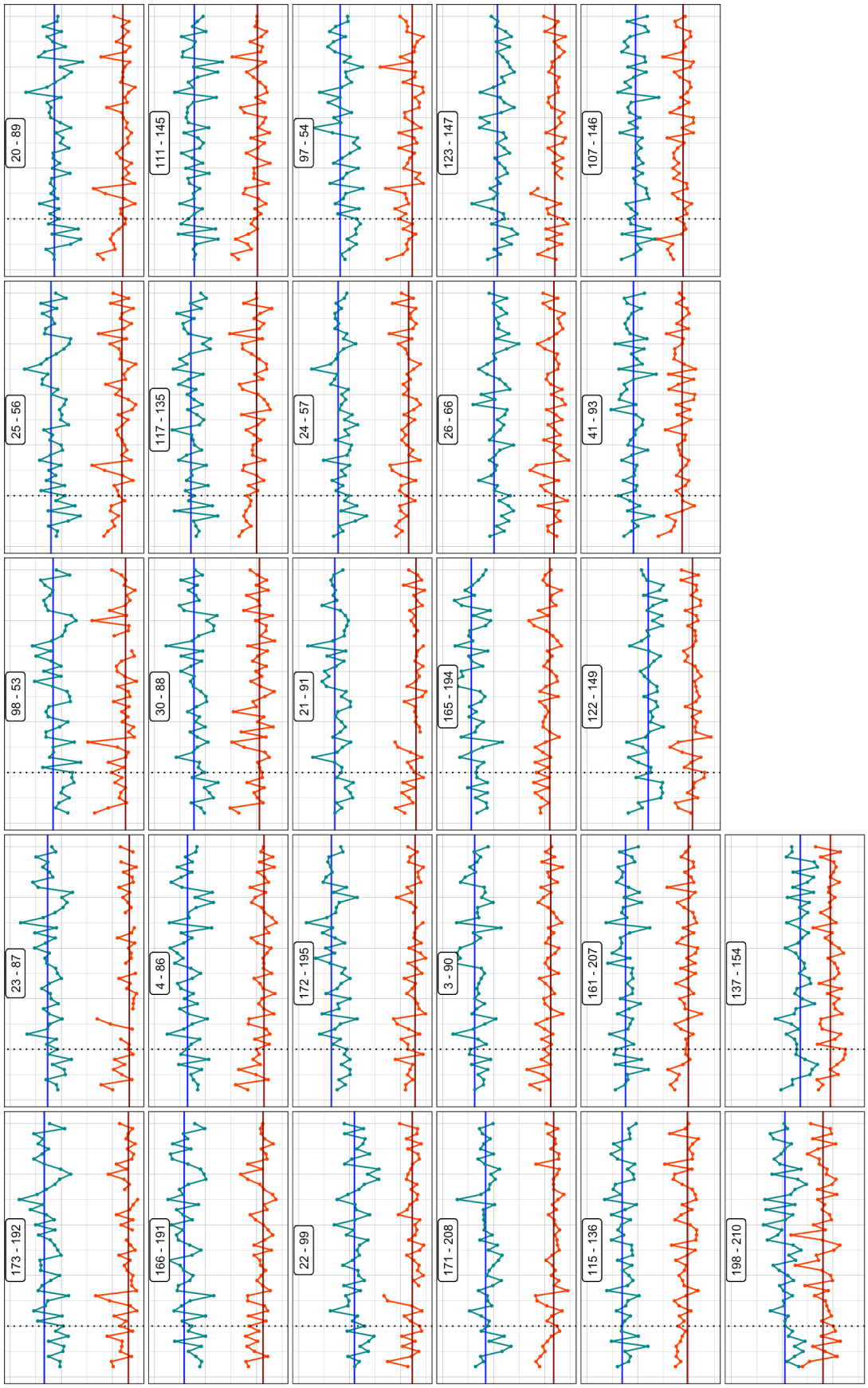


Figure 37: Code coverage by position results for all the 27 code pairs, first 50 codons only - AeropyrumPernix

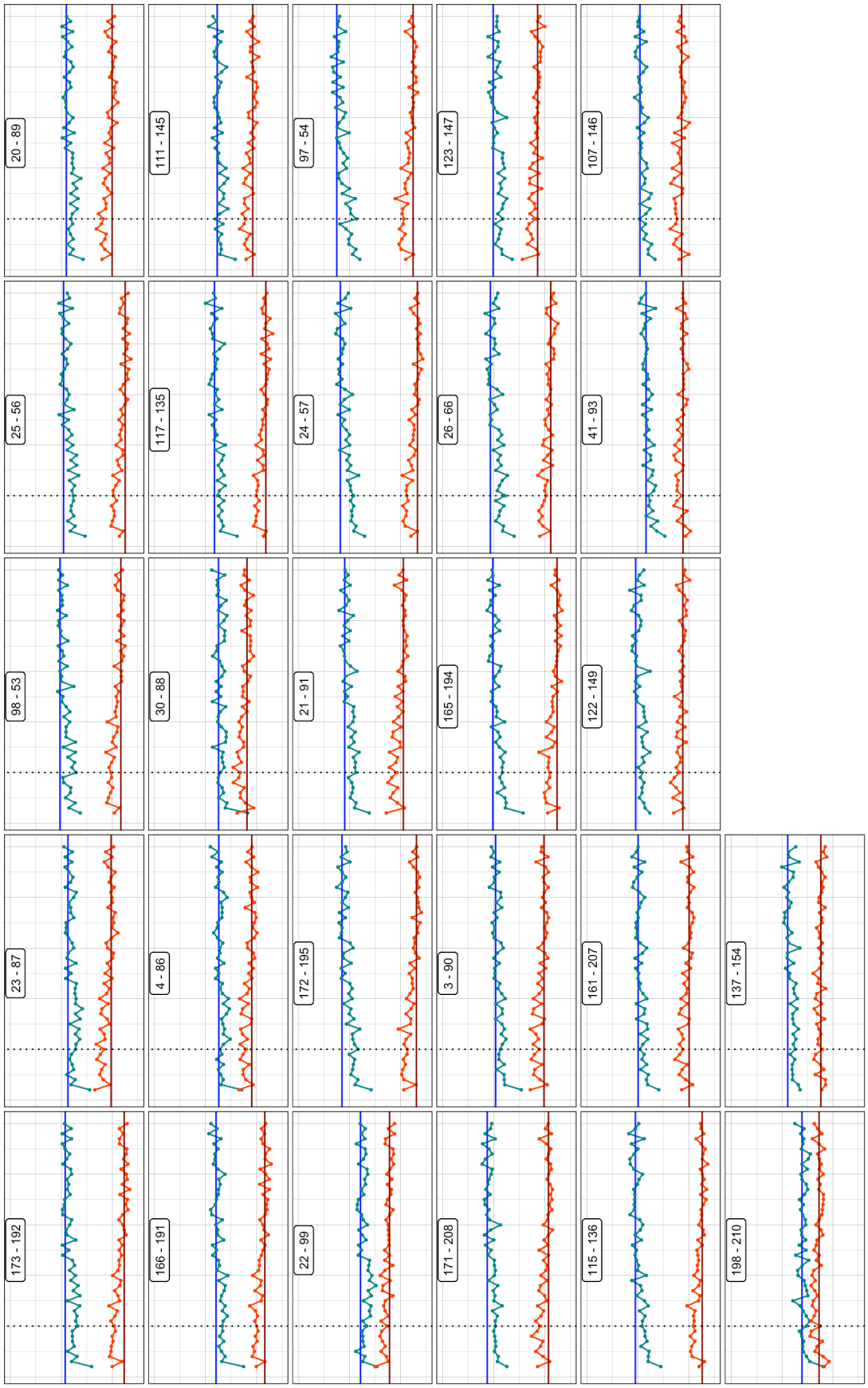


Figure 38: Code coverage by position results for all the 27 code pairs, first 50 codons only - Helicobacter.pylori

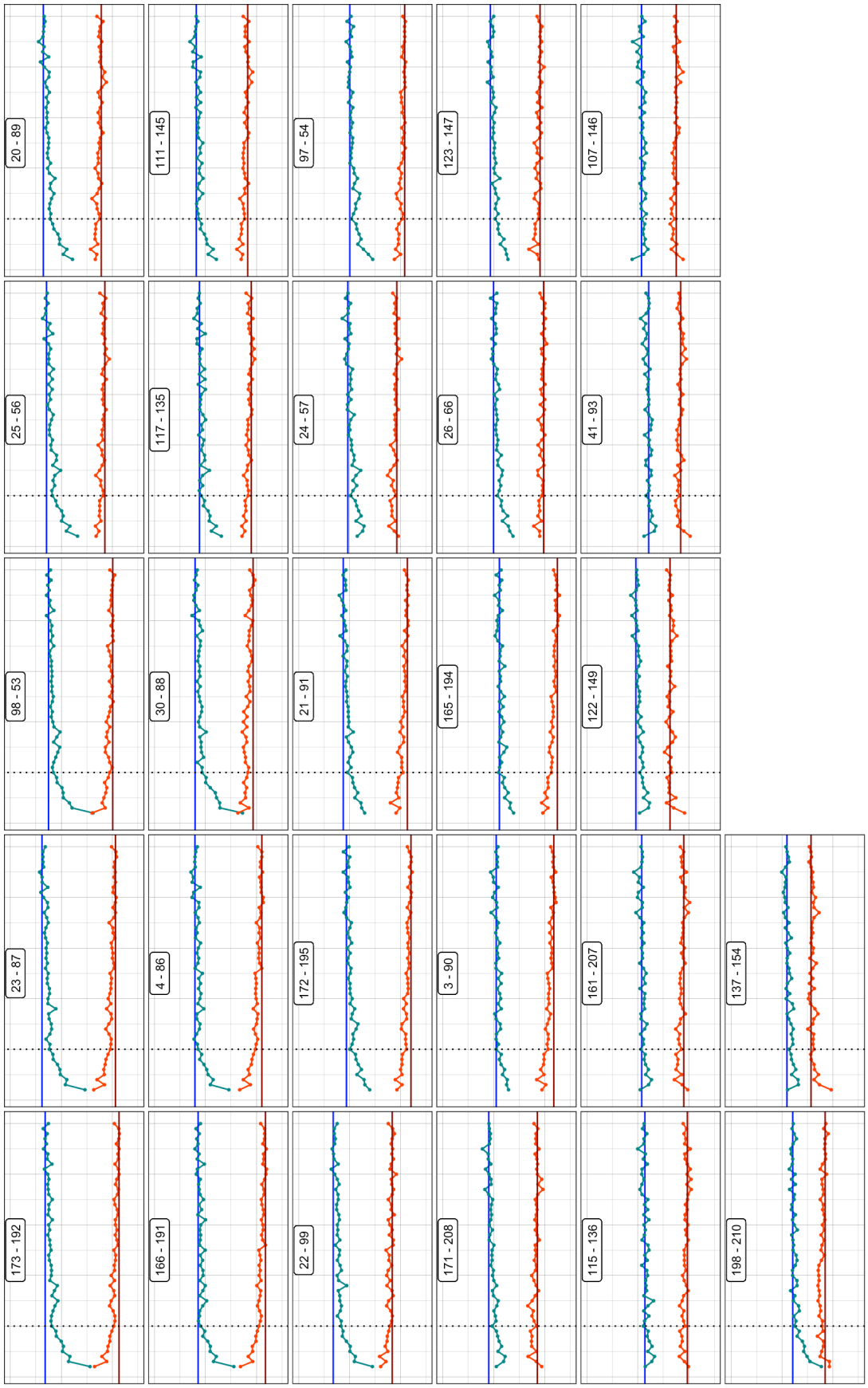


Figure 39: Code coverage by position results for all the 27 code pairs, first 50 codons only - Escherichia.coli

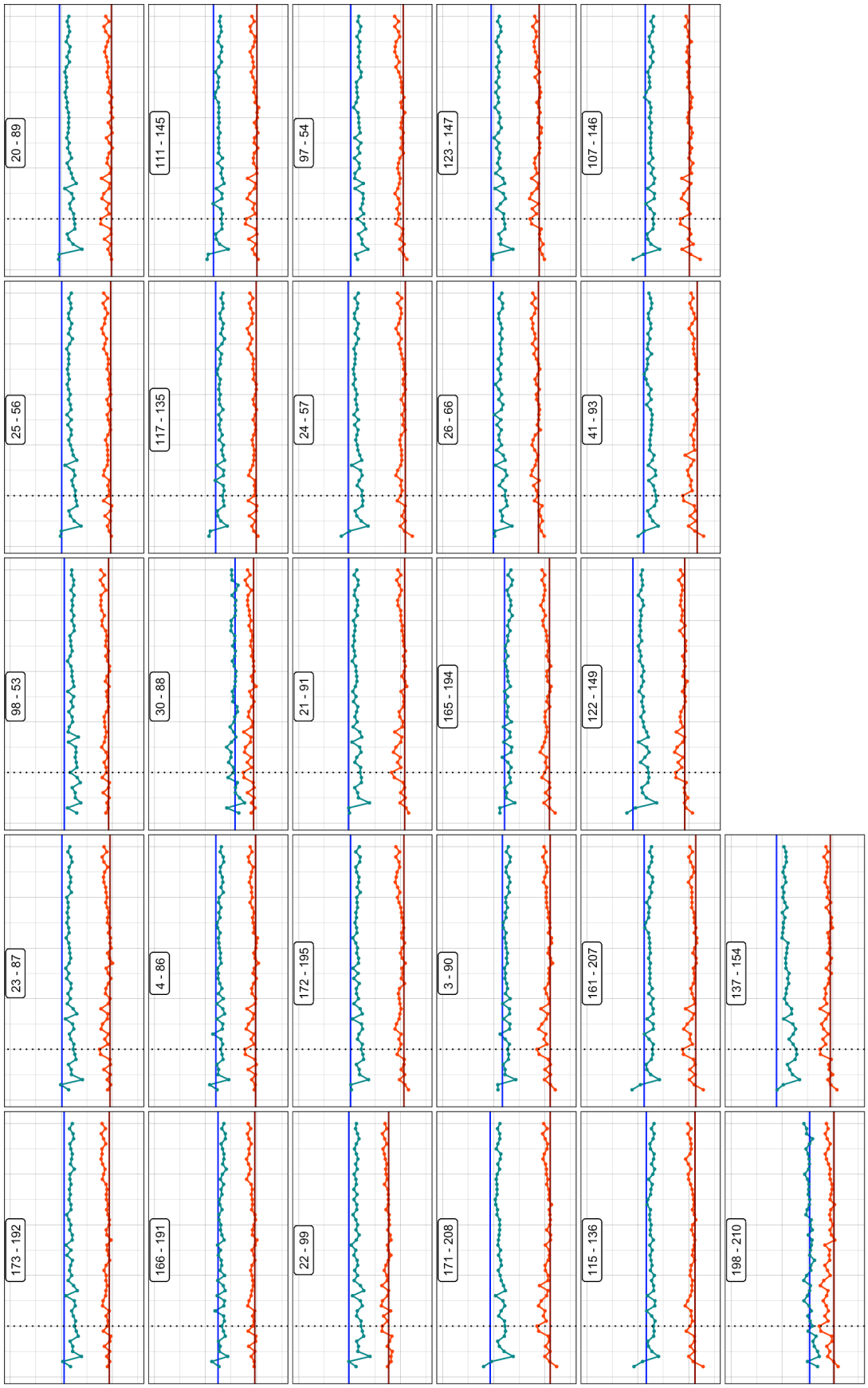


Figure 40: Code coverage by position results for all the 27 code pairs, first 50 codons only - Plasmodiumfalciparum3D7

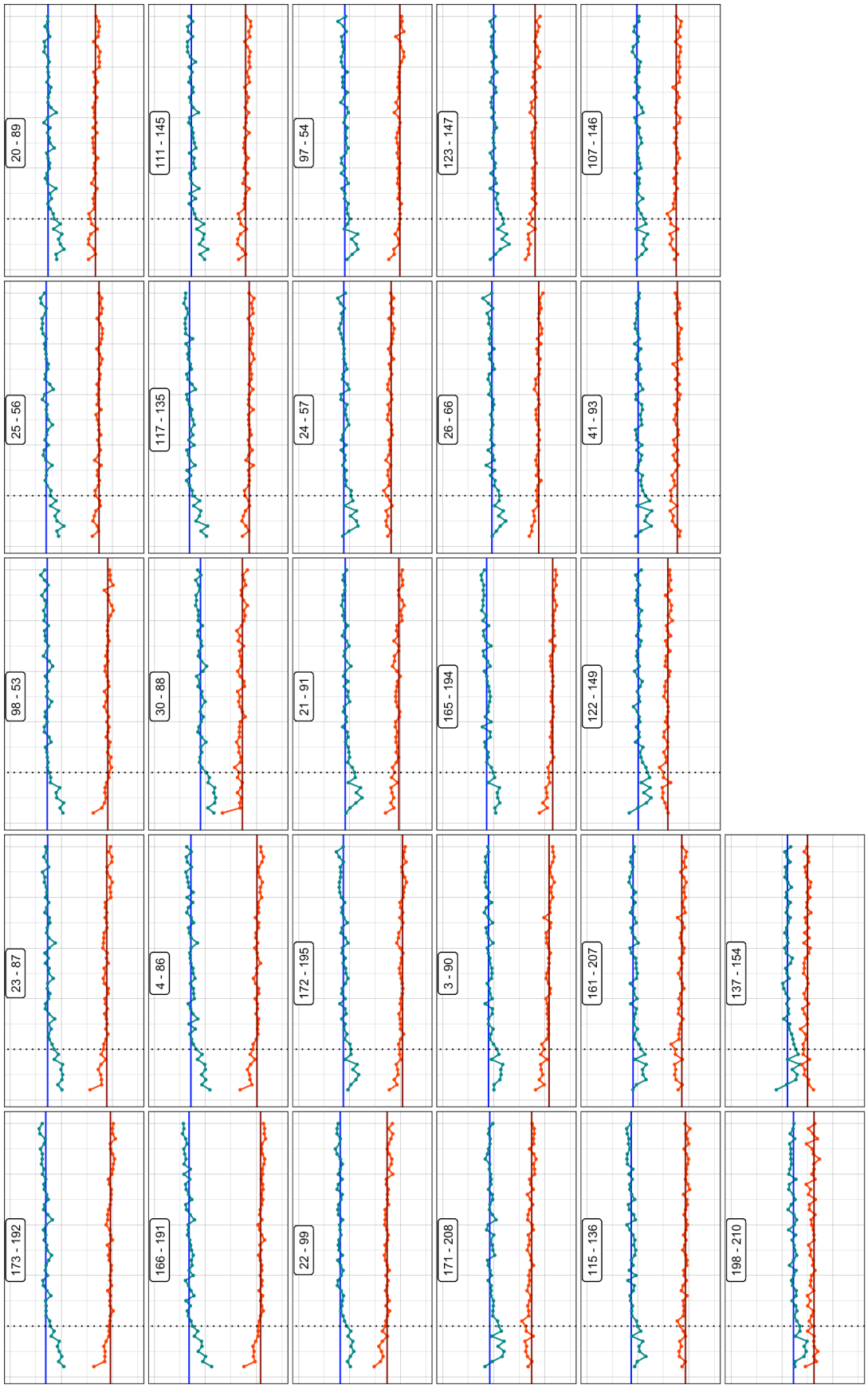


Figure 41: Code coverage by position results for all the 27 code pairs, first 50 codons only - Drosophila.melanogaster

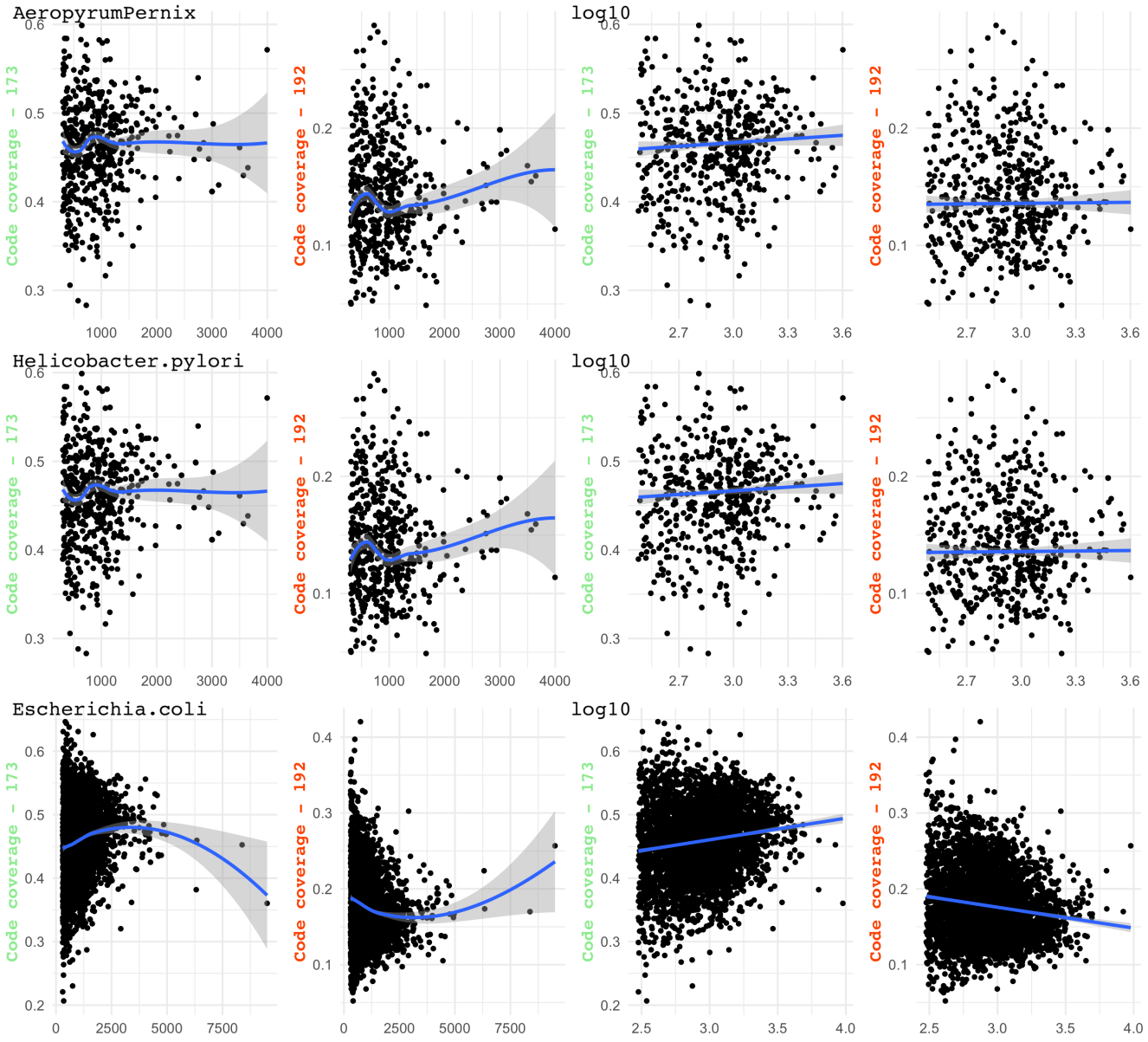


Figure 42: Sequence length effect - without and with log transformation, 'model' genomes (1)

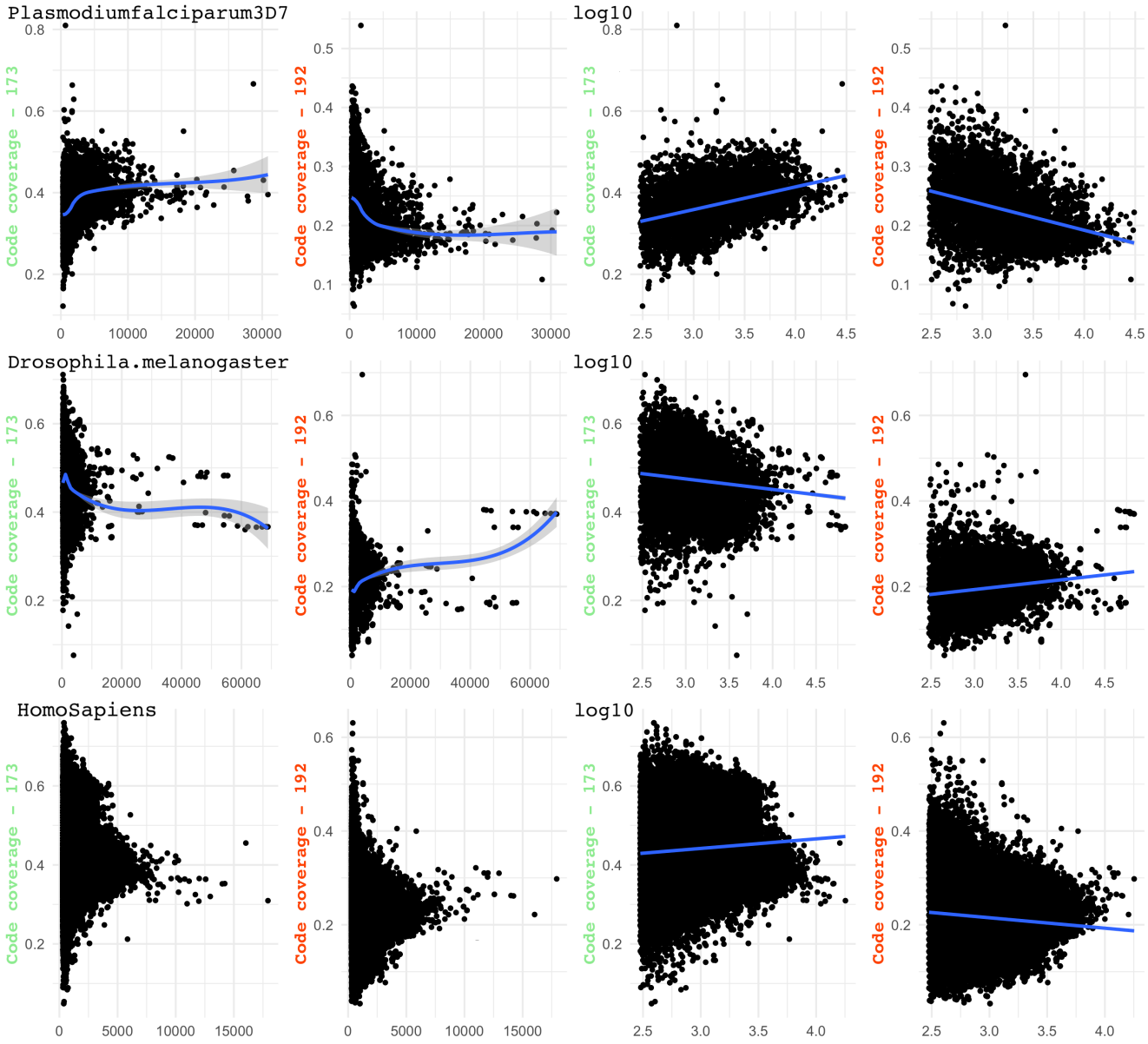


Figure 43: Sequence length effect - without and with log transformation, 'model' genomes (2)

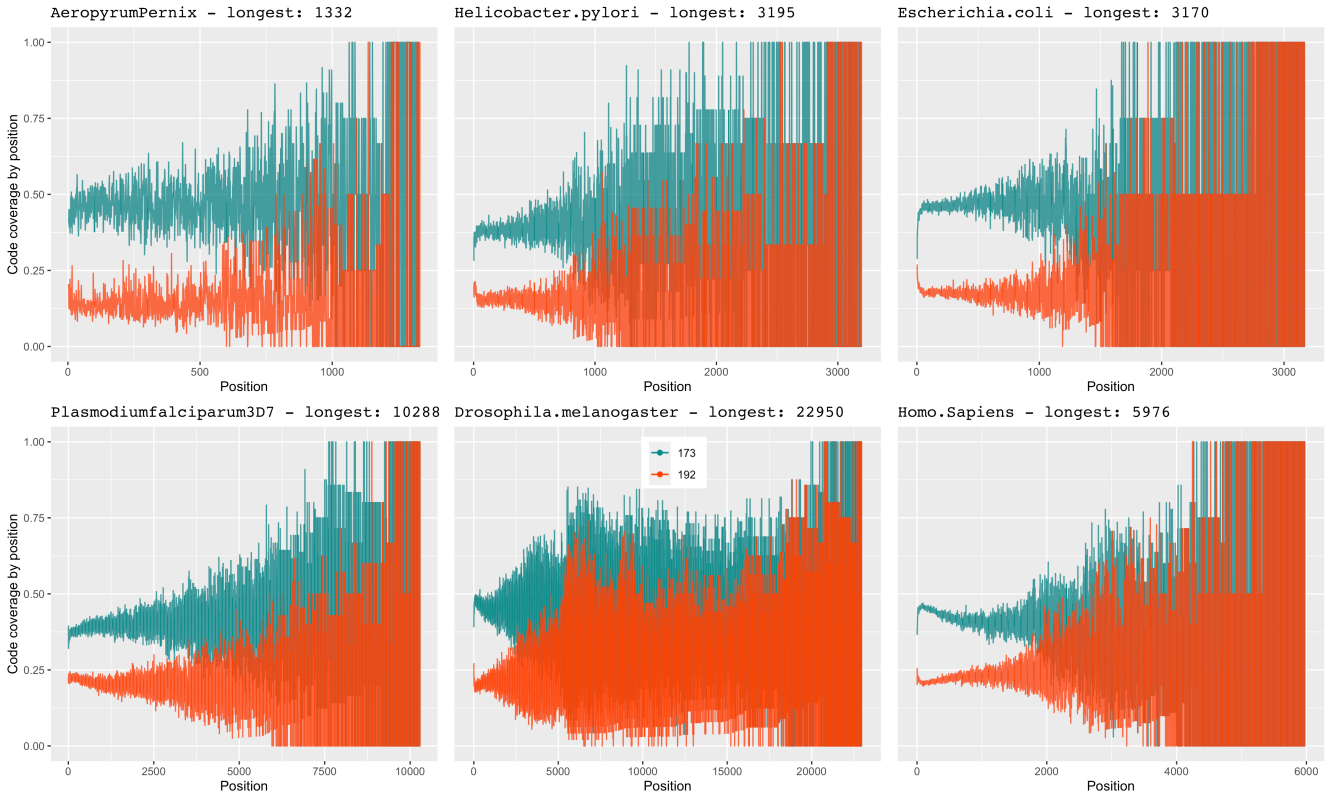


Figure 44: Code coverage by position - 'model' genomes, all x range

6.3 Appendix C: code

In this part will be stored the useful code for the analysis conducted. For the sake of clarity, the code parts will be organized as they were presented in the chapter **Data and algorithms description**.

For space issues, only the lines of code strictly necessary for the calculations previously described will be included in this section[5].

6.3.1 Useful and recurrent objects

In the following, we assume that the *Rdata* files are stored in a folder called *data*. Useful and recurrent object evaluated in the following code are:

[5]If you are interested in further analysis and are curious about other parts of the code (e.g. for minor calculations or graphics) or need some data, I will be glad to provide you with the material if you contact me by email.

```
lista.files <- list.files("data") # Rdata files list
lista.org <- unlist( # genomes name list
  lapply(
    lista.files,
    FUN = function(x) {
      unlist(strsplit(x, split = ".RData"))
    }
  )
)
norg <- length(lista.org)
isgood <- rep(FALSE, norg) # indicates if there is at least one cds in the genome
names(isgood) <- lista.org
```

And useful libraries for the analysis are:

```
library(seqinr)
library(gtools)
library(xtable)
library(mathDNA)
library(tastypie)
library(data.table)
library(ggplot2)
```

6.3.2 Cutting the sequences

This part will be recurrent in the analysis. Here we include the part that refers to only one genome, while in the following this part will be included in a for loop that runs through the genomes.

```
    vname     <- lista.org[i]
    cat(paste("Genome: ",vname,"\n",sep=""))
    load(paste("data/",lista.files[i],sep=""))

    xx0       <- get(vname)
    rm(list=(vname))
```

```
    ## cutting part - with cutseq2 (faster than cutseq)
    cut1 <- 39 # removed from the beginning
    cut2 <- 30 # removed from the end
    for(p in 1:length(xx0)){
      seq <- xx0[[p]]
      np0 <- length(seq)
      seq <- mathDNA::cutseq2(seq, head = cut1, tail = cut2)
      np       <- length(seq)
      if((np0 - np) != (cut1 + cut2)){
        stop("Error in cutseq()")
      }
      xx0[[p]] <- seq
    }
    cat(paste("Removed", cut1, "from the head and", cut2,
              "from the tail, applied to", p, "sequences.\n"))
    ## end cutting part
```

6.3.3 Codon usage on entire genomes

```
lett <- c("A", "C", "G", "T")
tre <- (permutations(4, 3, v = lett, repeats = T))
tre.s <- apply(tre, MARGIN = 1, FUN = paste, collapse = "")

cu0 <- matrix(0, nrow = 64, ncol = norg)
rownames(cu0) <- tre.s
colnames(cu0) <- lista.org

nseqs_cu <- rep(NA, norg) # number of sequences considered for codon usage
names(nseqs_cu) <- lista.org
for (i in 1:norg) {
  vname <- lista.org[i]
  cat("--- Codon Usage \n")
  cat(paste("Genome: ", vname, "\n", sep = ""))
  load(paste("data/", lista.files[i], sep = ""))
```

```
  xx0 <- get(vname)
  rm(list = (vname))

  ## cutting part
  cut1 <- 39
  cut2 <- 30
  for (p in 1:length(xx0)) {
    seq <- xx0[[p]]
    np0 <- length(seq)
    seq <- mathDNA::cutseq2(seq, head = cut1, tail = cut2)
    np <- length(seq)
    if ((np0 - np) != (cut1 + cut2)) {
      stop("Error in cutseq()")
    }
    xx0[[p]] <- seq
  }
  cat(paste("Removed", cut1, "from the head and", cut2,
            "from the tail, applied to", p, "sequences.\n"))
  ## end cutting part

  nseqs_cu[i] <- p # processed sequences for each genome
  xx0 <- unlist(xx0)
  n <- length(xx0)

  ncod[i] <- n
  xe0 <- matrix(as.vector(xx0), ncol = 3, byrow = TRUE)
  cu0[, i] <- as.vector(prop.table(table(xe0[, 3], xe0[, 2], xe0[, 1]))) # codon usage
}
```

6.3.4 Code coverage considering entire genomes

```
RES0 <- matrix(NA, ncol = 216, nrow = norg)
rownames(RES0) <- lista.org
colnames(RES0) <- 1:216
for (i in 1:norg) {
```

```
  # coverage frame 0
  RES0[i, ] <- t(apply(ccod,
    MARGIN = 2,
    FUN = mathDNA::cover,
    codons = tre.s,
    usage = cu0[, i]
  ) * 100)
}
```

6.3.5 Rolling means

For this analysis, we will use the table of the circular code groups (Table 1, object *res*) and the matrix of all the codons in the 216 code groups (object *ccod*).

```
thr <- 1000
bw <- seq(3, 31, by = 2) # spans
nb <- length(bw)

load("data/eqc.RData")
eqc2 <- eqc
eqc <- eqc[, c(1, 6, 5, 3, 2, 8, 7, 4)]
colnames(eqc) <- tran
res <- eqc

ccod <- t(as.matrix(read.table("codici_circolari.txt", header = FALSE, sep = "", as.is = TRUE)))
rownames(ccod) <- 1:nrow(ccod)
colnames(ccod) <- 1:ncodes

ind1 <- res[1, 1] # 173, best code group
ind8 <- res[1, 8] # 192, worst code group
RE1 <- array(NA, dim = c(thr, nb, norg)) # results for the best code
dimnames(RE1) <- list(1:thr, bw, lista.org)
RE8 <- RE3 <- RE1
for (i in 1:norg) {
  vname <- lista.org[i]
```

```
set1 <- ccod[, ind1]
set8 <- ccod[, ind8]
set3 <- setdiff(tre.s, union(ccod[, res[1, 1]], ccod[, res[1, 8]]))

cat("--- Rolling windows \n")
cat(paste("Genome: ", vname, "\n", sep = ""))
load(paste("data/", lista.files[i], sep = ""))
xx0 <- get(vname)
rm(list = (vname))

lseq <- sapply(xx0, FUN = length)
xx0 <- xx0[lseq >= (thr * 3)] # remove sequences shorter than thr

## cutting part
cut1 <- 39
cut2 <- 30
for (p in 1:length(xx0)) {
  seq <- xx0[[p]]
  np0 <- length(seq)
  seq <- mathDNA::cutseq2(seq, head = cut1, tail = cut2)
  np <- length(seq)
  if ((np0 - np) != (cut1 + cut2)) {
    stop("Error in cutseq()")
  }
  xx0[[p]] <- seq
}
cat(paste(
  "Removed", cut1, "from the head and", cut2,
  "from the tail, applied to", p, "sequences.\n"
))
## end cutting part

nn <- length(xx0)
nseq[i] <- nn
if (nn > 0) {
  isgood[i] <- TRUE
```

```
    lseq <- sapply(xx0, FUN = length) # sequence length
    ncod[i] <- sum(unlist(lseq)) / 3 # codons in i-th genome
    nomi <- 1:nn # names(xx0)
    re1 <- array(NA, dim = c(thr, nb, nn)) # intermediate results
    dimnames(re1) <- list(1:thr, bw, nomi)
    re8 <- re3 <- re1
    for (j in 1:nn) {
      sname <- nomi[j]
      x0 <- xx0[[j]]
      x0 <- apply(matrix(as.vector(x0), ncol = 3, byrow = TRUE), FUN = paste, collapse = "", MARGIN = 1)
      x0 <- x0[1:thr]
      cov1 <- as.integer(x0 %in% set1)
      cov8 <- as.integer(x0 %in% set8)
      cov3 <- as.integer(x0 %in% set3)
      re1[, , j] <- matrix(unlist(frollmean(cov1, n = bw, align = "center")), nrow = thr)
      re8[, , j] <- matrix(unlist(frollmean(cov8, n = bw, align = "center")), nrow = thr)
      re3[, , j] <- matrix(unlist(frollmean(cov3, n = bw, align = "center")), nrow = thr)
    }
    RE1[, , i] <- apply(re1, FUN = mean, MARGIN = c(1, 2), na.rm = TRUE)
    RE8[, , i] <- apply(re8, FUN = mean, MARGIN = c(1, 2), na.rm = TRUE)
    RE3[, , i] <- apply(re3, FUN = mean, MARGIN = c(1, 2), na.rm = TRUE)
  }
}
```

6.3.6 Codon usage for every sequence

The first few lines of code are dedicated to a loop useful for extracting the maximum value of the length of the sequences (parameter *maxlenseq*) and the total number of bases present in each genome (vector *totsize*). Moreover, as described in previous chapters, only solution A will be presented. In the vector *problematicn* will be stored the length (in bases) of the sequences that generates unacceptable results.

```
lenseq <- NULL
totsize <- NULL
for (i in 1:norg) {
```

```
  vname <- lista.org[i]
  cat("--- Max number of sequences and number of bases: \n")
  cat(paste("Genome: ", vname, "\n", sep = ""))
  load(paste("data/", lista.files[i], sep = ""))
  genom <- get(vname)
  rm(list = (vname))
  lenseq[i] <- length(genom)
  unlisted <- unlist(genom)
  totsize[i] <- length(unlisted)
  cat(lenseq[i], "    ", totsize[i], "\n")
  rm(genom)
  rm(unlisted)
}
names(totsize) <- lista.org
maxlenseq <- max(lenseq)

cu_each <- array(NA, dim = c(norg, maxlenseq, length(tre.s)))
dimnames(cu_each) <- list(lista.org, 1:maxlenseq, tre.s)

problematicn <- numeric(0)
nseqs_cu <- rep(NA, norg)
names(nseqs_cu) <- lista.org

for (i in 1:norg) {
  vname <- lista.org[i]
  cat("--- Codon Usage For Each Sequence \n")
  cat(paste("Genome: ", vname, "\n", sep = ""))
  load(paste("data/", lista.files[i], sep = ""))
  xx0 <- get(vname)
  rm(list = (vname))

  ## cutting part
  cut1 <- 39
  cut2 <- 30
  for (p in 1:length(xx0)) {
    seq <- xx0[[p]]
```

```
    np0 <- length(seq)
    seq <- mathDNA::cutseq2(seq, head = cut1, tail = cut2)
    np <- length(seq)
    if ((np0 - np) != (cut1 + cut2)) {
      stop("Error in cutseq()")
    }
    xx0[[p]] <- seq
  }
  cat(paste(
    "Removed", cut1, "from the head and", cut2,
    "from the tail, applied to", p, "sequences.\n"
  ))
  ## end cutting part

  nseqs_cu[i] <- p
  listofseqs <- xx0
  rm(xx0)

  for (d in 1:length(listofseqs)) {
    xx0 <- listofseqs[[d]]
    n <- length(xx0)
    xe0 <- matrix(as.vector(xx0), ncol = 3, byrow = TRUE)
    a <- as.vector(prop.table(table(xe0[, 3], xe0[, 2], xe0[, 1])))
    if (length(a) == 64) {
      cu_each[lista.org[i], d, ] <- as.vector(prop.table(table(xe0[, 3], xe0[, 2], xe0[, 1])))
    }
    else {
      cu_each[lista.org[i], d, ] <- rep(NA, 64)
      problematicn <- c(problematicn, n)
    }
  }
}
```

6.3.7 Code coverage considering every sequence

```
res_allseqs <- array(NA, dim = c(24, 151245, 27 * 2))
dimnames(res_allseqs) <- list(lista.org, 1:151245, c(res[, 1], res[, 8]))

for (i in 1:24) {
  cat("--- CU for each sequence, all 27 best and 27 worst groups: \n")
  cat(paste("Genome: ", lista.org[i], "\n", sep = ""))
  for (pd in 1:151245) {
    for (sg in dimnames(res_allseqs)[[3]]) {
      res_allseqs[i, pd, sg] <- mathDNA::cover(
        ccod[, as.numeric(sg)],
        codons = tre.s,
        usage = cu_each_bigg[i, pd, ]
      )
    }
  }
}
```

6.3.8 Code coverage by position

The first few lines of code are dedicated to a loop useful for extracting the length of the longest sequence among all the ones of the genomes of interest (parameter *maxlencod*). The sequences shorter than 200 codons are excluded from this analysis. In this case we do not have the *cutting part*, since we want to take into account the entire sequences.

```
maxlens <- NULL
for (i in 1:norg) {
  vname <- lista.org[i]
  cat("--- Longest sequence \n")
  cat(paste("Genome: ", vname, "\n", sep = ""))
  load(paste("data/", lista.files[i], sep = ""))
  genom <- get(vname)
  rm(list = (vname)) # rimuove il doppione
  maxlens[i] <- max(sapply(genom, length))
```

```
  cat(maxlens[i], "\n")
}
maxlencod <- max(maxlens) / 3 # codons
cat(paste("\n --- The longest sequence is "), maxlencod, "codons long. \n")
cat(paste("And it is from", lista.org[which.max(maxlens)], "\n\n"))

# thr <- 200

RESpos1 <- matrix(NA, nrow = norg, ncol = maxlencod) # results for the best code
rownames(RESpos1) <- lista.org
colnames(RESpos1) <- 1:maxlencod
RESpos8 <- RESpos3 <- RESpos1
for (i in 1:norg) {
  vname <- lista.org[i]

  set1 <- ccod[, ind1]
  set8 <- ccod[, ind8]
  set3 <- setdiff(tre.s, union(ccod[, res[1, 1]], ccod[, res[1, 8]]))

  cat("--- Means by positions \n")
  cat(paste("Genome: ", vname, "\n", sep = ""))
  load(paste("data/", lista.files[i], sep = ""))
  xx0 <- get(vname)
  rm(list = (vname))

  lseq <- sapply(xx0, FUN = length)
  xx0 <- xx0[lseq >= (thr * 3)] # remove sequences shorter than 200 codons

  nn <- length(xx0)
  nseq[i] <- nn # numero di cds
  if (nn > 0) {
    isgood[i] <- TRUE
    lseq <- sapply(xx0, FUN = length)
    ncod[i] <- sum(unlist(lseq)) / 3
    nomi <- 1:nn
```

```
    re1 <- matrix(NA, nrow = nn, ncol = maxlencod) # different from previous re1
    dimnames(re1) <- list(nomi, 1:maxlencod)
    re8 <- re3 <- re1
    for (j in 1:nn) {
      sname <- nomi[j]
      x0 <- xx0[[j]]
      x0 <- apply(matrix(as.vector(x0), ncol = 3, byrow = TRUE), FUN = paste, collapse = "", MARGIN = 1)
      cov1 <- as.integer(x0 %in% set1)
      cov8 <- as.integer(x0 %in% set8)
      cov3 <- as.integer(x0 %in% set3)

      re1[j, 1:length(cov1)] <- cov1
      re8[j, 1:length(cov8)] <- cov8
      re3[j, 1:length(cov3)] <- cov3
    }

    RESpos1[i, ] <- colMeans(re1, na.rm = T)
    RESpos8[i, ] <- colMeans(re8, na.rm = T)
    RESpos3[i, ] <- colMeans(re3, na.rm = T)
  }}
```

6.3.9 Bootstrap test

```
codtest <- function(cod, xf, B = 500, quant = c(0.05, 0.95), replace = FALSE, weight = FALSE) {
  y <- setdiff(codn, cod) # 44 codons
  ind <- which(codn %in% y) # indices of the new codons
  ncod <- length(y)
  yf <- xf[ind] / sum(xf[ind]) # codon usage for y re-normalized

  y2 <- matrix(unlist(lapply(y, FUN = strsplit, split = "")), byrow = TRUE, ncol = 3)
  GCy <- apply(y2, MARGIN = 1, FUN = function(x) {
    sum(x %in% c("G", "C"))
  }) # GC content for y
  codu.b <- rep(0, B)
  k <- 0
```

```
  if (weight) {
    prob <- yf
  } else {
    prob <- NULL
  }
  while (k < B) {
    cand <- sample.int(ncod, size = 21, replace = replace, prob = prob)
    GCca <- GCy[cand]
    S <- sum(GCca)
    if (S >= 30 & S <= 33) {
      j <- which(GCca == (S - 30))[1]
      if (!is.na(j)) {
        k <- k + 1
        codu.b[k] <- sum(yf[cand[-j]])
      }
    }
  }
  return(quantile(codu.b, quant))
}
```

List of Tables

1 The 216 circular codes grouped in 27 equivalence classes according to the 8 transformations of the dihedral symmetry group. The rows in bold refers to the 16 classes for which the codes corresponding to the *identity* (*I*, first column) and to the *Keto-Amino* transformation (*KM*, last column) have no common codons. 9

2 Equivalence class formed by eight circular codes. Each column contains codons in 8 of the 216 circular codons, related to each other by transformations of the dihedral group. 10

3 Size of genomes considered in the study, *model* genomes in red. 20

4 Preview of a small part of the results: codon usage of only 20 codons - *AeropyrumPernix* and *Homo.Sapiens*. 25

5 Preview of a small part of the results: code coverage of sets in the first equivalence class only - *AeropyrumPernix* and *Homo.Sapiens*. 27

6 Preview of a small part of the results: code coverage with the rolling mean approach with different spans - *AeropyrumPernix* and *Homo.Sapiens*. 29

7 Preview of a small part of the results: codon usage for the first 5 sequences and 10 codons only - *AeropyrumPernix* and *Homo.Sapiens*. 31

8 Preview of a small part of the results: code coverages of the first three pairs of best and worst codes in the first 5 sequences only - *AeropyrumPernix* and *Drosophila.melanogaster*. 32

9 Example of positional approach 33

10 Preview of a small part of the results: code coverages computed on the first and last positions - *AeropyrumPernix* and *Helicobacter.pylori*. 34

11 Differences in codon usage between entire and cut sequences for the codons in the best (*173*) and worst (*192*) codes and for the start codon - *model* genomes. These differences have been calculated using the formula below. 37

12 Differences in code coverage between entire and cut sequences for the 27 best codes - *model* genomes. These differences have been calculated using the formula below. 39

13 Differences in code coverage between entire and cut sequences for the 27 worst codes - *model* genomes. These differences have been calculated using the formula below. 40

14 Differences in code coverage between entire and cut sequences for the 27 *remainder* sets - *model* genomes. These differences have been calculated using the formula below. 41

15 R-squared of quadratic regression: codes *173*, *192* and remainder- all the genomes under analysis. . . 49

16 Spearman's rank correlation cofficients between code coverage on every sequence of the first three pairs of best and worst codes and length of the sequence - *model* genomes. 53

17 Number of sequences longer and shorter than 1000 codons - *model* genomes. 55

18 Comparison of code coverage results for best (*173*) and worst (*192*) codes considering the sequences split according to their length - *model* genomes. 55

19 Different values in codon usage (x100) between entire and cut sequences - 'model' genomes 69

20 R-squared of quadratic regression - codes 23 and 87 70

21 R-squared of quadratic regression - codes 98 and 53 71

22 Difference in code coverages (x100) between entire and cut sequences (entire - cut), best 27 code groups - all genomes 72

23 Difference in code coverages (x100) between entire and cut sequences (entire - cut), worst 27 code groups - all genomes 73

24 Difference in code coverages (x100) between entire and cut sequences (entire - cut), 27 remainder codes - 'model' genomes 74

List of Figures

1 The translation process (from NIH, National Human Genome Research Institute). 2

2 Translation table of the genetic code. 3

3 Comma free codes - example (reproduced from Giannerini et al., 2021). 5

4 Circular codes - example (reproduced from Giannerini et al., 2021). 6

5 Circular permutation - example. 7

6 Transformations of the nucleotides forming the dihedral group (reproduced from Giannerini et al., 2021). 7

7 Codon usage and code coverage - example (reproduced from Giannerini et al., 2021). 11

8 Circular code coverage - universal properties (reproduced from Giannerini et al., 2021). 12

9 Coverage of the best code (173, in blue) and of the worst code (192, in red) at the beginning and at the end of the sequences in *E.coli* (reproduced from Giannerini et al., 2021). 13

10 `mathDNA` package logo 21

11 Help pages for mathDNA 22

12 R help page for `cutseq2` from `mathDNA` 23

13 Code coverage distribution of best (173, in blue), worst (192, in red) and remainder (in green) codes - *model* genomes. 23

14 R help page for `cutseq2` from `mathDNA` 26

15 Coverages of the best code (*173*) calculated using the rolling means approach: cut (green box) vs entire (blue box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli.* 42

16 Coverages of the best code (*173*) calculated using the rolling means approach: cut (green box) vs entire (blue box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens.* 43

17 Coverages of the worst code (*192*) calculated using the rolling means approach: cut (red box) vs entire (yellow box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli.* 43

18 Coverages of the best code (*192*) calculated using the rolling means approach: cut (red box) vs entire (yellow box) sequences. The code coverage results computed considering the whole genome are displayed with blue and red lines respectively when the entire and the cut sequences are taken into account. - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens.* 44

19 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *AeropyrumPernix*. 46

20 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Helicobacter.pylori*. 46

21 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Escherichia.coli*. 47

22 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Plasmodiumfalciparum3D7*. 47

23 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Drosophila.melanogaster*. 47

24 Relationships between the coverage of the best (*173*), worst(*192*) and *remainder* codes for individual sequences - *Homo.Sapiens*. 48

25 Distributions of coverages of best (*173*, in blue) and worst (*192*, in red) codes computed considering every sequence - *model* genomes. 50

26 Relationship between coverage on every sequence of best (*173*) and worst (*192*) codes and length of the sequence. In blue the *LOESS* regression curve. - *Plasmodiumfalciparum3D7* 53

27 Relationship between coverage on every sequence of best (*173*) and worst (*192*) codes and logarithm of sequence length. In blue the simple linear regression curve. - *Plasmodiumfalciparum3D7* 54

28 Coverages of the best code (*173*) calculated using the rolling means approach: cut (light green box) vs entire (dark green box) sequences. The benchmarks this time are the values for global code coverage considering the group of sequences longer than 1000 codons (in red) and the group of sequences shorter than 1000 codons (in blue) - *AeropyrumPernix, Helicobacter.pylori and Escherichia.coli.* . . . 56

29 Coverages of the best code (*173*) calculated using the rolling means approach: cut (light green box) vs entire (dark green box) sequences. The benchmarks this time are the values for global code coverage considering the group of sequences longer than 1000 codons (in red) and the group of sequences shorter than 1000 codons (in blue) - *Plasmodiumfalciparum3D7, Drosophila.melanogaster and Homo.Sapiens.* 56

30 Coverages of the codes *173* (in green) and *192* (in red) by position, focus on the first 50 positions only. The coverage values obtained by considering the whole genome for code *173* (blue line) and *192* (dark red line) are also shown - *model* genomes. 59

31 Bootstrap test results for the 16 disjoint pairs of *best* and *worst* codes and all the 24 genomes under analysis, with 10000 bootstrap replications and $\alpha = 0.0001$. In every plot are displayed the bootstrap rejection bands under the null hypothesis at $\alpha = 0.0001$ that the relation is produced by chance (in green) and the results of the coverage of the *worst* (in red) and the *remainder* (in blue) codes. 62

32 Code coverage distributions considering every sequence - all pairs of code groups, 'model' genomes . 76

33 Code coverage considering every sequence: weighted percentage difference distributions - all pairs of code groups, 'model' genomes (1) 77

34 Code coverage considering every sequence: weighted percentage difference distributions - all pairs of code groups, 'model' genomes (2) 78

35 Code coverage considering every sequence: all the code groups - Drosophila.melanogaster 79

36 Code coverage considering every sequence: all the code groups - Homo.Sapiens 80

37 Code coverage by position results for all the 27 code pairs, first 50 codons only - AeropyrumPernix . 81

38 Code coverage by position results for all the 27 code pairs, first 50 codons only - Helicobacter.pylori 82

39 Code coverage by position results for all the 27 code pairs, first 50 codons only - Escherichia.coli . . 83

40 Code coverage by position results for all the 27 code pairs, first 50 codons only - Plasmodiumfalciparum3D7 84

41 Code coverage by position results for all the 27 code pairs, first 50 codons only - Drosophila.melanogaster 85

42 Sequence length effect - without and with log transformation, 'model' genomes (1) 86

43 Sequence length effect - without and with log transformation, 'model' genomes (2) 87

44 Code coverage by position - 'model' genomes, all x range 88