UNIVERSITY OF KENT

MASTER THESIS

PyCUB: A machine exploration of the codon usage bias

Author:
Jérémie KALFON

Supervisor:
Dr. Dominque CHU

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

September 11, 2018


University of
Kent

Declaration of Authorship

I, Jérémie KALFON, declare that this thesis titled, "PyCUB: A machine exploration of the codon usage bias" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 11/09/2018

"What I cannot create I do not understand."

Richard Feynman

UNIVERSITY OF KENT

Abstract

School of Computing

Master of Science

PyCUB: A machine exploration of the codon usage bias

by Jérémie KALFON

The codon usage bias is the DNA sequence pattern problem in protein coding genes, where sets of synonymous codons, blocks of 3 nucleotides which encode for the same amino acid in the transcription process, have a non random distribution.

As codons are very low level features in the genome, many factors might explain their particular distributions. Amongst the stated ones, the GC content of the genome, the tRNA pool, the replication speed of the cell, as well as the size, the evolutionary distance and pressure over the gene might all have a degree of influence over it.

How much does each factor help to explain the bias registered in the codon usage of genes? How much of the codon usage bias are they able to predict?

We set out to study such questions using advanced statistical tools, over a set of related species.

Acknowledgements

I would like to thank Yun Deng for trusting me in using and building upon some of her ideas, knowledge and work, for the development of PyCUB. I would like to particularly thanks Dominique Chu for accepting me in this project, giving me advice throughout the year and feeding me with ideas. Thanks to Tobias Van Der Haar for giving me a lot of information I did not have about comparative genomics and computational biology. Both of you were implied in this Master Thesis in the best way possible and gave me a great introduction to research.

I would also like to thanks Baptiste Tesson for providing some insight and feedbacks about my results. Monique Kalfon for her help in the harsh final days and Juliette Hirsch for her constant support.

Contents

List of Figures

Vocabulary

CS, **ML**, **aws**, **phylogenetics**, **ec2**, **amino acids**, **codons**, **fungi**, **cDNA**, **CDS**, **introns**, **exons**, **mRNA**, **tRNA**, **RNA**, **Lasso**, **CV**, **PCA**, **t-SNE**, **Spearman's rho**, **Fibonacci's heap**, **multinomial distribution**, **multivariate normal**, **partition function**, **entropy**, **AUC entropy**, **pseudo phylogenetic distance**, **Cophenetic matrix**, **cosine similarity**, **grid search**, **K-Means**, **K-Modes**, **Silhouette score**, **Calinski Harabasz score** , **Akaike Information Crierion**, **Bayesian Information Citerion**, **dynamic programming**, **Endres Schrindrelin distance**, **Kullback Leibler divergence**, **Gaussian Mixtures** , **Multilayer Perceptron**, **DBscan**, **JS**, **REST**, **FTP**, **scikit learn**, **CUF**, **CUB**, **GC bias**, **KaKs score**, **hydrophobicity**, **isoelectric point**, **HiC**, **fasta**, **CAI**, **evolutionaryCAI**,

other: API, json, doxygen, gzip, Python, synthetisation cost,glucose cost, Data Science, Computational Biology, Bioinformatics

Chapter 1

Introduction

The codon usage bias (**CUB**) is part of a long standing research in biology, trying to understand the genome, this lengthy protein composed of amino acids (**aa**) referenced to by their letters A,T,C,G,. Together, these bases coming in pairs, can encode for the building blocks of life: **proteins**.

Proteins are also made up of other amino acids. Each protein is encoded in parts of the DNA called **cDNA**. When read, cDNA is translated into **mRNA**, a proto-form of DNA where bases are A,U,C,G. This mRNA goes into a process of "clean-up" called splicing, where sub-parts are removed (**introns**) and others are kept (**exons**). The spliced mRNA can then be translated into proteins. Each amino acid of the protein is encoded as a triplet of base pairs in the coding sequences, called **codons**. A matching from codons to aa is done by a protein complex with **tRNAs**. As in a lot of information codes submitted to a noisy environment (Maheshri and O'Shea, 2007), in the DNA, some codons can encode for the same aa in the protein. They are called synonymous codons. However there seems to be more to it than a simple redundancy.

Data science has always been a big part of genomics and is what most of bioinformatics is based upon. With the recent advances in deep learning and the "ever" continuing Moores law (Goodfellow, Bengio, and Courville, 2016; Moore, 2006), it is a perfect moment to answer questions under the light of scalable statistical tools. Machine learning (**ML**) should be seen as a way to get interesting research directions and even make discoveries when sufficient data is available. Thanks to the increase of multi-modal genomics data from next generation sequencing platforms (NGS) (Schreiber et al., 2018; Lin Liu, 2012) and their widespread availability through open science, such endeavours are becoming possible to the many. Beyond simply describing the problem and what the codon usage bias is, we will set out to answer what metrics are best suited to explore it and what set of phenomenons can best explain it. Then, in closer relation to computer science (**CS**), we will answer how to work on the CUB and how to work with comparative evolutionary genomics. We will also learn how to logically orchestrate a bioinformatics project and how to do statistical computation easily and at scale.

PyCUB is a project that is not solely framed to be a scientific exploration, but also tries to make up to the ideal of reproducible and interpretative science (McNutt, 2014; Baker, 2014). It was thought to be used, to be built upon and modified in order to further the research and answer new questions about the CUB. Moreover this document is meant to allow its reader to have a look at how to build open, computational biology projects in 2018.

1.1 Research Objectives

The research objectives are cross disciplinary and open. They intend to build on Yun Deng's work on a measure of entropy for the Codon Usage Bias. By using computational tools, they will explore the type of data that emerges from this algorithm. They will compare its statistical significance against a set of biological data that have been, for some, related to the CUB. The questions are thus the following:

1. What can we learn about the CUB by using modern Machine Learning technologies?
2. What is the CUB most correlated with and for which species?
3. What are the best statistical measures for the CUB?

We'll try to explore the data in the most unbiased way and present it so as to be able to understand and interact with it in the easiest way.

Secondary goals are of reproducibility and reusability: can anyone reproduce my results? Future users should be able to apply PyCUB to other datasets and use **PyCUB** to answer other questions. They are also about scaling: can PyCUB scale with more datasets? What are the difficulties or errors in the scaling process? What are the tools and techniques used to scale it and what could be changed?

1.2 The Problem

We are presenting here a literature and technology review presenting the problems tackled. The project being interdisciplinary, we will present it along the 3 different paths leading to it, the general questions they pose and how they are relevant to our project.

1.2.1 Problematics Regarding The CUB

How to understand the codon usage bias? What does it entail? What can it explain? And what are the theories behind it?

The codon is one of the early DNA motif. It is at the source of the protein and is related to many different cell processes and evolutionary events. It is generally acknowledged that codon biases reflect a balance between mutational biases and natural selection for translational optimization. It has been shown that Baker's yeast and related organisms' CUB reflect in part the composition of their respective **tRNA pool** -which is the density of various tRNAs in the nuclei-. It is thought that in rapidly growing organisms, a correlation to the tRNA pool might help drive the accuracy and speed of transcription: in this case by natural selection on highly transcribed genes (Trotta, 2013a). Gene expression level is thus linked to the CUB and should, above some threshold, create this transcription bias via natural selection.

GC composition and GC skew also seem to be linked to the CUB. One refers to the general presence, i.e.percentage of G(uanine) and C(ytosine) base pairs in one genome -traces of horizontal gene transfer or mutational bias- whilst the other refers to this presence in specific regions of the genome called strand-specific mutational

biases. **GC bias** seems to be one of the most prevalent effect explaining the CUB (Trotta, 2013a). A similar idea, random mutations on synonymous codons -thus not affecting the composition of the underlying protein- seems to favor particular codons and make some others, called rare codons, under-represented.

Moreover, amino acid conservation, protein hydropathy, and even RNA stability have been shown to have effects on the CUB. Natural selection can be a factor, for example in thermophilic bacteria where replication speed is linked to certain biases, possibly related to the stability of mRNAs at high temperature (Singer and Hickey, 2003). Differential nitrogen availability, due to differences in environment and dietary inputs, is also a determinant of synonymous codon usage in both bacterial and eukaryotic microorganisms (Trotta, 2013b). Low nitrogen availability seems to force species to use nucleotides that require fewer nitrogen atoms to encode the same genes, compared to species surrounded by high nitrogen availability.

Finally the rate of transcription and the particular moment at which a gene has to be transcribed can also play a role in the CUB (Shin et al., 2015). It has been shown that some viruses, which benefit from delayed transcription of their genetic information, have a different CUB from those of their host cell, creating the required delay in the expression of their genetic information by being more difficult for the host cell to transcribe.

All the pieces of information above show that by its fundamentalism, the CUB is a good proxy for a plethora of core events of the cell, thus explaining the wide range of statistical relationships to the CUB and their differences among species and biological fields. However, the question remains: What explains the CUB and what is explained by it? What strengths does each statistical relationship have, given the set of studied species, and are they sufficient to explain the Codon Usage Bias, or is there more to it?

Today, the CUB is mostly understood from the **Mutation-selection-drift balance Model**, which states that "selection favors major codons over minor codons" due to the tRNA pool effect on the speed of translational elongation. "But minor codons are able to persist due to mutation pressure and genetic drift". This is mainly because of the GC content and rare codons effect. It suggests that "selection is generally weak, but that it scales in intensity for higher expression and more functional constraints of coding sequences" (Hershberg and Petrov, 2008).

The CUB has mostly been used as a concept, while trying to answer other related questions about genetic biases and natural selection. It has not, or little, been a research subject in its own right. It may be explained by a somewhat difficult to achieve requirement on the amount of data needed to find statistical relevance across all potential factors.

1.2.2 Problems Regarding Computational Genomics

Since the emergence of large scale genomics and open medical and biological databases, the fields of computational biology & bioinformatics have been booming. Filled with enthusiast biologists and data scientists, genomics and computational genomics have witnessed a radical shift from being a purely wet lab driven science to an extremely diverse and multidisciplinary science.

From the first human genome sequenced (Consortium, 2004), we are now able to gather hundreds of thousands of genomes from different ethnicity and thousands of reference genomes from different species (Sudmant et al., 2015; Zerbino et al., 2018). Big research-friendly public registry and databases, such as the ones in Israel and Denmark are allowing the use of new tools and the discovery of new correlations (Arriel Benis, 2017; Schmidt et al., 2015; Abelson et al., 2018).

The work of Anshul Kundaje, Brendan Frey and other computational biologists give us examples and ideas on what might come out, in the future bioinformatics papers and bio-medical software. The stated goal here is of bridging the gap between phenotype and genotype and at some point... silicon and carbon (Frey, 2017; Church, Gao, and Kosuri, 2012).

However, many problems remain. It has been pointed out that from the sequencing, to the prepossessing, alignment and analysis, a final genetic information can become a giant probabilistic mess. Together with wild approximations this can lead to results which are hard to decipher and also to unfounded claims. A challenge for data scientists. New evaluations and metrics will have to arise. Despite the increasing availability of datasets, there is not yet any public repository matching the requirements to gain significance over genomic data for most research questions (Li et al., 2018).

Moreover, datasets are often dispersed around thousands of lab servers and sometimes neither cited nor shared with their related publications. This is a problem, especially when trying to create a data driven approach to genomics and biology, and it is part of a bigger question on reproducibility in science: not only are datasets hard to get but as in many other fields, reproducing another researcher's work can be extremely complicated. In biology, moreover, software or system requirement specification ("IEEE Guide for Developing System Requirements Specifications" 1998) (**SRS**) are not often given with their corresponding research.

A lot of software exist in genomics, but unfortunately they are often packaged graphical interface software or mere websites to which one has to upload and retrieve its data manually. Code in research is meant to be built upon, and graphic user interfaces on paid software might slow research more than anything else. Code is meant to be improved when a new generation of machine comes up, or changed and merged with little restriction.

Lastly, another challenge is, of course, to present one's work in the best possible way. Allowing interaction and providing for interpretation over the created algorithms and the resulting data (Chris Olah, 2018). A full black box scenario is a no for exploratory research using ML technologies and extracting simple concepts from complex inferences is a requirement for the computational biology and democratic bio-medical software to come (Kalfon, 2018).

1.2.3 Questions regarding ML Technologies

From its roots in statistics with linear regression, Bayesian analysis and latent factor models, in linear algebra with PCA, matrix factorization and in optimization k-means, A* search methods and evolutionary algorithms, machine learning (Alpaydin, 2010) has continuously improved; becoming its own field, being used across scientific domains. Nowadays, it is very much linked to computer science and data

science. Starting with the ongoing big data revolution, machine learning algorithms which quality and statistical power improve with available data, have risen and now receive worldwide recognition, e.g. Neural networks, Bayesian graphical models and other gradient based optimization methods.

Hinton's neural networks (NN) (Hinton and Salakhutdinov, 2006), powered by LeCun's gradient descent (Cun, 1988) have allowed to extract features from images at a scales never achieved before. Grave's attentional Natural Language Processing (NLP) Neural Networks with LSTMs (Mnih et al., 2014) have permitted the witnessed improvement regarding speech recognition and synthesis. All such methods have made way for powerful inference and feature extraction algorithms. Interaction is also achieved by using the power of neural networks with a reinforcement learning framework first introduced by the neuro-dynamic programming book (Bertsekas and Tsitsiklis, 1996). It has allowed research companies to produce the best go players ever and to beat the strongest Dota2 players.

However, we are now talking about requiring Petaflops/s/day to train some types of deep reinforcement learning pipelines, with self play for example. The computer power required to train and produce an output has grown dramatically with the progress of machine learning techniques. This trend is not solely constrained to neural networks. Many optimization problems, potentially solvable by gradient descent, have de facto become much faster, much easier to use, but also more computationally intensive. It is not a problem per say and many argue that algorithms and especially ML programs that scale linearly with the computer power available are the ones that will stay with us in the long run.

Here, the focus is more on the exploration in itself than the ML software. Throughout this exploration, however, different problems will require different strategies and algorithms to be solved quite efficiently.

We will handle multiprocessing t-SNE using Barnes-Hut algorithm (Maaten and Hinton, 2008; Barnes and Hut, 1986), features selection, the **DBSCAN** clustering algorithm (Ester et al., 1996), Gaussian mixtures, etc. Doing so, we will find which ones are the most effective, what their caveats are and how to use them. Such a thought process is a requirement for any data scientist and it appears that biology and genome data are a great way to test the reasoning and limits behind some of these algorithms(Ching et al., 2018). Finally, we will also look at how to interpret their output and whether or not they can be more effective than more regular algorithms such as PCA and k-means.

1.3 Related Work

In the related work, continuing the separation of concepts, we will see the power of ML and its lasting presence in genomics. We will also explain why we have not used deep learning in this project. Why, given the data and objectives, regular ML techniques provided the best tools. We will then consider the packages available today to do research fast and easily, and how biology represents a valuable experimental framework for state of the art ML tools.

Then we will discuss the CUB and how people have tackled this problem, presenting the different metrics and ideas to understand the CUB today, finally explaining more about Yun Deng's work and ideas behind it.

1.3.1 On Genomics Using ML Technologies

A lot has been tried for almost a decade with some important results. They range from improving on state of the art benchmarks with new models, to enabling new research directions and helping to make discoveries that would not have been possible otherwise (Libbrecht and Noble, 2015).

ML applied to genome research is keeping the strong link genomics has with the bio-medical industry, which also uses this technology to improve diagnostics, drug discovery, etc, hoping to reduce costs (Li and Ping, 2018; Benhenda, 2017). DeepGenomics has demonstrated the use of neural networks to predict the specificities of DNA- and RNA-binding proteins (Alipanahi et al., 2015) called *DeepBind*. Kundaje's Lab has worked on predicting transcriptional regulatory activities with Deep Convolutional Networks (Paggi et al., 2017) called *DragoNN*. It is a big step toward deciphering the regulatory DNA sequences and non-coding genetic variants (Movva et al., 2018). Some companies, like Google, are working on it too (Kelley, Snoek, and Rinn, 2016).

The power of neural networks in this domain is starting[1] to be demonstrated, further pushing the benchmarks in many contexts. But more vanilla architectures are also used: in *Clairvoyante* (Luo et al., 2018), researchers demonstrate a multi-task five-layer CNN for predicting variant type: SNP or indel, zygosity, alternative allele and indel length from aligned reads which achieved 98.65% F1-score for whole-genome analysis.

Moreover, far from being restricted to deciphering sequences, work has been undertaken in many other areas: protein interaction network for example. Using novel matrix representation, better comparisons and inference methods and new feature extraction, we are starting to understand more about the mechanisms of the cell (Wang et al., 2017; De Smet and Marchal, 2010). (Niepert, Ahmed, and Kutzkov, 2016) have shown how to use regular convolutional networks to infer better embeddings, i.e. representations of any network data, a very recurrent topic in biology. It also one that has always been very difficult to tackle and has often required the code generating embeddings to be dependent on the studied system.

We also see the use of matrix representation and factorization for multi-omics data (data from many different modalities), enabling feature representations that are the most predictive and explanatory of given processes, creating embeddings which are particularly suited to convey complex biological concepts in a way that allows prediction, and thus discovery (Schreiber et al., 2018). In (Liu et al., 2018) for example, we can see that embeddings can be learned without any feedback or training data, a big leap forward in a field where some data are inherently scarce and difficult to produce -and reproduce- in an unbiased way.

A recurring idea is that the DNA code and inner working of cells might very well be too complicated to be directly understood by humans, especially with the current top-down approaches. ML helps to convey concepts by taking the opposite path, allowing to point out relationships that would not have been found directly otherwise. (Khan et al., 2001; Parrella et al., 2014) have shown that multiple factors and

[1] we are talking here about "recent work". But it is important to remember that no idea is completely new, especially when we can find -somewhat- similar work and ideas from decades past (Khan et al., 2001)

mutations can result into the same outcome. The idea that genes are extremely interrelated is somewhat new and has not been put into practice much, as it is most often an intractable problem without getting a prior knowledge of the studied data.

Graphical models are also quite relevant in Data-Bio-sciences. Their ability to discover patterns even when the training size is small makes them particularly suited to approach many biological questions (Aguiar et al., 2018). Many other related matters can or could be tackled by advanced computational learning methods (Beam et al., 2018; De Cao and Kipf, 2018).

As one may see, deep learning is taking a more and more important part of the recent advances in this field and will likely continue to do so for the next few years (Ching et al., 2018). However, it will be noticed that in this work, no deep learning or other learning methods have been used. It is explained by 3 caveats of such methods that are well displayed in the context of our requirements:

1. We have to iterate, switching between questions and results.
2. We have to explore and understand the obtained results.
3. We have few samples for many tasks and no concept of training sets.

In addition, we are required to try different results fast. These requirements illustrate many weaknesses of today's learning algorithms.

1.3.2 On The CUB From A Biological Standpoint

Recently it has been shown that the CUB can be a proxy to understand even more than previously thought about the genome:

Countless studies have related the synonymous codons to different factors in various organisms: (Diament, Pinter, and Tuller, 2014) have shown that a similar distance between two protein coding genes on the frequency of their codon usage -here both on the entirety of the 64 codons and on synonymous codons-, is correlated with the 3D distance between these genes in the nuclei of eukaryotic species. As it has been shown that this distance is not random but linked with ongoing transcriptions events and protein functions, it can have implications in the future role of simple metrics such as the CUB to infer gene position and transcription (Poyatos and Hurst, 2007; Osborne et al., 2004).

The CUB is also useful when creating new DNA sequences for the production of drugs, bio-fuels or other components with genetically modified bacteria (Athey et al., 2017). Moreover, as stated from this research paper: "An area of research that is currently gaining attention pertains to how codon usage may affect protein structure. It has long been assumed, based on Anfinsen's theorem (Anfinsen, 1973), that since synonymous mutations do not affect the primary structure of a protein, they also should not affect the secondary and tertiary structure." However, it now seems that "synonymous codon changes can affect the translation rate of a protein, which in turn may modulate the folding of the nascent polypeptide chain" (Sander, Chaney, and Clark, 2014; Buhr et al., 2016; O'Brien et al., 2014). As the translation kinetics of a protein depends, at least in part, on the frequency of its codons, having access to codon usage information can be valuable in determining effects of synonymous mutations on protein structure. Links to effects on nucleosome structure, transcription factor binding, splicing efficiency, RNA-protein interactions, micro-RNA

binding, and RNA secondary structure are further described in (Bali and Bebok, 2015; Weatheritt and Babu, 2013; Chamary, Parmley, and Hurst, 2006; Tuller and Zur, 2015).

Recently it was shown that non-optimal codon bias, in terms of adaptation to the tRNA pool, is a mechanism to achieve circadian clock in *cyanobacterium Synechococcus*, where non-optimal codon usage was selected as a post-transcription mechanism to switch between circadian and non-circadian regulation of gene expression as an adaptive response to environmental conditions (Xu et al., 2013).

Another study suggested that transcription factors located within exons provide additional evolutionary constraints that shape the CUB of genes. Approximately 15% of human codons are dual-use codons "duons" that simultaneously specify both amino acids and Transcription Factors (TF) recognition sites, hinting to the fact that TF-imposed constraints are a major driver of codon usage bias (Stergachis et al., 2013). The current research thus focuses on finding to what extent the CUB is predictive of different genome processes and their variations for different organisms and species kingdoms.

An especially related research and one we are building upon is about finding ways to measure the Codon Usage Bias. Different metrics have been achieved and are used according to the questions they are trying to answer (M. Comeron and Aguadé, 1998). One of the best known CUB measure is the Codon Adaptation Index (**CAI**) (Sharp and Li, 1987): which is explained as "being a measure of the deviation of a given protein coding gene sequence with respect to a reference set of genes composed of highly expressed genes, so that CAI provides an indication of gene expression level under the assumption that there is translational selection to optimize gene sequences according to their expression levels". Similar ones exist such as the Relative Codon Adaptation (**RCA**) (Fox and Erill, 2010), the frequency of optimal Codons (FOC) and the effective number of codons (Nc) (Wright, 1990), which measures how far the codon usage of a gene departs from equal usage of synonymous codons. It is linked to population genetics measures and has been called "one of the best measure to display the CUB" (Stergachis et al., 2013).

It needs to be pointed out that the ideas behind such measures are driven by some prior knowledge on the way the CUB works, that is, the evolutionary drift model and the effect of translational selection. Entropy measures however, together with raw frequency and the Nc are measured with little to no prior knowledge on what the values are and what drives them. It can be much more effective in finding what correlates with the CUB. Yun Deng's work builds on this idea, looking at the **entropy** of the distribution of synonymous codons, assuming a **multinomial distribution** (*Analysis of discrete data: The Multinomial Distribution*):

$$\mathbf{X} \sim \mathcal{M}(\mathbf{N}, \mathbf{P}) \tag{1.1}$$

with $\mathcal{M}$ being the Multinomial:

$$\mathcal{M}(\mathbf{N} = n_1, \dots n_m; \mathbf{P} = p_1, \dots p_m) = \frac{n!}{n_1! \dots n_m!} p_1^{n_1} \dots p_m^{n_m} \tag{1.2}$$

we have $\mathbf{p}_{obs} = \mathcal{P}(\mathbf{N}_{obs}, \mathbf{P})$ where $\mathbf{N}_{obs}$ is the distribution of synonymous codons for a given amino acid and where $\mathbf{P}$ is the expected probability that each synonymous codons has to appear in the sequence, given the amino acid. The distribution here is only the number of times the different synonymous codons are used in the sequence.

Thus, the entropy of this amino acid's codon distribution is:

$$\mathcal{H}(\mathbf{N}_{obs}) = -\ln(\frac{\mathbf{p}_{obs}}{\overline{\mathbf{p}}}) \tag{1.3}$$

where:

$$\overline{\mathbf{p}} = \mathrm{E}(\mathbf{X}) \tag{1.4}$$

With $\mathcal{H}$ being the entropy and $\overline{\mathbf{p}}$, the density of the expected distribution of this amino acid codons.

A theoretical entropy definition allows to represent the amount of information conveyed by the CUB for each of the 18 amino acids being encoded by more than one possible codon. Here, the length of a sequence $L = \sum n_i$ and the number of synonymous codons m should have an impact on the entropy value as more information can be conveyed by a bigger and more complex distribution. In addition to this idea of entropy, Yun used another concept of A_{value} of entropy or *entropy cost* which can also be seen as a way to normalize the entropy. It represents how much a given entropy is far from the expected one. It can also be seen as a normalized cost to achieve the distribution:

$$\mathcal{A}(\mathbf{N}_{obs}) = \sum_{\mathbf{Z}_i \in \mathbf{Z_{sub}}} \frac{\overline{\mathbf{p}}}{\exp(\mathcal{P}(\mathbf{Z}_i) \times L)} \tag{1.5}$$

where:

$$\mathbf{Z_{sub}} = \{\forall \mathbf{Z}_i \in \mathbf{Z} | \mathcal{H}(\mathbf{Z}_i) > \mathcal{H}(\mathbf{N}_{obs})\} \tag{1.6}$$

With $\mathbf{Z}$ being the partition function of X. Here 1 will be the least expected entropy given the size of the sequence and number of possible synonymous codons and 0, the expected mean entropy value given the distribution. This measure requires all possible entropy values to be computed, i.e. it requires the partition function $\mathbf{Z}$ for the given multinomial distribution.

Even more theoretically based: simulations and modelling of molecular behaviors have always been behind the theories put forth for the codon usage bias as it is a simple enough molecular system to be modelled and tested for. (Reis, Savva, and Wernisch, 2004) shown that tRNA gene copy numbers (**CN**), i.e. gene redundancy and genome size are interacting forces that ultimately determine the action of translational selection, and that an optimal genome size exists for which this kind of selection is maximal using such a biological model.

The different links and effects on the CUB displayed in recent work, despite drawing a complex and multidimensional picture, stay mostly within the Mutation-selection-drift balance model. We are seeing here the many complex effects allowed.

1.4 The Project

This project started unofficially in December and, during that time, the goal was first to gather as much information as I could about the CUB. I researched about the founding work in the CUB and some more general genomics knowledge. It was mainly achieved by reading papers, summaries and courses. The second was to have a look at the data and the previous work done by Yun Deng. Here, an exploration of

the datasets, some discussions and re-coding some of the main components of Yun's Matlab code were used to familiarize ourselves with the subject.

In parallel, the code was taking shape. I looked at the state of the art in computational biology and scientific programming using python, as well as the many toolkits available to speed up the development.

It was in itself an iterative process, starting from the data exploration and up to different trials, while following an informal SRS, taking inspiration from many scientific data tool kits such as scikit learn , Biopython and CaImAn (Giovannucci et al., 2018). One of the main challenges was to create the memory architecture in order for it to be versatile and comprehensible. During that time I also had to further my understanding of the CUB and the related work, more recently undertaken on the subject.

Feedbacks with my supervisor were done weekly and a 4 to 8 hours a week study was given for the project before the full time start, post final exams. It allowed me to have a head start at the official launch of the project, to review problems and misunderstandings earlier in the development and to get to understanding the subject more broadly than I would have done otherwise.

1.4.1 The Data

From the start, we knew that having different types of data was a requirement if we were to say anything meaningful about the codon usage bias. Thus the goal was to think about what types of data were important. **Homologies** are sets of protein coding genes among a group of species, that are known to come from a common ancestor and which most often share the same function. I.e. they produce proteins having similar functions. Their similarities vary. Homologies are often inferred from the similarity of raw sequence as well as from the **phylogenetic** relation of their species, i.e. their ancestry and common ancestors. Knowing the homologies, we have an information for each gene, to which other species' genes it is related. An important information according to the evolutionary drift model of the CUB. Another important knowledge is to what extent species are related to one another. It can be achieved by retrieving a well-known phylogenetic tree.

For the genetic sequences, we were more interested in cDNA than the raw DNA, which contains information that will not be conveyed with amino acids -which is not to say that this information has no relation whatsoever to the CUB-. Knowing the effect of the tRNA pool on the CUB, tRNA CNs is another important data to be retrieved. It can be found from the labeling of genes.

Moreover, given the wide range of relations to the CUB, there are a lot of meta-data that may be used. Some of them can be extracted from the sequence such as the CAI, the GC content, the length, the reference gene, etc. Here, by reference gene we are referring to the gene used to find the homologies. This gene belongs to the reference species which is a well-known and well-studied species for a particular kingdom. As a rule of thumb, reference species are Homo Sapiens and Mus Musculus (mouse) for vertebrate, Cerevisiae Saccharomyces (yeast) for fungi, Arabidopsis Thaliana for plants and E. Coli for bacteria.-

Some other meta-data may be extracted from elsewhere. Almost anything one can think of about an organism may be extracted, and most of it can be found for reference species. For fungi, we are interested in the noxiousness of each species and its relation to its surroundings and to the temperature. Generally important meta-data would be:

- The function of each protein
- How much expressed each gene is , i.e. mRNA abundance
- How much preserved or recent each protein coding gene is - keeping in mind it will only be known for the reference species genes-
- How regulated it is, e.g. how many regulatory elements it possesses, if the final protein is for internal use or if it is to be secreted by the organism
- The half life of the protein, i.e. how long it stays within the organism before being denatured, giving us information about the speed at which it would need to be re-created.
- The cost of producing the protein for the organism.

We could also be interested in more information about the protein such as its size, hydrophobia, weight, etc. A good proxy for this molecular information is to use the molecular information of each amino acid -only 20- and assemble it according to the composition of the protein. Among the databases that could have been used -see *logbook*- only a few of them were, as connecting the code to the APIs is time-consuming.

1.4.2 The Code Architecture

Here I am talking about the code in itself, the architecture I have followed to build it -its memory and functional logic- and the different packages that will be used in it. I will first talk about the goal meant to be achieved before explaining the different functions in more details. You can find out more about the code from the code itself or its online documentation on Github at `www.jkobject.com/PyCUB`.

Goals

The code is expected to extract information from databases and other data sources. It is expected to represent and to link all the information together and then let the user interact with it. It should have the ability to extract as much details from the raw data it has access to, as possible. Then it should be able to relate different data and to give metrics for this relation. We do not know whether there will be statistical relationship among the data or not. But we know that everything is a result. Getting results that are in line with the general theory would also be an interesting information, as confirming results is something very important. The goal will also be redefined by the time constraints of the project.

We will use different packages either to have access to new tool-kits and possibilities that would hardly be accessible in Python otherwise, or to speed up the workload by not having to re-implement everything. Many python packages are built using C-python and are thus much more efficient than a similar version in python.

Moreover, we wanted to prove the need for more software engineering quality code and/or code presentation in bioinformatics because of reproduction difficulties that have plagued research. We also wanted to show some applications for which the entropy of CUB might not be the best measure to use. In this context, we focused to provide a reproducible version of the work of Alon Diament on the correlation between codon usages and gene location within the nuclei of eukaryotes. It was shown recently that centromeres, telomeres, transfer RNAs (tRNAs), chromosomal breakpoints and early replication origins in SC tend to be co-localized (Duan et al., 2010). This makes one wonder if the position of a protein coding gene can be inferred from some specific features about it such as its function and its codon frequency. In his work, Alon Diament showed exactly that. We tried to replicate his demonstration and to show if our result could be the same as his and if the entropy was a good measure to carry out such kind of work.

Brief Description

workspaces The code has a center object: PyCUB, that is initiated only once and allows the user to interact with the rest of the project for an entire session, i.e. each PyCUB object defines a workspace or session for the user. Other python objects[2] are created and used by the workspace.

homology Object defining a particular homology with each of its species, CUB values[3] and other related data, e.g. data that belongs to a particular gene or to the reference gene[4].

homoset Python mutable objects that are used to interact with sets of homologies. These mutable objects are resembling a python dictionary or a JavaScript Object. They use the [] operator to access a specific homology object. The homoset also contains many functions to interact specifically with groups of homologies. Moreover it contains related data such as groupings of some of the values present in the homologies and some averages across homologies.

espece -French for species- is an object which contains specific information about given species. They are stored in a dictionary owned by PyCUB. Here as well, a set of functions related to the species have been implemented.

Important Packages

Among the numerous packages[5] that have been used, the most noticeable ones are:

[2] All of these objects make the architecture of the code as it was defined at the beginning of the project.

[3] In this architecture we are referring to the entropy values of the CUB as CUB values. It is because entropy is not the only way of measuring, storing and working on the CUB, we can have in PyCUB. We have created the objects with an open notion of parameters and CUB. It is because the code can use other measures of bias, other homologies, species and meta-data. This versatility was one of the requirements of the project.

[4] Overhaul the data is mostly stored as lists or Numpy arrays: Numpy arrays are better for storing big multidimensional arrays and to do computations over them. They are also much more memory efficient than python built-in lists.

[5] Other were listed as interesting packages A.1.5. Among these, some PyCUB should contain in the future. However they either required too much time to implement or their use only revealed itself at the end of the project.

- **Tool-Kits**:

  scikit-learn A package for machine learning and data science. Its reputation in the data science community is explained by its ease of use, reliability and simplicity. It allowed us to iterate across many clustering, regression and dimensionality reduction algorithms and infer the right one to use for different situations and data.

  Biopython A package that is a must when using python for biology and genomics. It integrates the API of the most famous Databases (DB)s and some of the necessary algorithms and data representations used across the community. It was mainly used here to interact with the fasta file format (Pearson and Lipman, 1988) . It may have been used to replace more function in PyCUB code.

- **Data Packages**:

  Pandas Helps to interact mostly with data represented in csv files and other tabular datasets, also called PAN(els) DA(ta).

  Scipy Package for scientific computation in python, to work on subjects ranging from complex matrix representation to probabilities.

  Numpy Mostly used for vector and matrix computations. It is a subset of the Scipy package. Together, these 3 packages are used across our code.

- **Visualisation**:

  matplotlib The most famous visualization toolkit in python. Inspired from Matlab plots, it is as easy to use and possesses very intricate mechanisms to guess how to plot different types of data. It is filled with options and functionalities. However, it is not interactive.

  bokeh A visualization package built on JavaScript to do interactive data representation by sending the information in a json from python to a small bokeh-built server. HTML files from the plot can also be exported.

  holoviews A toolkit for plotting, built on top of bokeh and matplotlib, which allows to plot complex datasets very easily, by providing ready made functions.

- **Other Tools**:

  ete2 A tool to interact with NCBI's API.

  functools32 A famous, default package in python, used to do all sorts of powerful tricks. Here, it was mostly used to speed up some computations by caching data that would take a lot of time to retrieve or to compute otherwise.

  pdb Similarly a default package, pdb is used to debug the code by setting traces and enabling a lot of debugging functionalities within the trace.

  rpy2 a python API to have R code called and run from within a python function. It is very useful when wanting to use packages which are only available in R, in python

- **Utilities**: **ftplib**, **urllib**, **joblib** are the preinstalled utilities to access ftp server, URL pages and to do parallel CPU computing. Other simple utilities packages were used such as: **os**, **json**, **shutil**, **gzip** ,**copy** and **requests**.

Chapter 2

Results

Here, I present the code with the different functions it uses, some remarks and the caveats I have found, while reflecting on it. Moreover, I am displaying my results and the tests I have set out in order to make sure my program is working. I also discuss the significance of my results and the assumptions I have made. Then, I interpret what the output might say and how this interpretation would need to be further discussed and tested.

2.1 The Code

You can find out more about the code from the code itself or its online documentation on github at `www.jkobject.com/PyCUB` . For more statistics and remarks about the code: `www.github.com/jkobject/PyCUB/blob/master/remarks.md`. More results can be found from `www.github.com/jkobject/PyCUB/blob/master/stories.md` and the notebooks.

2.1.1 Functions

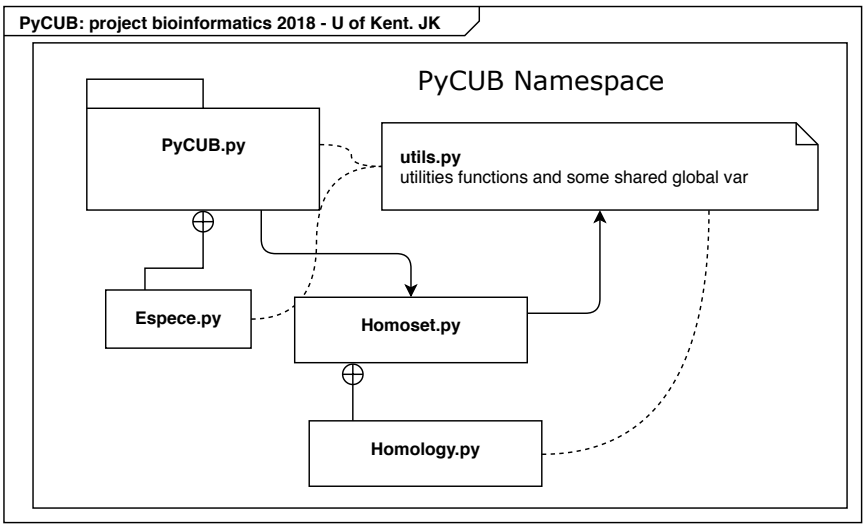


FIGURE 2.1: Relational diagram of PyCUB

Here are the different subset of functions with a summary of what they do[1]. If the reader is not familiar with the code, it is very much advised to have a look at it in order to comprehend the results and the work achieved here:

1. Loading and saving
2. Retrieving the metadata
3. Retrieving the data
4. Further data extraction
5. Computing the A_{value}
6. Functions on the homology object
7. Clustering of sets of homologies
8. Clustering of homologous genes
9. Plotting all the genes together
10. Functions on the homoset object
11. Final comparisons
12. Computing 3D genomic correlation

2.1.2 Remarks

When developing the software we noted some interesting behaviors related either to python or to some functions. It appears that:

- Because of its use of hyper optimized C functions, it is sometimes faster to run some un-optimized python functions relying heavily on C libraries (pandas, numpy, scipy...) than to try to optimize one's algorithm in python: "the pythonic way". Thus, it is very much required to tests one's ideas against others, e.g. with the python decorator *%timeit*.
- Python is able to display forever bigger or smaller numbers. However, Numpy with its own implementation of number representation, may render some very small values, e.g. inferior to 10^{-100}, as zeros, called overflowing. When drawing from the multinomial, we encountered the problem, preventing us from computing too big and too small numbers at one go. Here we had to use the ln exp trick.
- We could also have used this trick in *jerem_compute* and it may have improved its speed. We have also used non-linearities in other contexts such as when displaying the color gradients for different values, to highlight their differences better.
- We often used logs to display where the computation were. It really helps to make decisions about the computation and the code.

[1]For a clearer and lighter dissertation, the functions are presented in the appendix A.1.1 and can be further appreciated by looking at the documentation and the code.

- Paralleling to a high extent, e.g. up to 32 processes for 8 threads when doing parallel loading of the data is very efficient and increases the speed almost linearly as long as there is internet speed.
- The data is almost always unformatted and it makes it both very hard and time consuming to process and debug, e.g. when there is rare un-parsable strings.

2.1.3 Caveats

After reflecting on the code, some of the problems and weaknesses we have found are explained here:

- Unfortunately, the load/save functions, because of their architectures, do not scale well. The resulting json file can quickly get very big, when not zipped. In order to put everything as a json file, the dictionary takes a lot of memory, e.g. more than 20Gb for a resulting compressed file of 500Mb and 4Gb uncompressed.
- Another problem arising from the use of shared global variables in the *utils* namespace, is that memory leaks can happen with any code error followed by an *autoreload*: re-instantiating the global at their default values[2]. We had to create quick functions to load them back into memory (*loadspeciestable*, etc).
- Moreover, "jsonizing" data creates typing issues as json stores strings in unicode format which kept us converting back to python string format. Here we can see that the saving function is not the most efficient and shows how complicated it may be to save some complex objects using python. We have thought about serializing the data, using packages like pickle but it is not possible on python with the data we are using, as it would be on other languages like java and it would not let us export and share the data easily.
- *hyperlinkF2getmetadata* produces the problem of not being as automatized as the other parts of the pipeline, it only gets metadata for fungi and not all the species. It is because of the difficulties to find such information and under the same format. It is why this function compels anyone to code and to add a way of loading their own meta-data.
- In *getdata*, the function to compute CAI reference -and thus the entire CAI computation-, as for now only works for fungi.
- Moreover, sequences can contain many other letters apart from A,T,C,G, with letters representing combinations of possible nucleotides. When computing the codons, it may be very tricky to use such information. Removing the codons would be too big of a bias. We have thus decided to replace them by a random possible nucleotide.
- In *homoset clustering*, we might want to try different hyper-parameters for the homologies, to optimize over more metrics and use some regularizers as well. Moreover, **AIC** and **BIC** measures, or **silhouette** and **Calinski Harabasz** scores have not been used as much as we could, e.g. for hyperparameter search (Akaike, 2011; "Bayesian Information Criteria" 2008; Rousseeuw, 1987; Cengizler and Kerem, 2017). The first two are Bayesian information criteria. For

[2] As of today, we have not found any fixing of this problem.

the parameters we use AIC to remove low information ones and BIC to compare the models and choose the best one. In the silhouette, the best possible value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster. The last score is defined as a ratio between the within-cluster dispersion and the between-cluster dispersion.

- On the mathematical side, we did not manage to compute the complexity of the *computepartition_sorted_full* algorithm for $nbcod = 6$ -see logbook-. Here (**nbcod**) is the number of possible synonymous codons A.1.1.
- *homoset_all_plot*, cannot display information when considering the datashaders and heatmaps. However, in order to understand what the plots show, it is essential to have such information. Unfortunately we did not have the time resources to implement it.
- In *Final comparisons*, we could have used classifications instead of regression. It is true that because of noise and a high number of binary variables, it may be wiser to see whether a variable predicts a cluster instead of a regression.
- Finally, while recreating the *compute3DGD* function we had to face a lot of difficulties trying to understand what the paper was supposed to display about the exact way it was processing the data. With little information in the supplementary material, it was left for the reader to interpret, sometimes sparse explanations, about the inner working of the code[3]despite our efforts we were unable to find the code corresponding to the paper. We had to try 6 different algorithms before getting any significant results. The first difficulty was handling sequences. Many of them often had their centers, related to the same locus in the contact map. Should we average their values? Should we display them as being connected? If so, what distance should we use? 1 or 0? Should we also try to match them to the 200 values that only exist on the locus2+chrom2 part of the contact map [4]? It was also hard to comprehend when the binning should happen in the computation, and when the distances should be computed, as the explanation of the results was not following the processing of the data.

  Moreover, some caveats of the Endres-Shrindelin distance computation were not explained, e.g. what to do when there are zero values.

2.2 Datasets

We went to the EMBL database: ensEMBL[5], to get most of our data. We used two of their platforms, the **FTP** server containing most of their full genomes and their REST server, to which one can send complex requests to receive json and xml formatted replies.

Moreover the ensembl database has partially available tRNA information. It also has a much more diverse set of data available that might have been very helpful, such as ancestral sequences, age of sequences, protein annotations, regulatory features, etc.

[3],

[4]We purposefully kept some of our trials to comprehend the paper in the code. For the correction version however, only use "diament2".

[5]The fungi genomes in ensembl are under a secondary branch of the server called ensemblgenomes.

However, due to the compendium of species we were analyzing and the scarcity of the other types of information, we were constrained only to a few subset of the possibilities offered by ensembl.

We also took data from different research papers for the protein information and for the species information. The protein half-life, mRNA concentration and protein excretion come from a previous research[6] by (Haar, 2008).

Finally the 3D genome HiC data come from the data in supplementary information in (Duan et al., 2010) that Alon Diament also used and which were already pre-processed and pruned.

2.3 Assumptions

Reflecting on this project and on many ML and bio-science pieces of work, it seemed important for us to go over some of the facts we assumed to be true. Assumptions are necessary but may sometimes be overlooked in a desire to present a more compelling story. First we assumed that the data we had was true and unbiased. However from sequencing to alignment and post processing, there are many possibilities of getting erroneous values. We happened to find such values in ensembl data for example 3. We also admitted the different protein computations we had: protein cost, protein hydrophobicity, etc, A.1.1 were giving good proxy to the real protein data, we know however that this is more complex. We assumed that saccharomyces cerevisiae, by its role of reference species, could serve as a good proxy to a "common ancestor". A species for which some metadata for some genes could be used as reference for other genes in the homologies.

We assumed that replacing SNPs by precise base pairs would not bias our dataset. Then, we considered that the CAI reference genes can be used across homologous species, to create eCAI]eCAI references -see A.1.2- and that, on average, codons are equally distributed. We also assumed that the tRNA annotations, despite being incomplete, yielded the right amount of tRNA CN per species. Finally, we assumed that the pseudo phylogenetic distances we created from the phylogenetic dendrogram tree are somewhat related to an actual concept of evolutionary distance.

2.4 Outputs

Here I will discuss and display the different results. Some of them are related to the computations, others depend on the algorithms used and some other on biological results. I will explain them, present some compelling explanations and some possible questions regarding them. They will lead to possible further developments, pointed out in the next sections.

You can find more information about the results by looking at the notebooks, and find more results by yourselves by using them. Beware, some of the plots might have disappeared -jupyter notebook is far from perfect- you can however, visualize

[6] In the context of this research, we were interested in yeast's and related species' genomes. However, PyCUB works for any kingdom of life and can already compute the same values for plants and bacteria. The metadata will not be all available: one would have to find the HiC data for the other kingdoms and to find half-life, concentrations, etc, for homologies and relevant species metadata.

them in the *utils/savedplots* folder of the project. You can also have a look at more results in B.

It is true that several hypotheses can be made on the many different plots and one would require the functions of different proteins and more knowledge about the species. I am often referring to unknown latent factors which display the limit of our exploratory work, hoping that interested readers continue the investigation.

2.4.1 Computations

Loading the dataset from ensembl and computing all the necessary information took, on average, 90 minutes to run. Saving the full data took 5mn and required 20Gb of RAM for 5800 homologies and 510 species. The final files needing 3.5Gb uncompressed and 0.3Gb compressed.

Moreover, it takes 10mn and 16Gb of RAM to load the full homology matrix into the homoset object. In the end, there are 1,730,558 genes retrieved for 6,692 possible homologies, i.e. SC genes. The final number of valid homologies, i.e. ones with recorded related species, is 5,807. We retrieved 523 species with 15 being false positive.

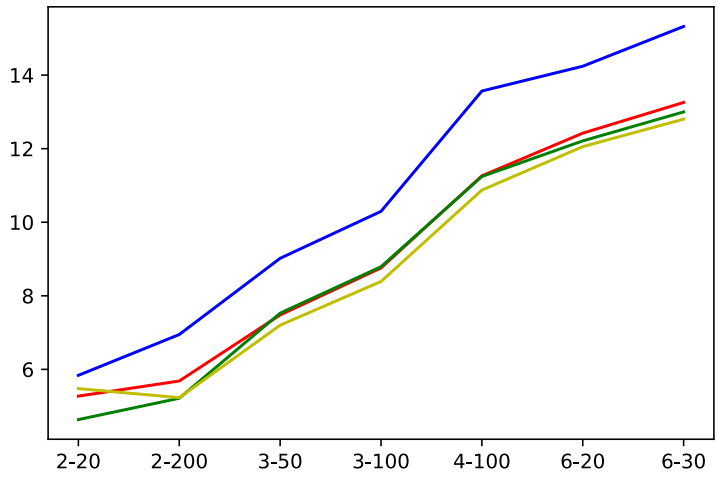


FIGURE 2.2: Logarithmic time scale, with x representing sequence length, nbcod

On the partition function, *computejerem* (G) is faster than any other regular partition function (R,B) and stays faster than even the *randomdraw* function until reaching approx 100,000,000 possibilities. We can see however that the normal approximation (Y) is more efficient for most of our values.

It is understood because of the way *computejerem* works, which removes many of the computations. It is however less of an increase than one might think, mostly because scipy's function is highly optimized and may use C code. The thought-to-be-faster partition function without permutation, was in fact always slower than the regular one. It may have been caused by this same discrepancy between scipy's pmf function and the permutation function we have implemented in raw python.

However, the complexity is technically still less for the partition function which does not compute the vector permutations.

All in all, it takes approximately 120 minutes for all 210^6 genes to compute the A_{value}, with possible partition function sizes of 10^{20} and us using the random draw function for partition size of more than 10^7. Globally, compared to the matlab function of Yun, we achieved hundreds of folds speed up, the computation taking an hour for a full kingdom instead of weeks for Yun's.

Finally, t-SNE, with its requirement to look at the full data distribution, is very memory inefficient when applied to a large dataset. We used 3 different versions. The parallelized one was more than 10x faster than the regular scikit-learn version (sota). We also found that the PCA vs t-SNE plots were really displaying different kinds of information and used both to visualize and understand our data in different ways.

2.4.2 The homology set

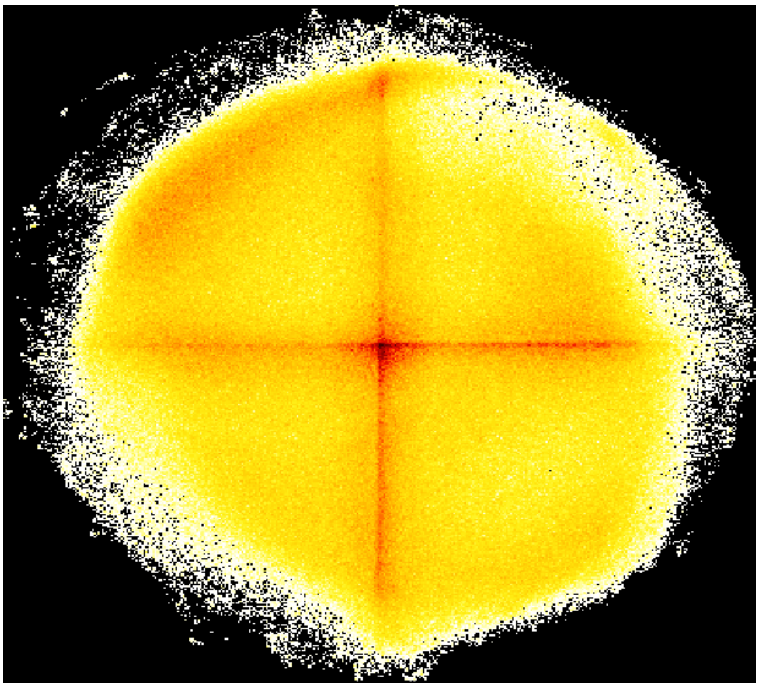


FIGURE 2.3: t-SNE revealed 4 denser areas, where the datapoints clustered. They seemed related at some extremities and then spanned further away from each other. The 4 clusters and the continuity is the most we can express on such a plot[7].

This figure was only revealed by t-SNE and it shows that, in the entire 5,800 homologies, there are 4 directions spanning on a continuous line of entropy variations where an important subset of the genes entropy values are distributed. It may mean that 4 distinct latent factors influence, on average, the CUB in a similar way.

Moreover, we found no shared homologies across all species in ensembl data. However, for such related species, it is known that some genes are shared across all of

[7] In accordance with t-SNE guidelines, we had different representations for different perplexities. However the plots were consistent across different trials. For example, plotting the entire set of homologies or only the one shared by most species led to the same plot. Changing the measure also led to a similar plot -with various amounts of noise- see B.1.

them even concerning humans as they encode for proteins that are necessary in any living organism. We mostly clustered the homologies into 3 to 4 groups getting pretty good clustering scores: silhouette score: 0.435136981254, calinski-harabaz score: 6397.88903542. Choosing the densest one, we were left with 3192 homologies comprising 1443683 genes. See B.2

2.4.3 Homologies

We found different results across the three measures of the CUB we used.

Most of the time, entropy defined clusters had smaller phylogenetic distances than the mean $70\% \pm 20\%$[8]. Even if, on average, the clustering scores were low and little related with phylogenetic distance scores, they were the best ones across measures.

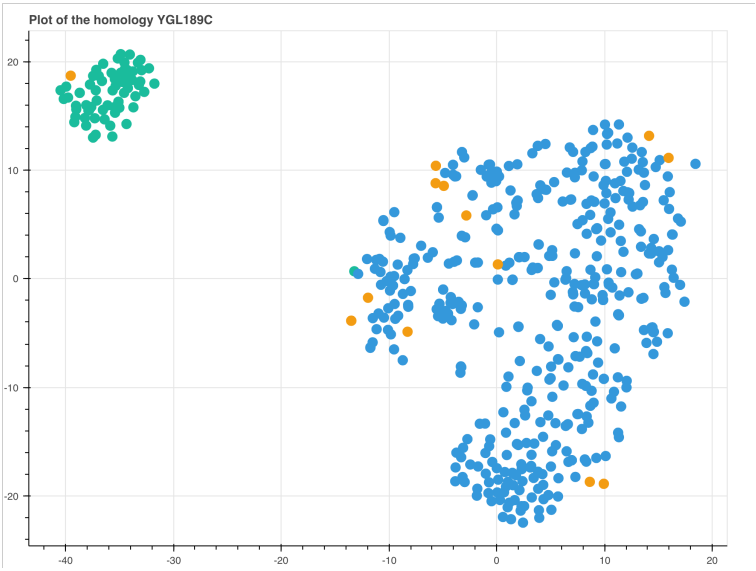


FIGURE 2.4: On the entropy values, we found many homologies 5% with clearly distinguishable clusters.[9]

The correlations to similarity scores 0.3 ± 0.2, Nan[10] 0.15 ± 0.15, CAI 0.25 ± 0.25 and GC content 0.30 ± 0.20 were very different among the homologies. B.12

We found a lower-than-expected correlation between eCAI and sequence similarity 0.48, meaning that they do not capture the same relations. This was shown throughout our pipeline.

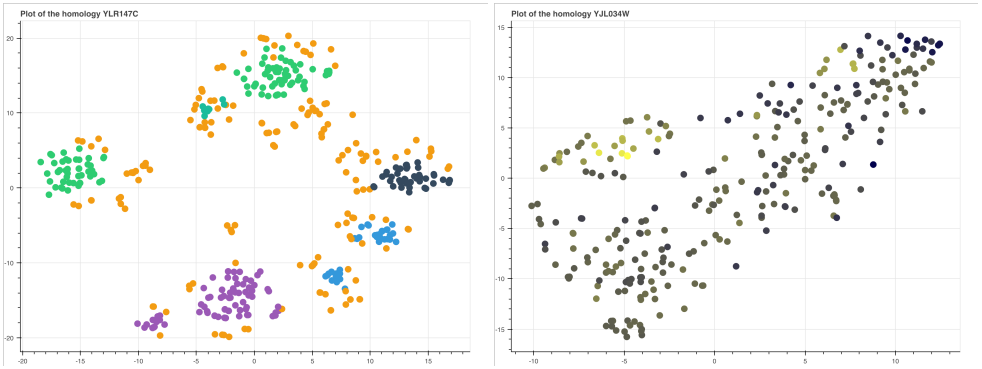
For homologies with more variance, the length began to be more related to the mean entropy 0.00 ± 0.50, We also found that the GC content had a more consistent impact across different homologies. It was also shown that low variance homologies have little correlation to these values.

A_{value}, however showed the best notion of clusters in any type of plots, using t-SNE. The clusters were numerous, however regular clustering metrics gave consistently lower score on this measure. For most homologies there seemed to be an equal

[8] with $p_{value} < 10^{-7}$, always.

[9] We could see that t-SNE permitted the appearance of non linear secondary clusters also picked by DBSCAN.

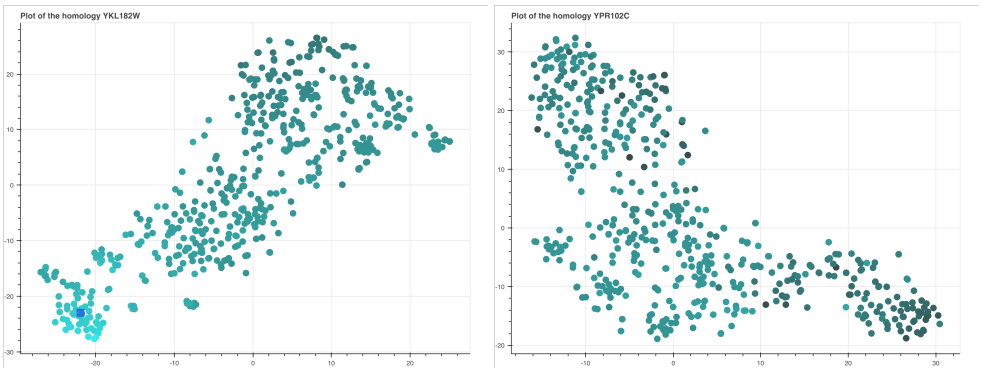
[10] We used the nan values and considered them as an information on the quality of the sequencing.



(A) Found from A_{value} (B) Found from entropy

FIGURE 2.5: Homologies, (A) with DBscan clustering and (B) with similarity to SC.

correlation of the value to nans and length, respectively -0.20 ± 0.20 , 0.10 ± 0.50 but less to any other given metadata. It also appeared that the manifold found by t-SNE was sometimes not related to any simple meta information we had. Contrary to the two other measures, the -somehow good- cluster found did not even relate to phylogenetic distances (phylo. dist.). See B.11 and B.14.



(A) From frequency (B) From high entropy homology

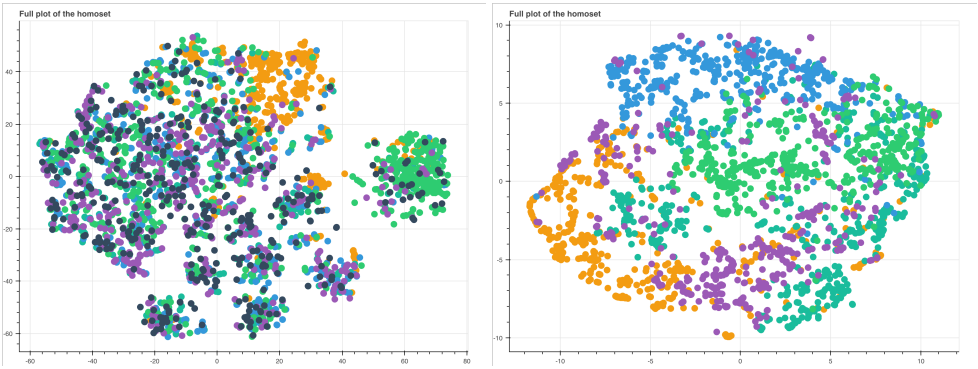
FIGURE 2.6: Two homologies, (A) correlated to CAI and (B) correlated to the GC content

Finally, for the frequency (synCUF) we witnessed how much it is linked to the CAI -0.9 and the GC content 0.9[11]. It only had low correlation to similarity and length and its clusters were only slightly related with phylo. dist. . We can also see some different homologies which had much more variances, up to the point of not being able to be clustered together. For them, the GC content almost entirely explains the 2D plot that we have. Most of the time, the manifold retrieved by t-SNE is very similar to the GC, eCAI, CAI variances[12] B.13.

By comparing groups of homologies together, we found interesting information as well:

[11]Here the correlation is against the Endres Schrindelin distance of the synCUF of each gene to the reference gene.

[12]It seemed, throughout this pipeline, using frequency, that it did not merely show as much information as the other ones. It can be explained because the pipeline was meant to be thinking about the types of general information that entropy could give: it both validates the efficiency of entropy as a measure and also validates this pipeline together with this project in its goals, intuitions or findings



(A) From A_{value} (B) From entropy

FIGURE 2.7: Multiple homologies, with colors representing distinct homologies

Even if some homologies can be spread out, homologous groups exist, hinting at some factors related to translation. With little relation to the mean entropy or length, more complex factors thus might explain it. Secreted, highly expressed homologies, seem to have a higher CUB. It follows the general theory about the CUB driven by gene expression. For entropy cost plots, there is again a better clustering of the datapoints, independently from homology or length. The 9 clearly defined clusters thus represent some other latent factors.

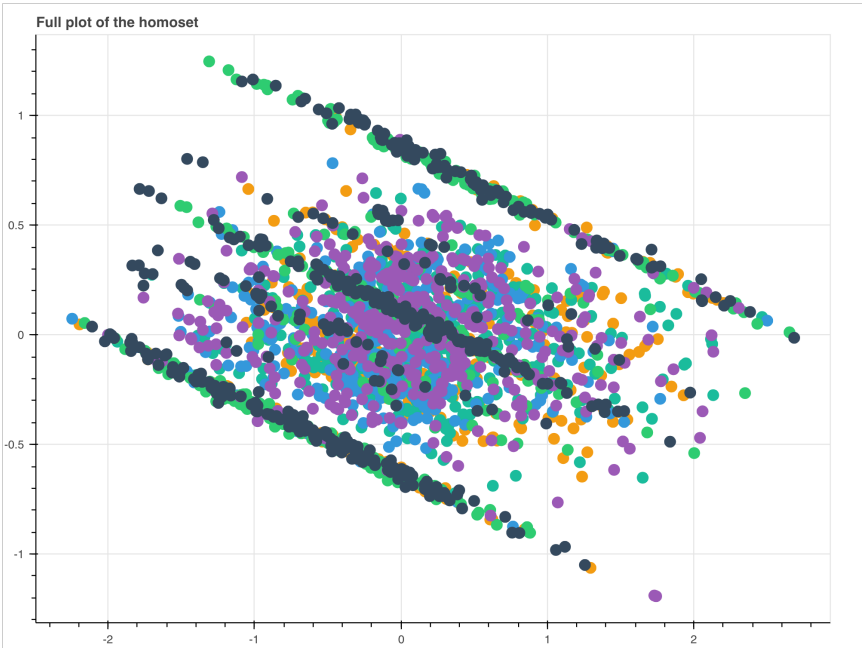


FIGURE 2.8: For frequencies we could see that there were some very interesting groups where only a few lines described the CUB of some homologies, e.g. ['YDL224C', 'YNL315C', 'YDL183C'] and the rest was clustered around the center.

The 2.8 shows there is two homologies and parts of other homologies which have most of their variance in only one direction, which could be some amino acid -since we were looking at a PCA linear reduction-. We were also able to see it in low variance homologies for other measurements. Some had a huge entropy for one

or two dimensions only, meaning that the entropy was only driven by mostly one amino acid.

Finally, when considering the distribution of clusters among all the homologies, we found one and only one species, called **Hepatospora Eriocheir**, a pathogen parasite of crabs (Ding et al., 2018), where most of its genes 70% were pertaining to the outlier cluster B.9. It means that the codon usage bias distribution of this species was largely higher than for any other. We found that many more species had a strong majority of their genes pertaining to outlier clusters or secondary clusters. However, most of them had only a few homologous genes to cerevisiae and a higher phylogenetic distance, which would hint at evolutionary drift more than evolutionary adaptation.

The clustering of the homologies was always pretty efficient. We had however high variances in frequency and entropy measures[13]. Thus, the global hyperparameters of DBSCAN were not adapted for 20% of the homologies, i.e. either too important or too small B.8. We overcame the difficulty by computing different hyperparameters for different groups of homologies with more or less variances -overhaul the change is not that important: ranging from 5 to 15 for entropy for example-.

2.4.4 3D correlations

When reproducing Pr Diament's work, we found different results which, if true, would diminish the hypothesis his paper make about the reasons of the correlation. The correlation of the 3D distance to the codon usage frequency (CUF) values were lower than the one of Alon Diament: 0.71 with d_{ES} and 0.6 with $d_{euclidean}$ with $p < 10^{-190}$. We found, however higher values for the synCUF correlation, of respectively 0.84 and 0.76 $p < 10^{-320}$. We were able to use the CUB and show a really low inverse correlation with it: -0.13. However, by using the average across each homology -thus without even using the values of SC- this correlation doubled for the CUB values, we also found CuF correlations of 0.76 and a correlation with the entropy cost of 0.44.

2.4.5 Measurements

We can see here that there is a strong correlation of the entropy to the number of synonymous codons. However, it is not enough to be able to classify it. For example PRO -line 14,4 codons- is less informative than LYS -line 12, 2 codons-

Moreover, one can see it has been decided not to normalize the entropy vectors. It is because of the bias this would inherently create[14].

[13] We have to note that we found homologies that were wrongfully considered as having a secondary cluster whilst it was their first cluster.

[14] We might want to compare the different values within the vectors, different vectors among homologies or different vectors among different homologies etc. Rescaling the vectors within themselves would not change anything when comparing the different values since we would still have the same discrepancy between different dimensions and we would not be able to compare different vectors. Rescaling all the vectors from within a homology would prevent us from comparing different vectors together.. etc. I.e, not rescaling here seems to be the better option -for any other variable in the regression step we pre-process and normalize all of them-.

entropy	A_{values}	frequencies
5.945 ± 2.766	0.945 ± 0.223	$0.212 \pm 0.121, 0.324 \pm 0.157, 0.188 \pm 0.129, 0.274 \pm 0.145$
7.551 ± 2.918	0.907 ± 0.289	$0.144 \pm 0.119, 0.254 \pm 0.185, 0.118 \pm 0.104, 0.169 \pm 0.135, 0.121 \pm 0.105, 0.192 \pm 0.202,$
2.210 ± 1.892	0.747 ± 0.315	$0.625 \pm 0.224, 0.374 \pm 0.224$
2.118 ± 1.963	0.753 ± 0.317	$0.470 \pm 0.203, 0.529 \pm 0.203$
0.747 ± 1.077	0.532 ± 0.370	$0.415 \pm 0.298, 0.584 \pm 0.298$
1.996 ± 1.838	0.722 ± 0.318	$0.439 \pm 0.238, 0.560 \pm 0.238$
3.071 ± 2.276	0.750 ± 0.314	$0.574 \pm 0.219, 0.425 \pm 0.219$
6.802 ± 2.883	0.948 ± 0.225	$0.293 \pm 0.179, 0.131 \pm 0.111, 0.217 \pm 0.131, 0.357 \pm 0.186$
1.173 ± 1.361	0.701 ± 0.334	$0.429 \pm 0.261, 0.570 \pm 0.261$
5.884 ± 2.555	0.932 ± 0.213	$0.524 \pm 0.209, 0.132 \pm 0.133, 0.343 \pm 0.163$
12.639 ± 3.693	0.936 ± 0.244	$0.084 \pm 0.127, 0.192 \pm 0.128, 0.276 \pm 0.166, 0.162 \pm 0.09, 0.207 \pm 0.130, 0.076 \pm 0.066$
4.184 ± 2.525	0.791 ± 0.311	$0.329 \pm 0.213, 0.670 \pm 0.213$
1.999 ± 1.743	0.772 ± 0.317	$0.369 \pm 0.212, 0.630 \pm 0.212$
3.985 ± 2.246	0.924 ± 0.261	$0.262 \pm 0.147, 0.197 \pm 0.146, 0.256 \pm 0.182, 0.283 \pm 0.181$
6.067 ± 2.604	0.927 ± 0.265	$0.183 \pm 0.115, 0.174 \pm 0.114, 0.141 \pm 0.104, 0.210 \pm 0.125, 0.178 \pm 0.107, 0.112 \pm 0.084$
4.285 ± 2.267	0.940 ± 0.236	$0.329 \pm 0.174, 0.220 \pm 0.140, 0.214 \pm 0.151, 0.235 \pm 0.145$
1.636 ± 1.592	0.686 ± 0.322	$0.380 \pm 0.235, 0.619 \pm 0.235$
6.399 ± 2.668	0.958 ± 0.220	$0.115 \pm 0.100, 0.376 \pm 0.180, 0.245 \pm 0.137, 0.262 \pm 0.152$

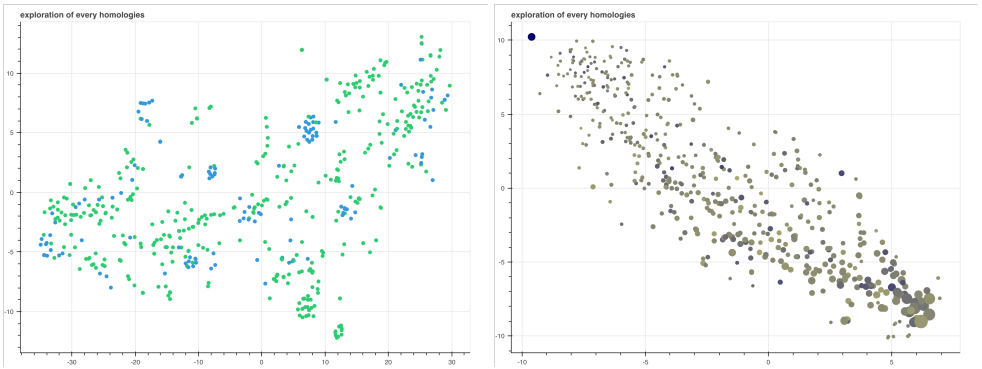
FIGURE 2.9: table representing the mean±variance values for the 3 different measures. The amino acids are listed alphabetically. -see www.jkobject.com/PyCUB- for more material

We have also found some different values for the subset (*working homoset*) of highly-conserved homoset, even if low such difference is of statistical significance given the size of the set.

Some very high entropy values were recorded, with the lowest aa at 52.940 (CYS) and highest at 703.439 (LEU). The levels are really high. One can understand such values as requiring 703 bits of memory to encode if the encoding was defined on the assumption of an equal unbiased multinomial distribution. The distribution shows that the expected values, at least for the set of fungi species we got, are definitely not symmetric. We can also see that some amino acids are very little pushed to an unusual entropy value.

2.4.6 Final outputs

WHEN CONSIDERING THE AVERAGE CUB VALUES FOR THE SPECIES: their entropy all seem to go toward one high entropy and relatively high variance direction, which PCA also helped us to see. This is consistent across measures and for homologies as well B.10.



(A) From frequency (B) From entropy

FIGURE 2.10: Plots on species CUB average from homology. (A) with blue color for animal pathogenicity and (B) with the mean GC content over the entire coding sequences.

However, not much information could be extracted for species plot for the entropy or A_{value}. We still found some correlation with the number of genes -0.2, average size of homologous genes: 0.24 and the total GC content of the DNA sequence of the species 0.26[15].

Moreover, on entropy measures the species with the highest variances seem to be all surrounding the one with less variance in their CUB B.15. For frequencies however, the plots revealed that clusters of species could really be seen. They mostly represented sub-genres that are highly related, which also made the plot cluster well the metadata. Here it is most surely not a causation.

TRYING TO REGRESS ON THE SPECIES METADATA: we found little evidence of correlation. Lasso had a lot of difficulties removing uninformative parameters for the entropy values. However, increasing the training size and removing some uncorrelated data increased a lot the efficiency of the algorithm: from an r^2 of $-1,000$ to one of 0.

[15] $p_{values} < 10^{-6}$

Finally, we achieved a significant score, with A_{value}, which was predicted given the correlation we found for it. Here, Lasso was able to remove the uninformative parameters and achieved an r^2 of 0.255. However low, it could be the maximum given the training size and number of uninformative and low correlated parameters.

LOOKING AT THE PLOT SHOWING AVERAGE CUB VALUES ACROSS HOMOLOGIES: we were able to see some CUB distribution correlations: to the similarity scores, with the GC content and with the length. Again, there seem to be clusters of secreted proteins and a relation between highly regulated/lengthy/high-variance/high-entropy homologies. We can also see a relationship which is quite visible on the plot[16]. Evo-

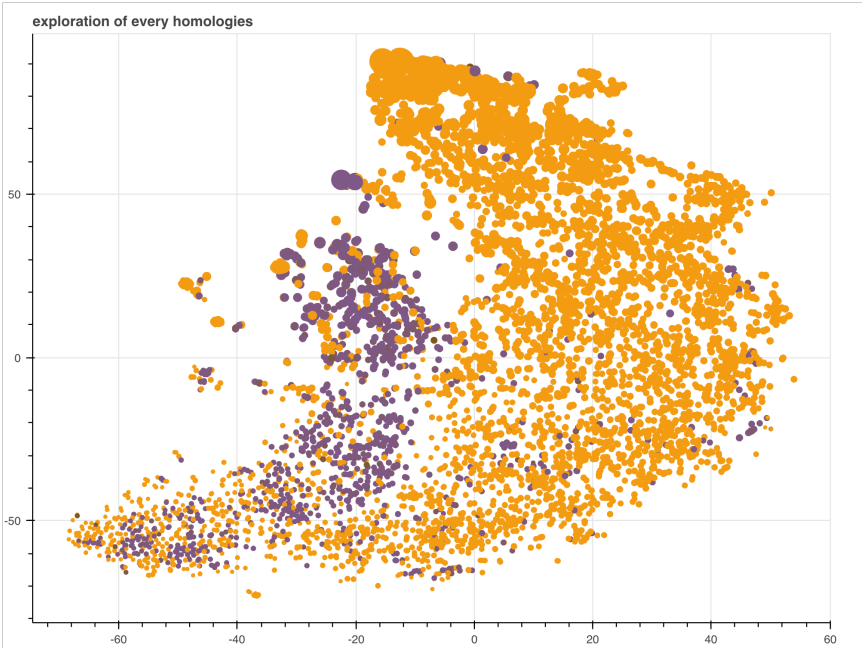


FIGURE 2.11: It seems that all conserved homologies are mostly present on one part of the plot and the other ones, called recent homologies, on the other part[17]. This bi-clustering is only present using t-SNE as it finds relative distances and not only high-variance directions.

lutionary CAI also seems to be highly predictive of this shape. We can see that low entropy homologies and high-entropy/high-variance/conserved homologies are at the opposite of the spectrum with degradation between the two (which is not the case of the similarity values).

Using PCA on the A_{values} plot, we have also found that secreted homologies seem to be more clustered than non secreted ones. High mRNA abundances seemed to be mostly clustered on the outside of the plot, and some low entropy homologies also seemed to be driven by the nan values, i.e. the quality of sequencing. Finally, similarity score seemed to be influenced by the length -which is not trivial-.

WHEN TRYING TO REGRESS ON HOMOLOGIES: we found some important correlations that were, for most of them, related to what we saw -with different degrees

[16] Because of a number of outlier values and time constraint, we were not able to assess well the impact of the distance to tRNA usage bias values; the GC difference between full sets of genes and the homologous ones and the impact of the tRNA CN

[17] We have found,not to our surprise, that the working homoset is basically just one of them (the "highly conserved genes".

given that one is an average and the other is a manifold retrieved by t-SNE-. Respectively for entropy and A_{value}: nans -0.35 and -0.67, length[18] 0.80, 0.61, GC count 0.22,0.10, protein abundance 0.18,null, protein half life 0.14,0.15, mean hydrophobicity 0.11, 0.12. We found additional correlations with entropy such as whether the final protein is secreted or not 0.09 or the gene number of cys-regulatory elements 0.13. Additionally, we found that A_{value} was also correlated with the mean CAI -0.11 and eCAI -0.32[19]. Some of this relations were much more clearly defined when looking at the 2D dimensionality reduced plot instead of the homologies.

Then we found that the lasso was also able to regress as well as on the species given the chosen measure. On entropy cost it was better able to infer which parameters to remove. However, when feeding an MLP neural network with handpicked parameters -considering the relationships we found on the plot and the Spearman correlations-, we achieved unprecedented results. This 2-layer 22-neuron model was able to achieve an r^2 of 0.65 ± 0.10 -depending on the random weight initialization-. We found that the nan values were of extreme importance to this NN. Removing them got you back to an r^2 of 0.28 ± 0.07.

2.4.7 Other remarks

- during debugging when looking at the unknown nucleotide, i.e. polymorphisms -also called SNPs-, we could see islands of genes with a lot of recorded SNPs. It is often the case in biology for some genes to possess a higher number of SNPs (Rafalski, 2002).
- We found many lengths that did not seem plausible -some homologies with average length of 43- and 3% of genes in homologies with aberrant lengths. Some being more than 10 times shorter than the regular ones. They probably came from the alignment which finds homologous genes that are not used anymore, 90% of which has been removed during evolutionary events.
- We added random homologies to see if our pipeline made sense. It is obvious that the entropy would give 0. It would also give frequencies and averages variations with respect to the size of the fake gene. Using random entropy values and testing our clustering algorithms and dimensionality reduction algorithms, we were really able to infer and see the differences with respect to a real world dataset. Moreover, the randomness showed better using t-SNE, while PCA could catch one more noise. This can be explained as t-SNE is based on the theory of probability.
- The phylogenetic distances computed were mostly in concordance with the species we had. However the precision was low and even lower when we spanned further away from highly related species.

[18] We also found high correlation to values computed from the proteins by summation, but the bias to the length was too important: We could not think of ways of removing this length effect without biasing the values in some other ways.

[19] $p_{values} < 10^{-12}$: all p values and exact values can be found in the project and are linked in the supplementary materials.

Chapter 3

Discussion

The low correlations compared to the very clear ones sometimes given by t-SNE plots was surely because of the way they looked at the distribution. In the future, we should consider scores on the reduced dataset instead.

The results tend to confirm the idea that entropy and A_{value}, pick on different informations, i.e. remove different informations. Moreover, there are other latent factors than those provided to us. We postulate that even if the information seems more scrambled in A_{value}, it might also remove some trivial relationships such as those we find with the frequencies. All in all, it is sure that an important part of the variation in CUB values can be described by the cumulative effect of length, GC content and other latent factors shown by CAI etc...

t-SNE seems to retrieve the 2D manifold representing the direction which most expresses the impact of CAI, eCAI in frequency measures. We think that the found clusters might also be very much predictive of latent factors related to ancestral events which could have influenced the CUB of sets of species for a commonly shared gene. It would mean, for example, that some evolutionary events might have changed the use of the protein for the subset of species.

Moreover, we found by overlaying previous knowledge about the CUB, that different clusters of homologous genes are influenced by different drivers. Plots, -such as those representing lines where the entire CUB values are defined over just one dimension-, show that there are some homologies requiring that only some aa can have their CU modified. It may be related to the specificities of the encoded protein. The protein many not possess different aas or may be related to the secondary conformation of the protein. Different uses of codons for a given aa might lead to a different conformation of the protein. Here, a protein would thus have constraints on the CUB of some of its amino acids. This would explain why we have 3 different lines, related to 3 different conformations, however, given the small length of the protein this seems hard to believe.

We have been able to see how entropy cost presents itself as a good way of normalizing the data. We think that such normalization helped both t-SNE, DBscan and our MLP infer more about the CUB compared to the pure entropy value.

We have also been able to relate our measures to many of the known factors influencing the CUB. It both proves the great measures that entropy and entropy cost may be, also validating the selection and usage of the different hyperparameters that we used. Retrieving correlations discovered over many years by multiple research groups using purely computational means is not trivial. It also shows what can be gained from using homologies. Even when using many species that have

lived in different environments for hundreds of thousands of years, we were able to link them to the measures and metadata that mostly belonged to SC without even having to use SC's genome.

After looking at the results from our MLP regression and its heavy use of nans we postulate that, far from being a problem, it shows that the accuracy of unreliable components is used by the neural network to consider which parameters to use according to such level of accuracy. It shows how unreliable components affect the CUB that we are investigating. The neural network is not picking on nans to predict the CUB, else Lasso -which is also using nans- would be able to better regress on the values as well. We think we might be able to have a better look at their influence on our model by plotting the error of regression against it with and without the nans, to see how they influence the regression.

All in all, as guessed earlier with the plottings, it is confirmed for regression that the entropy cost is better than regular entropy. Moreover, the NN is way better than the lasso on this measure, highlighting the possible non linear effects the A_{value} might display. Moreover, as always, removing uninformative parameters was very important for the regression of the MLP.

About the correlation to 3D genomic distances, there might be some unexplained processes that I have not accounted for in my code, but it would not explain the results. They might be explained, however, by differences in the datasets: I am using ensembl genome while Diament's correlations were using NCBI one. Another explanation might look at slight differences of computation: I am only using codons with at least 1 synonymous codon for the synCUF computation and don't compute 0 values when using Endres Schrindlelin distances -instead of replacing them by the mean frequency, as it is done in some cases for example-. Finally it might be a mistake in Diament's work. However it is very hypothetical and far fetched for a research of this quality. The real problem is that it cannot be investigated as long as the code is not available.

Considering our results, we would also like to postulate what we think might explain them: that gene functions may be really diverse and are often related to a plethora of effects. It may not be trivial to assume that genes for which proteins have similar functions should be co-localized. However it seems important, for organisms which are under evolutionary pressure for replication speed, such as SC, that the genes with similar CUB bias should be co-localized. This may be because the tRNA pool, due to unknown processes, does not possess the same density of different tRNAs across the Rough endoplasmic reticulum (RER), which creates a pressure for the genome to relate genes using more particular tRNAs to regions of the nuclei matching such region in the RER. It may in turn increase the tRNA frequency in this location and the pressure for the CUB co-localization. The high results we achieved with the averages over homologies show that pressure effects over the core gene function might really explain their co-localization. This would need more investigation but we deem that the research direction that Pr. Diament postulated might need more proofs.

Finally, from our difficulties to find tRNA values, from the length errors in what is probably ancestral genes wrongly considered as homologies, the presence of NaNs, etc, we assume that ensembl database does not provide fully reliable data.

We were not able to postulate on what the heatmaps might show. We felt the need for further research 4.2 to get a better view of the phenomenons at play.

Chapter 4

Conclusion

We have presented PyCUB, a comparative genomics pipeline in Python for the statistical exploration of the Codon Usage Bias. We have introduced the questions and ideas related to Data Science today and presented the CUB and its many different problematics. In this work we have shown how to explore the CUB and described the tools we used. We have also demonstrated new metrics to explore this genetic motif and shown we were able to retrieve most of the research done on what supports the CUB. We have retrieved quite a few correlations that were found and described in the information research literature. We have used a compendium of species from the fungi kingdom in the EMBL database and found many caveats and errors in the data offered to researchers, which had not been described at any stage in their documentation. We have also been able to extract metadata and metrics and to create our own metric on the CUB. It was also proved to be efficient on our dataset. Finally, we have also shown different and new results from those in Pr. Diament's work. We have given possible explanations about the results we have found and raised new questions that might be answered by using our software and some additional information. However this pipeline is a task in progress and much work and debugging are left to do A.1.1. An urgent need of investigating the correlations we encountered is left to be satisfied. More directions are presented in the future research section: 4.2.

4.1 Evaluation

One of the difficulties of this project was its interdisciplinarity and its broad goals. No clear cut objectives were defined and I was not asked to set one or else it would not have been a "true data exploration". Despite my love for bioscience, I had here my first encounter with high volumes of biological data from many places. It was an interesting, although complex experience. Biological data exploration requires an interdisciplinary knowledge to yield good results and I have received noticeable help about the biological aspect of my work from Tobias Van der Harr, Yun Deng and Baptiste Tesson. Another difficulty was the low reliability of the datasets. Most of them would differ, contain false values, outliers, nan values, etc.

On the whole, these problems were mostly resolved.

All in all, this project has achieved its goals. It has managed to be a reliable data science software, proving the reproducibility of computational genomic research and proved the assumptions and the results obtained. Its high amount of documentation and versatility will enable anyone to build upon it, achieving the goal of reusability.

With 30% of parallel functions, and the work on function complexity and memory usage. It partially achieves its goal of scalability. It also allows for a lot of interpretability with all of its 11 plots being interactive, with explanatory jupyter notebooks and a function verbosity. Using HDF5 would have allowed the code to be performed on more complex data sets than fungi e.g. vertebrate or plants. The architecture used makes PyCUB pretty straightforward and allows for many research questions to be investigated. Moreover, the partition functions have allowed me to explore High Performance Computing (HPC) and programming issues on python, as well as the many tricks to improve efficiency. It was a great exploration of computational complexity.

We have globally achieved our goal of exploring the CUB. Even if exploration is not an enterprise that one can complete. This first experiment has yielded a lot of interesting information about the CUB and might reveal more with further computation, time and data. The results go in accordance with the quite general Mutation-selection-drift balance framework and they shed a new light over the CUB, powered by the massive dataset of ENSEMBL. Finally it stood up for providing information about the usefulness of the entropy and A_{value} as metrics for the CUB.

While trying to present and explain my task, I kept hoping that this thesis would be a good introduction to the subject for anyone interested in it, as well as a valuable feedback to any graduate student wishing to continue the research or to build his own comparative genomics pipeline.

Finally, this thesis gave me many new questions and insights that I would not have fancied if I had only coded and iterated on datasets. I was given the proof that writing and organizing one's own thoughts or finding one's questions and reflecting over them for many days can be tremendously helpful. Some of the numerous new questions that happened to arise then, together with some potential ways of tackling them, are given next.

4.2 Future Research

Here is an overview of where some ideas about the project could be led by a student or by any interested researcher in the future.

For now, this code has only meant to explore the codon usage bias in an evolutionary comparative genetics framework. However, specific work could be undertaken to bring such software to the next version: First, one should retrieve ensembl code to compute ancestral sequences from homologies and use them instead of those of the reference species. Then one should add the HDF5 abilities and scale up PyCUB to larger sets of data. So doing, one should render all the necessary functions parallel (we have found that up to 80% of the functions can be easily parallelized). When using HDF5, one might look for a better way of loading and saving the data, maybe by using the hierarchical file system. Lastly, it would be very interesting to create a web version for researchers to look at the homology plots and add their own metadata to answer questions about particular genes or set of genes. One would precompute all the homologies and just present them on request.

About the more biologically related research questions: they could explore the evolution of the species more deeply, the distinct secondary cluster and their stories, and look at the protein function of the homologies. One question can also be put

forth: What would the correlation be if I were to use entropy value given a non symmetric probability vector? This question is the same for the 3D genomics correlation. It may retrieve even more or less correlation maybe with a distinct distribution. To investigate results even further, it would be needed to relate tRNA presence to the 3D genomic distance. One way of doing so would be to look at species with different tRNA density and compare their 3D distances/CUB correlation to it. Considering the results obtained, it seems important that one should contact Pr. Diament and investigate this issue further by looking at the possible reasons for such a difference in the results.

About the unreliability of Ensembl, we should try to look at known protein functions that are known to come from a common ancestor and to be conserved across all species, to see the extent of ensembl homology reliability. It might also be necessary to investigate better ways of finding homologous relationships. It might be interesting to have a look at what factors influence the accuracy of such algorithm, to be able to account for it (Pervez et al., 2014). More data, e.g. a mix of DNA sequence alignment, protein alignment, RNA seq and taxonomic information, might help greatly to improve the accuracy of such software.

Finally, to understand more about the strange densities in the full plot of genes, one should have a look at some averages across the bins of the histogram used to create the heatmap. These averages might give information about what might explain this distribution. Spearman might be appropriate here to understand how correlated each average is with the mean values of the heatmap. Using phylogenetic relationships from species might also help at the single gene level here and on homology and multi homology plots, to understand the clusters. Lastly, the different clusters in homology and multi homology plots might very well be related to the environment of the species. It would be interesting to look at more species specific information at the single gene level. Different parasitic species often behave very differently than the rest of fungi, as we could see with Hepatospora Eriocheir. It could also be interesting to look deeper at specific parasitic species and compare them to the rest.

For now the clustering in homology uses pre-existing functions which are really reliable. However, it may be interesting to code one's own clustering algorithm. Additionally, using classification instead of regression on the final computation functions should be interesting. It seems really difficult to regress on the CUB values. Because of that, we had to regress on the mean CUB value, which removed a lot of information and made it even more complicated for linear models to pick on interesting data. Classification would make it easier and maybe more insightful. It would let the use of more explanatory linear models over the full distribution.

Lastly, more trials should be carried out with the already built pipeline. Using the plants and bacteria kingdoms could allow to see whether the assumptions made for the CUB of fungi are similar to those of plants and what could explain the dissimilarities between them both. With a future version of the code it might even be possible to use it for the vertebrate kingdom and further the generalities of the retrieved relationships.

Appendix A

Additional Information

A.1 The code

The code documentation is available at www.jkobject.com/PyCUB

A.1.1 Functions

Load/save *load, __init__, dictify, undictify, loadspeciestable, savespeciestable*

To be able to switch between sessions and save objects that might take a lot of resources to compute, we had to get the loading and saving of our data. To enable it, we have created the loading and saving of the entire session as a **gzip**ped json file. The "jsonize" function requires dictionaries and regular python objects only. It is one of the flaws of the current approach. Another solution would have been to save everything as its own object and each object as its own folder of objects, etc. It was not applied here, as many values are small and as it would not be a practical way of sending the datasets and exploring them among other algorithms. Here, each function has a *dictify* method, putting all its parameters as python objects into a dictionary that can then be jsonized. Objects are instantiated on loading by sending the sub dictionaries as a *kwargs* to their constructors.

getmetadata *getHomologylist, get_metadata_Ensembl, get_mymetadata, import_metadataTobias, getyun, mymeta, retrievenames*

Functions that retrieve data which are not sequences themselves, i.e. metadata, are called metadata functions. The first one retrieves every possible protein coding gene ID for a given species -IDs are used to find corresponding homologies-. The other one will download meta-data files directly from ensembl for a given kingdom. *get_mymetadata* is a function to be overloaded with anyone's metadata, to load them from a url. For now it retrieves ours. Then there are functions loading the meta-data to their corresponding species and homologies, e.g. *import_metadataTobias*, *mymeta*. *retrievenames* and *getYun* are specific functions to load Yun's metadata, e.g. to retrieve the name-order-links of all the species Yun has. For fungi, there are basically two ways of retrieving the same kind of data. You can load directly Yun's data or reload them on your own from ensembl by using *getdata*.

getdata *createRefCAI,loadfromensembl, preprocessing, get_tRNAcopy, loadfromensembl, process, compute_meta, reference_index, computeCAI, computeECAI, computeyun, getdata*

For the functions that retrieve data directly from the sequence, we have the central one, calling every other function -and some of them in parallel-. If it is not provided with a homology list, it will be found on its own from the metadata. Then it will require to create the reference set of genes in order to compute the CAI -explained earlier-. *loadfromensembl* function will send **REST** requests to ensembl and extract the necessary information with *process*. It will then return an instantiated homology object. We can order the *speciesname* list alphabetically to help in further pre-processings and by doing so we need to order every other field that contains information on the species such as CUB values, CAIs etc... Which is also done here. *process* will pre-process the sequences, and retrieve meta information about them such as *taxons, proteinids, species, geneids, similarities,* ***KaKs_Scores*** (Yang and Bielawski, 2009) and possible nan values or other outlier values. From the sequence, we were able to compute the GC content, the length of each synonymous codon sequence, the CAI and another measure called eCAI. With the sequences, *computeYun* and *computeMeta* can then be called. The first one will compute the entropy -see 1.3-. The second one will compute the reference protein chemical information (*synthesis steps, glucose cost, hydrophobicity, volume, isoelectric point, conservation*) using the information from the codons, i.e. future amino acids of the proteins. It is computed by simple summations and averagings. However, a particular equation is used for the isoelectric point as described in (Kozlowski, 2016) Finally, this function is also able to compute the A_{value} instead, but doing so here is not the most efficient and we will see next what is the best way to do it to speed up this computationally heavy measure. Once every homology has been created, one can find the full set of homologous species and encode them into integers -to save space- and store the encoding into a global dictionary *speciestable*. This step is called *preprocessing*. *getdata* can then create -the object of- every species and compute its tRNA copy numbers by calling the *gettRNA* function. Using the REST API of ensembl it will try to match tRNAs to their respective codons by parsing the annotations and compute its CUB value -by considering each tRNA codon match as a codon in a genetic sequence- if the statistic is sufficient. Else, ensembl also provides an expected number of tRNAs for the species.

further data extraction *get_evolutionary_distance, compute_ages, compute_averages, get_full_genomes , _compute_full_entropy*

Once everything is loaded, it is also possible to extract more out of the information we have. By using the taxonomy codes, we ping NCBI's database to retrieve a phylogenetic tree. Then We use an R function to parse this dendrogram tree and create a distance matrix, thanks to the **cophenetic** matrix function -used to compute pseudo distances from dendrograms-. It is then returned to the python code as a dataframe object, which is accessible as a shared variable called *phylo_distances* (Rohlf and Fisher, 1968). From this matrix we can also compute the "ages" of each homology. Here we use two assumptions: The first one supposes that if the sequences are highly similar and shared among most of the species, it is a highly conserved homology. The second one assumes that if the sequences are shared among only a subgroup of

species and this subgroup has its average phylogenetic distance that is only 80% of the total average, then the sequence is recent and its measure is this percentage. Another very simple but important set of information that can be extracted is the averages and variances of different values across homologies and species: *compute_averages* here. Finally, for each species we are also able to compute its average entropy and GC content for all its genes instead of only its homologous genes to *cerevisiae*.

Getloc *computepartition, randomdraw, compute_entropyloc , getloc, computepartition_sorted_full, computepartition_without_permutation, computejerem, computepartition_normal_approximation, mlen, permute, last_match_index*

Here we have a look at the computation of the A_{value} -abusively called **entropy-location**-. It can be computed by calling *getloc* from the *getData* function which will pass the length matrix and the entropy matrix of its current homology or, by calling it from a homoset object, which will pass the entire concatenated matrices of CUB values and length values 1.5. One can see that it is fairly straightforward, as long as one has the partition function of the multinomial distribution parameterized by the length, the number of possible synonymous codons (nbcod) and the probability vector. Here, we can use a symmetric probability vector which assumes that each synonymous codon has the same probability of appearing as any other. We can also use unequal probabilities, however this will constrain the available partition functions.

It is much more efficient to use the *compute_entropyloc* function of homoset as it holds the fact that many CUB values have the same partition function, not to recompute a partition twice. On getdata, it will only know about the ones of the current homology, limiting the trick.

Much of the other functions are here to compute the partition. There are four possible algorithms that have been derived to do it as fast as possible. The first one is the simplest and will compute every possible distribution of dimension nbcod and repetition and compute the probability mass function (**pmf**) of each of them. The second will only compute distributions for which their values are not permutations of any other[1]. It will then duplicate the pmf values by the number of possible permutations of each vector -using a recursive permutation function-. The third one: *computejerem*, uses dynamic programming and recursive functions to take advantage of the repetition in the computation of the multinomial, and thus recompute the least possible for each following pmf, It also uses the repetitions from symmetric probability vectors in multinomial distributions and thus does not work for asymmetric ones. The last one takes advantage of the fact that a multinomial distribution can be approximated by a **multivariate normal distribution** (Carter, 2002; Do, 2008). It describes the computation under the continuity assumption -which will get more precise as the length increases-. For more information about speed: 2.4.

Finally, when the computation gets too high, e.g. more than a few million samples, where the number of samples is $leng^{nbcod}$, the *random_draw* function is used. It creates a subset of the available partitions vectors by sampling randomly from the partition. It also uses dynamic programming and coding tricks, to speed it up. For example, when we only increase the length by X

[1] In the context of symmetric probability vectors permutations will make the same pmf.

-$X < 10$- and not the nbcod, we randomly add this length increase to each vector[2] in the partition vector array.

homology *reduce_dim, plot, clusterize_*

Each object has its own functions. Homology -which exists in homoset- has 3 particularly useful ones: *reduce_dim*, which will use the CUB values matrix and a choice of two algorithms to reduce the number of features, from X, with $X > 2$, e.g. $X = 18$ for entropy or $X = 59$ for frequency, to Y, with $Y < X$, here $Y = 2, 3, 4$, to be able to display the values on 2D/3D plots. The first algorithm is called Principal Component Analysis (**PCA**) and will multiply the matrix by the Y first eigenvectors (Shlens, 2014) of the matrix. It is a very famous linear algorithm. It is well-known, which makes biologists trust its results. The second is **t-SNE** for t-stochastic near embeddings. It is an optimization method whose goal is to minimize the Kulback Leilbler (**KL**) divergence -which is a famous measure of divergence between any two distributions- between the data distribution in X dimensions and the one in Y dimensions. Compared to PCA, this method is non linear and thus, does not conserve a Euclidean notion of distances[3]. However, it will be able to display information about the data which is impossible for PCA to retrieve. t-SNE is very much used in the ML community because of this fact and its ability to find clusters of data even in highly non linear spaces (Filion, 2018). Having these two highly different views over high dimensional datasets is the best way of exploring it and understanding the two techniques.

Then, *clusterize_* is the function to cluster the homologous CUB values. Here we want to extract as much information from it as possible. Clustering enables us to find groups in the dataset that are similar. For that, we use two clustering methods: **DBSCAN** and **gaussian mixtures** (Reynolds, 2015). We don't know what the shape of the data is, but we assume that there may be one or many clusters which regroup most of the points and a group of disparate outliers "going different ways", i.e with some dimension of their CUB vector being higher than most. DBSCAN creates n-spheres (spheres of n dimension) of radius '*eps*' around each data point. If a group of overlapping points contains more than '*min_samples*' points, it creates a cluster containing each of them. All the others are then considered outliers. Given our assumptions, it is one of the best ways of clustering our homology dataset. The other tries to fit N gaussians on the data point distribution. It is agnostic of the shape of the data, but won't create outliers. However it can create almost flat "gaussians" if needs be. Mixtures of gaussians are "Bayesians" and allow us to compute Bayesian criteria. It also enables us to sample new data points from the model given the infered prior. Lastly the homology plot function, whose architecture is kept the same for all the plot functions across PyCUB, will take the reduced vectors and plot them: Either from matplotlib by using colors to display different clusters, or by using bokeh and sending all interesting metadata, such as *doublons, clusters, similarity_scores, KaKs_Scores, nans, ecai, cai, length and GCcount* to the bokeh JavaScript (JS) server. We have then created a JS function to display each

[2] We note that given our current computation, $A_v alue$ should have some bias. With the number of possible values increasing, changes in the computation to random draw will lead to an increasing bias as random draw does not scale the length of its subset of partition values.

[3] Moreover, because it is a stochastic optimization method, it may not reproduce twice the same output.

of them as color gradients and some in boxes appearing when the user's mouse hovers over the data points. Each color gradient can be chosen by clicking on (radio)buttons. Additionally, matplotlib can display up to 4 dimensions, the 4^{th} being shown from dot sizes.

homoset clusterings *get_clusterstats, compare_clusters, find_clusters, findbest_eps, _barplot, plot_simiclust, plot_distcub*

The clustering function seen previously is always called from a homoset. It will decide a hyper-parameter for all homologies. They can either be given or computed from a grid search algorithm. The grid search will try to find the hyper-parameter which will create clusters with the smallest average phylogenetic distances. The phylogenetic distances of the data-points being those of their corresponding species. Once the clusters have been found, two functions try to extract some information out of them. *get_clusterstats* will, for each homology and each species, find what percentage of their data-points are in primary, secondary or outlier clusters. Where secondary clusters refer to any other clusters than the first ones: as most should only have one cluster. For each, *_barplot* will create stacked plots representing the percentages, allowing one to find specific homologies and species of interest.

Finally, *compare_clusters* will create a comparison matrix between each homology showing their similarities regarding different metrics: such as their mean cai; mean ecai; similarities; their cluster similarities that is, how much clustering of the species they have in common overlap each other; their CUB similarities that is, the average distance of the CUB of the species in common. Additionally it can plot a notion of difference in *mean nans, lenmat, GCcount, similarity_scores* and display all other information about each homology. Matrices are plotted with bokeh, using hover-able rectangles of different color gradients and (radio)buttons to select the similarity parameters.

clustering homoset *cluster_homologies, kmodes, loadhashomo, orderfromclust, plot_hashomo, plot_homoperspecies, plot_speciesperhomo*

Clustering homosets, is about finding homologies that share similar species, in order to do our computation over groups that can be compared. Moreover one would most often want homologies that contain many species, e.g. more than 200, to have relevant statistics. We have created a binary matrix representing each homology and whether or not it holds each possible species of the set. From this matrix, we can sort out rows and columns to be left with a matrix with clusters that are highly similar. We have found that the best method was to use k-means, even though we are talking here about binary 500 dimensional vectors. We knew we had to get a distance notion meant for binary values, as presented in (Choi, Cha, and Tappert, 2010). That is why we used a special version of k-means for binary data called **k-modes**. Finally, to visualize the clusters and similarities among species and homologies, or on the homologies and species they share, we plotted **cosine similarity** matrices -which is a measure of angular distances among vectors-.

homoset basic *get_working_homoset, get_subset, __getitem__, __setitem__, __delitem__, __iter__, iteritems, __len__, update, remove, clean_species, size, add_random_homology*

Here we find most utilities for sets of homologies. First of all python magic functions to set a new item, get an item, create an iterator over items, create

a special iterator over items and retrieve the number of items. *Update* allows you to add a new key-item pair. You can also remove a key-item and remove a specific species within every homology that has it. You can get the total number of CUB values across all the homologies, create a random homology which is only filled with random values, to see if any of our clustering and dimension reduction algorithm gets tricked by it. Finally, one can select sub-samples of the homosets. *get_working* does that by selecting a group of homologies from one of the different homoset clusters -see above-. Except if explicitly expressed, the homologies will only be references to *all_homoset* homologies and thus modifying them will also modify those of other homosets. *get_subset* does the same as a homoset but one needs to state exactly particular homologies and can also filter species.

homoset all plot *plot_all, tsne_batches, Multicoretsne, heatmap, loadfullhomo*

Here it is about having a deeper look into the genes, regardless of homologies. Genes are considered together and plotted. A computational decision is taken according to the size of the sample as there can be more than 2,000,000 data-points and it may create some difficulties. PCA works well in any case here. It is very efficient and scales linearly with the sample size. However it is not the case for t-SNE. We thus use a particular version that can scale better by parallelizing some of its workload (Van Der Maaten, 2014; Ulyanov, 2016). It uses stochasticity in Barns-Hut optimization method to scale and it achieves more than 8-fold speed improvement, while requiring less memory than scikit-learn's version -which overflew any memory size we threw at it-. This is thanks to batch processing, which is also explored in a third, non parallel version of t-SNE called t-SNE batches. All of the methods here are recurrent in Machine learning to speed up optimization processes. Finally we had to make the choice about plotting, as 2 million points won't be possible to display. We use datashaders and binning to display a sense of density instead. We use holoview's API 1.4.2 to plot them easily on both bokeh or matplotlib. **Datashaders** will display real point cloud whereas **heatmap** uses binning, where we put every point on a 2D histogram and where height is represented by gradients of color.

Final comparisons *compare_species, regress_on_species, compare_homologies, regress_on_genes*

The final comparison goal is to find correlations between the CUB values and the metadata. We have 3 complementary ways of achieving this goal and 2 different sets of functions to compare either species or homologies, with all their metadata. The first technique is a visual one, with interactive bokeh plotting using color gradients and dot sizes. It allows one to explore relationships visually and to get hints about some species or homologies and what can explain their values. The second uses regression (Zhao and Yu, 2006), either on the CUB values or the average CUB value and uses all given metadata as regressors. Two distinct regression algorithms can be used here: either a simple **Lasso**-powered linear regression, with a regressor selection by cross validation (**CV**) (Kohavi, 1995) -allowing one to see only the values that have true linear correlations with the CUB- or a **multilayer perceptron** (Van Der Malsburg, 1986): a simple type of regressive neural network. This method however is non linear and uses ReLU non-linearities (Glorot, Bordes, and Bengio, 2011). It will also use every supplied parameter and will give less as a comprehensible result. It might find however a relation that lasso can't. We have used

two layers of the size of the number of attributes to constrain this neural network. Each algorithm is agnostic on the given parameters, i.e. the user can add new parameters to the object and it will still be taken into account. Parameters are normalized. Lasso and Perceptron continue on the idea of using different methods. A simpler, more consistent and famous one, with a more complex one which may yield results the other can't. Finally, we have used the **spearman's rho**,a measure of correlation between each continuous variable, to see if it was directly correlated with the CUB. Together, these functions should be enough to investigate the predictive power of different variables to the CUB. An idea here of considering a predictive power for each homology and species is that it permits to average the values across the inner homologies and species. It may thus remove either evolutionary or functional influences.

Compute3DGD *Compute3DGD, endres_distances, kl, search*

In the context of computing the 3D genome distance, we are retrieving the genome of the reference species for a given kingdom. The function should be given files containing inter and intra-chromosomal distance information for the species.There should be tables, containing for each index, values in the form locus1, chr1 and locus2, chr2 information which state which chromosome+position has been found close to which chromosome+position. As explained earlier and further explained in the Science paper, we compute, for each gene, entropy values, codon frequency values, synonymous codon frequency values and chromosome + locus position of the gene center. We can then match-sort each of them with its closest chrom+locus position on the HiC tables. Alternatively, an approximate binary search can be used. We can then create a distance matrix containing the one-hop distance between each gene and compute a distance matrix from each codon usage value as well. The recommended metrics to compute distances of such data are either Euclidean distances:

$$d_{euclidean}(\mathbf{q}, \mathbf{p}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{A.1}$$

or **Endres-Schrindelin distances** (Endres and Schindelin, 2003):

$$d_{ES}(\mathbf{q}, \mathbf{p}) = \sqrt{d_{KL}(\mathbf{p}, \mathbf{m}) + d_{KL}(\mathbf{q}, \mathbf{m})} \tag{A.2}$$

where $M \equiv \frac{(\mathbf{p}+\mathbf{q})}{2}$ and d_{KL} is:

$$di_{KL}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \log \frac{p_i}{q_i} p_i \tag{A.3}$$

A Endres-Schrindelin distance also uses Kullback-Leibler divergences -see above-. The values should then be sorted by their codon usage distance values. Each value -both 3D distances and codon usage distances- should be placed and averaged into X bins: X varying according to the size of the genome. e.g. 2,000 for cerevisiae). Then a Spearman correlation is computed between the values.

A.1.2 More about eCAI

eCAI is an unofficial measure that we have decided to create and use, to get more meta information about the genes. It is based on the CAI measure. Here instead of measuring the deviation of synonymous codon usage distribution to synonymous codon usage distributions of highly expressed genes, we compare it to the reference gene distribution. It gives an idea of evolutionary drift. One can see that we often use the reference species as a surrogate for a common ancestor species.

A.1.3 More about the 3D genome correlation paper

Using **HiC** data from different species gives information about which part of the genome has been found close to which other part and how often (Belton et al., 2012). It gives rise to a binary network, linking genes that have often been found close together. With graph, we compute a one-hop distance matrix with **Dijkstra's algorithm** boosted with **Fibonacci's heap**. It computes the shortest path between any 2 points of a distance matrix and gives a method of computing the minimal distance between any two genes. It is boosted by using a hierarchical ordering of the memory and computations. One can then correlate this distance with the distance between the two vectors representing the codon usage of each of these two genes. To put it simply, the matrix of actual distances between the genes is related with the similarity distance of codon usage between them. -see B.5-.

Alon Diament, for Saccharomyces Cerevisiae (SC), found a correlation of 85%, when binning the distance matrices into a smaller set 2,000- values which represents the average of the values put in each bin. He then found a correlation of 50% when using the CUB frequency, i.e.the frequency over synonymous codons only-

$$c_i = \frac{n_i}{\sum_{j \in AA} n_j} \tag{A.4}$$

where:

$$\sum_{i=1}^{64} c_i = 21 \tag{A.5}$$

He posited that the organism activity is localized and that this is true in the nucleus as well. Transcription factors, tRNAs and other mechanisms related to the protein function are thus localized in the organism. The works inferred that it has an impact on the 3D conformation of the genome. Indeed, it tries to conform in such a way as to have its genes with related functions in similar places in the nuclei.

A.1.4 More about the Development pipeline

The development pipeline was the following:

Python2.7 with Conda+jupyter. Sublime as the IDE with Anaconda package and some other packages for better integration, linting and documentation access. Everything was done under UNIX systems and version with git+github. Moreover, to speed up computation, an EC2 machine from AWS was used. CD5 xLarge with 8GB of RAM anf 50Gb of slow swap memory, 4 cores and 3.5Gb of bandwidth and 100Gb SSD NVME -see the code for installation-. We used remote sublime, screen

command and a jupyter notebook server for the communication. This development pipeline is taken from iterative development techniques whose goal is to be able to pivot easily during the prototyping phase. It is inspired from a data science project which heavily relies on python for its ease of use, fast development and most of all numbers of data science related packages. AWS is used when comes the need to test the code on large batches fast and is also heavily used in data science. To produce the documentation of the code, We have also used doxygen -a tool to turn inlined documentation into a full software engineer documentation with diagrams, relationships, types etc. we have used an additional package called doxypypy to enable doxygen to read the pythonic inlined documentation we were writing.

A.1.5 Other Packages

H5py It is a python API for HDF5. It uses numpy arrays as a way of storing the data. HDF5 is a computational framework to do fast computation on multi terabyte objects directly from disk -and thus keep them stored there-. This toolkit would have been extremely interesting for some of the multi gigabyte matrices we produced, as well as to scale the computations to other, more furnished kingdoms.

gffutils It allows you to interact with gff file format used by ensembl and which contains genomic annotations. It could have been useful for other databases as well but we found that most of the genomic annotations were already available from the REST API of ensembl.

genetic-collections A library for connecting and comparing genetic records with specimen data using NCBI's database. However, it required Python3.

biothings It would have been a useful package to finish our project, as it allows you to create easily a web API with it.

elasticsearch It contains many more specifically designed tools for genomic data. It could have helped us to present the output of our work as a web service.

rnacounter It estimates the abundance of genes and their different transcripts from read alignments. Exons and introns can also be quantified when using it. It could also have been useful for some databases. However it was not needed for ensembl database, which already provided such pre-processing.

codontools A python package containing various tools for codon optimization. which could have been very interesting, but was not documented at all.

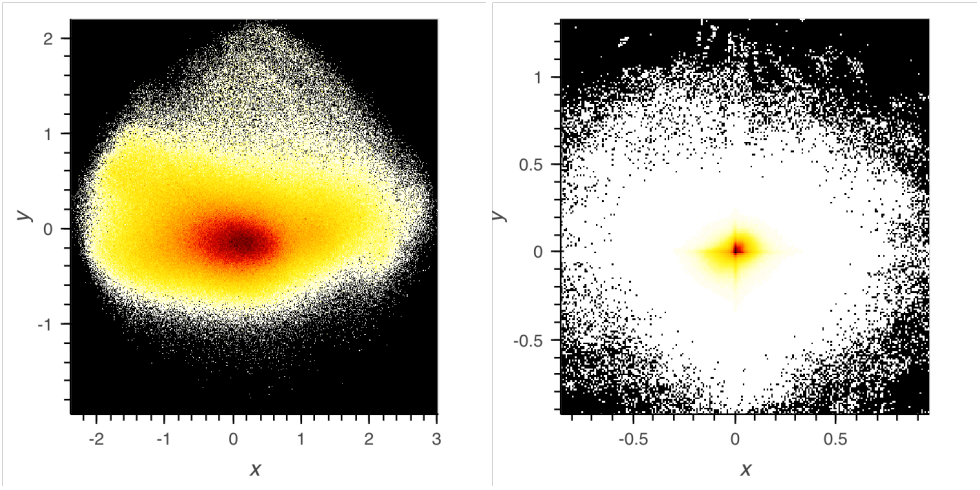
popsc It is a toolkit which, according to population genetic sequences, allows you to compute 45 different statistics about it. It could have been used to get more information about phylogeny, by considering all the species as parts of the same population.

phylogenetics A python package to manage phylogenetic projects, which could have helped me quite a lot, considering PyCUB is a phylogenetic project. However it seemed to be too highly levelled and too restrictive for the pipeline we wished to build.

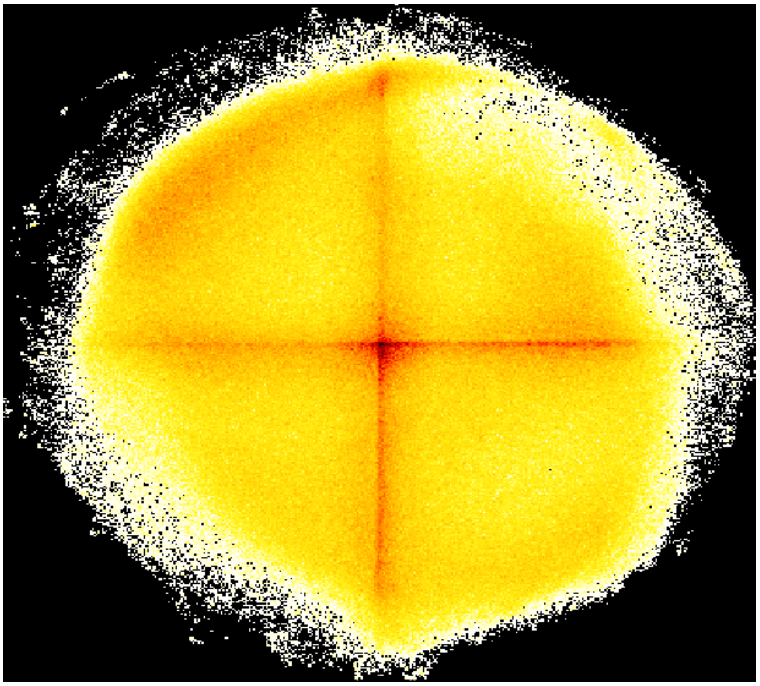
Appendix B

Supplementary Results

B.1 Supplementary plots



(A) PCA with frequency (B) the same but from t-SNE.



(C) from t-SNE on A_{value}

FIGURE B.1: t-SNE finds back the 4 clusters but with much more difficulty. we can see that PCA shows some blob from frequency measures.

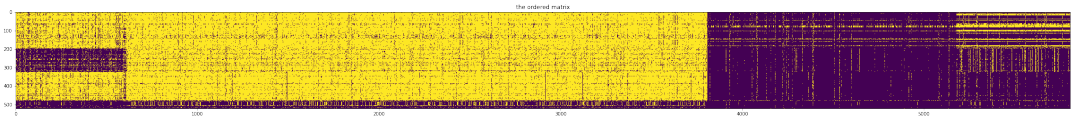


FIGURE B.2: The list of whether or not each species is homologue with SC on each of its gene.

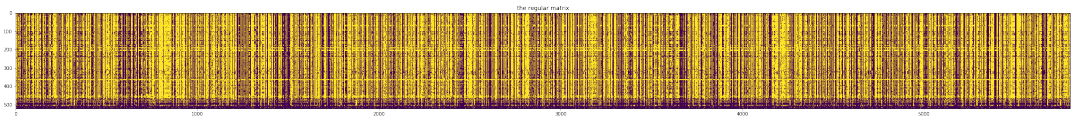


FIGURE B.3: same before clustering.

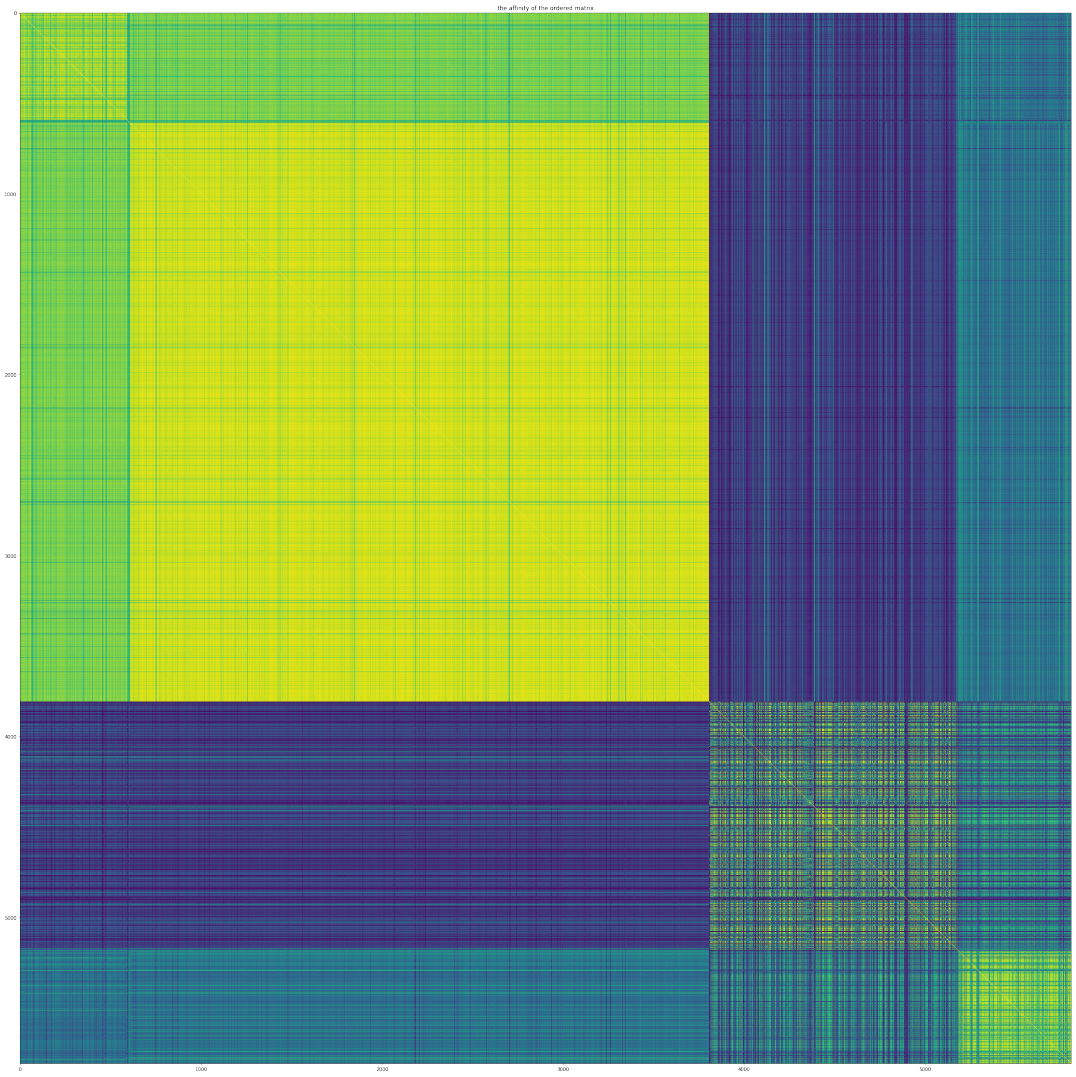


FIGURE B.4: now a similarity of each homology by its number of shared species.

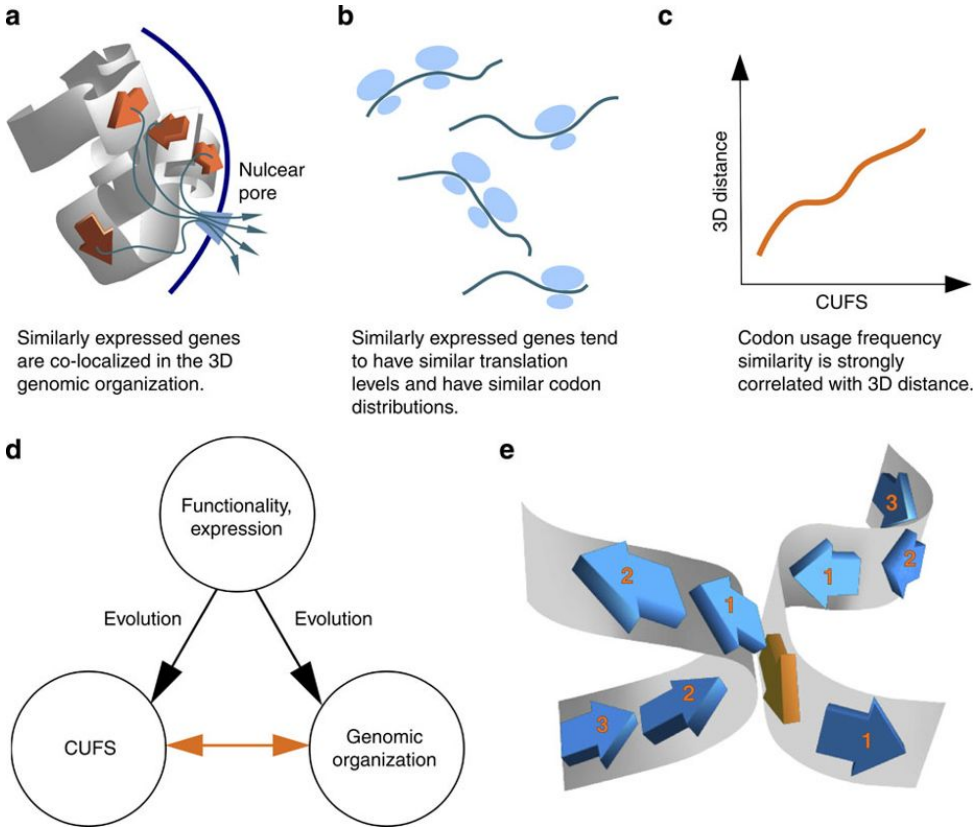


FIGURE B.5: (**a**) An illustration of examples of hypothetical evolutionary processes that contribute to the observed strong correlation of CUFS and genomic distance. Genes with similar transcription levels are expected to be clustered in the 3D genomic organization, and thus are inclined to be closer in the 3D genomic organization. (**b**) We expect that genes with similar translation levels (that tend to have similar transcription levels) will have similar CUB; for example, highly expressed genes usually undergo stronger selection for codons that are more adapted to the intracellular tRNA pool to improve translation efficiency27, and thus have more similar CUB. (**c**) Eventually, a correlation between CUFS and 3D genomic distance is observed, although CUFS is related to translation and not only to transcription. (**d**) Plan of study: CUFS is known to be related and thus to evolve with gene expression and functionality (left arrow); we want to show that genes functionality and expression are strongly related to their genomic organization (right arrow); thus, by showing that there is a strong relation and adaptation between CUFS and 3D genomic distance (orange arrow), we actually show that there is strong correlation between the functionality and expression of genes and their 3D genomic organization. (**e**) Representation of the data. The diagram displays a single measured interaction between two DNA fragments, based on Hi-C data. Each arrow is a protein-coding gene, as well as a node on the graph. The interaction was mapped to be between the two nodes (arrows) closest to the point of interaction. The orange arrow is a reference node for all distances in the diagram (denoted in orange numbers on each node); see further details in (Diament, Pinter, and Tuller, 2014) Methods section.

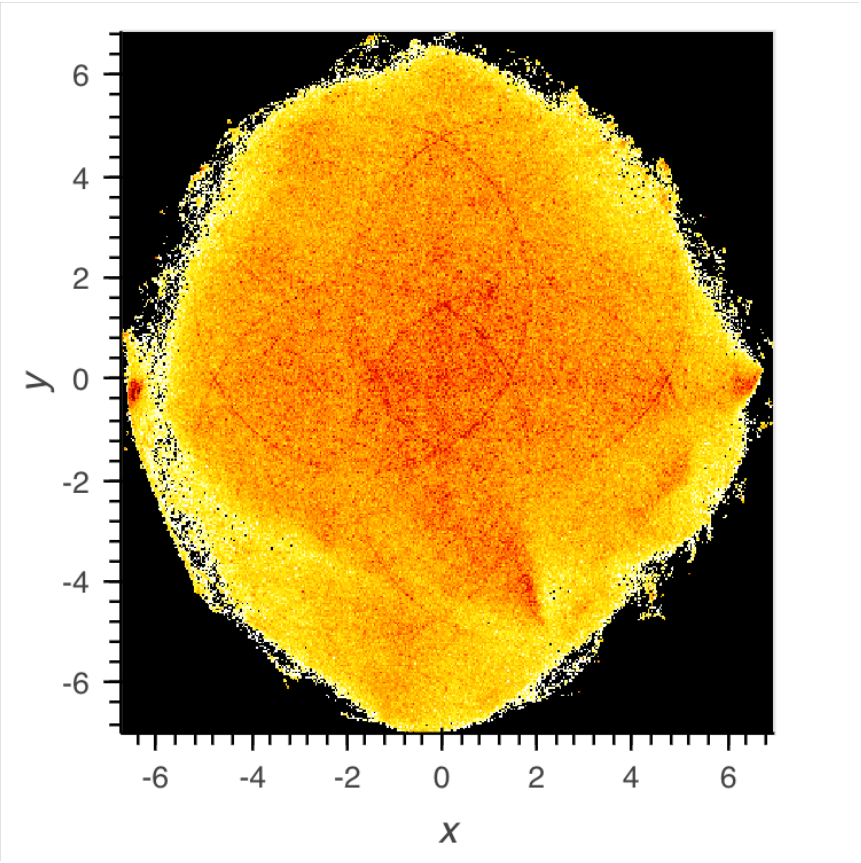
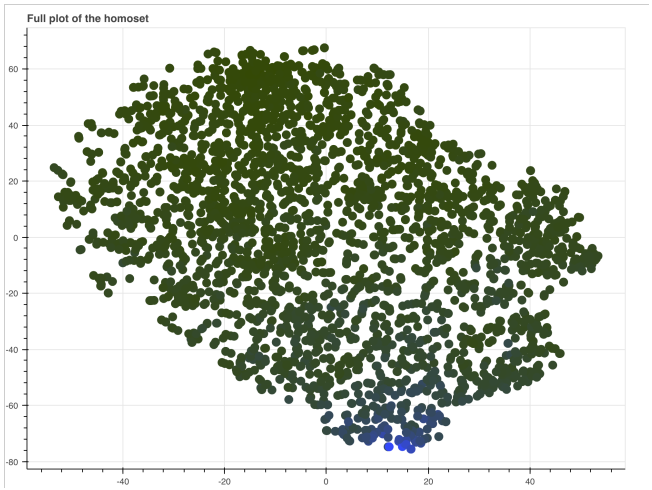
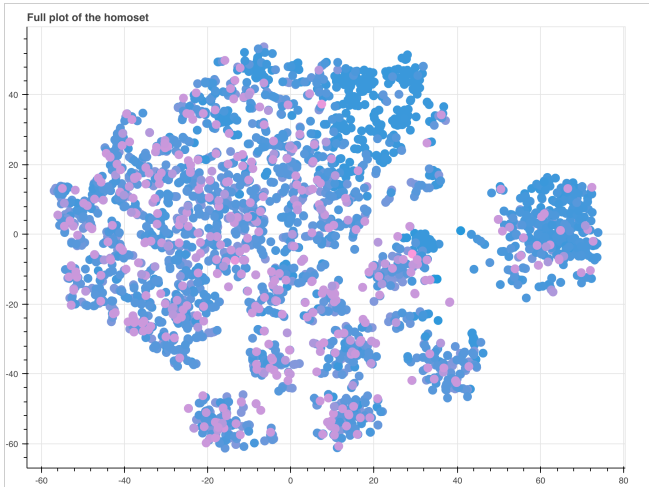


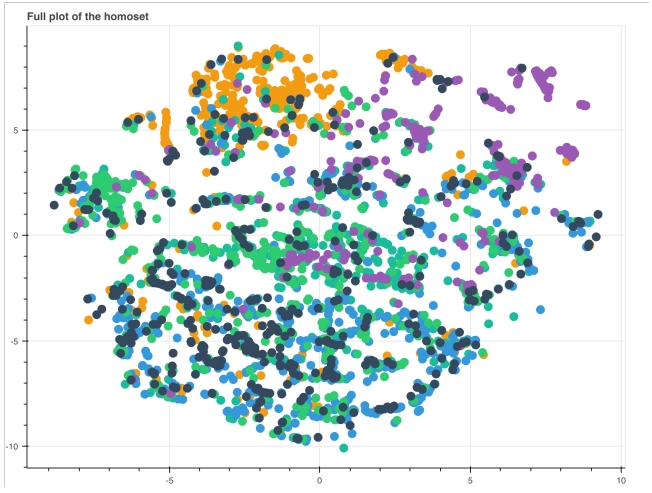
FIGURE B.6: the heatmap of all homologous genes from entropy, with different perplexity. Here we can see that the 4 clusters are represented as half circle instead of lines.



(A) comparison of entropy to mean.



(B) comparison of entropy cost to length.



(C) From high entropy homology.

FIGURE B.7: multiple homology plots, (A) correlated to CAI and (B) correlated to the GC content.

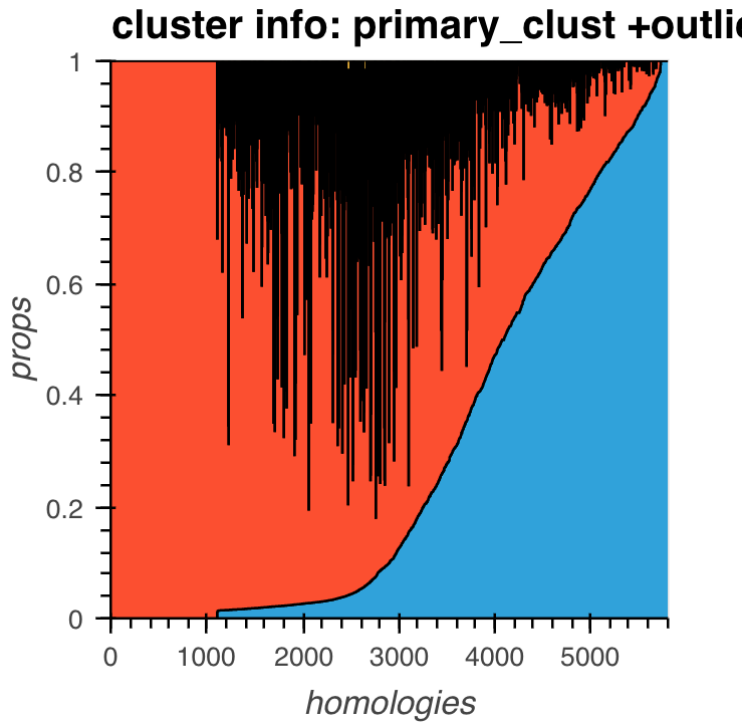


FIGURE B.8: some variation of cluster presence in homologies from frequency showing wrong hyperparameters on the extremes of the plot.

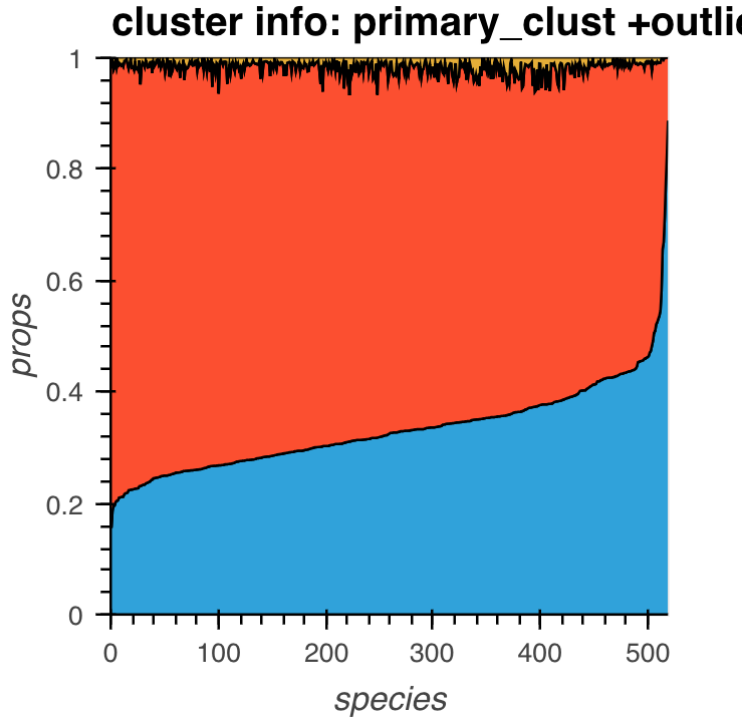
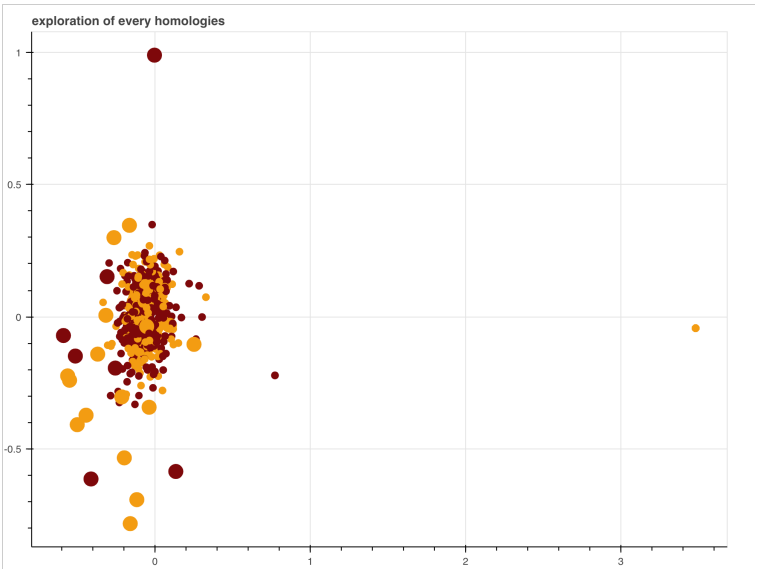
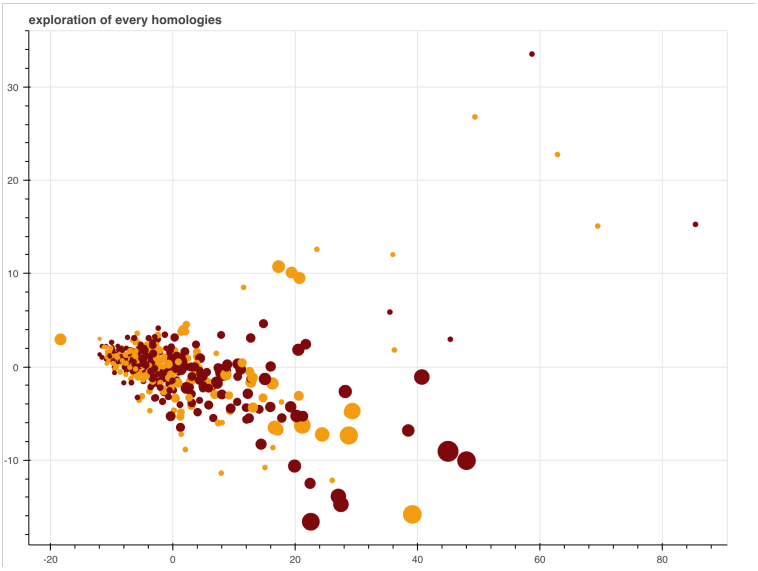


FIGURE B.9: A sort of the distribution of cluster per species. we can see some with very high number of outlier clusters on average.

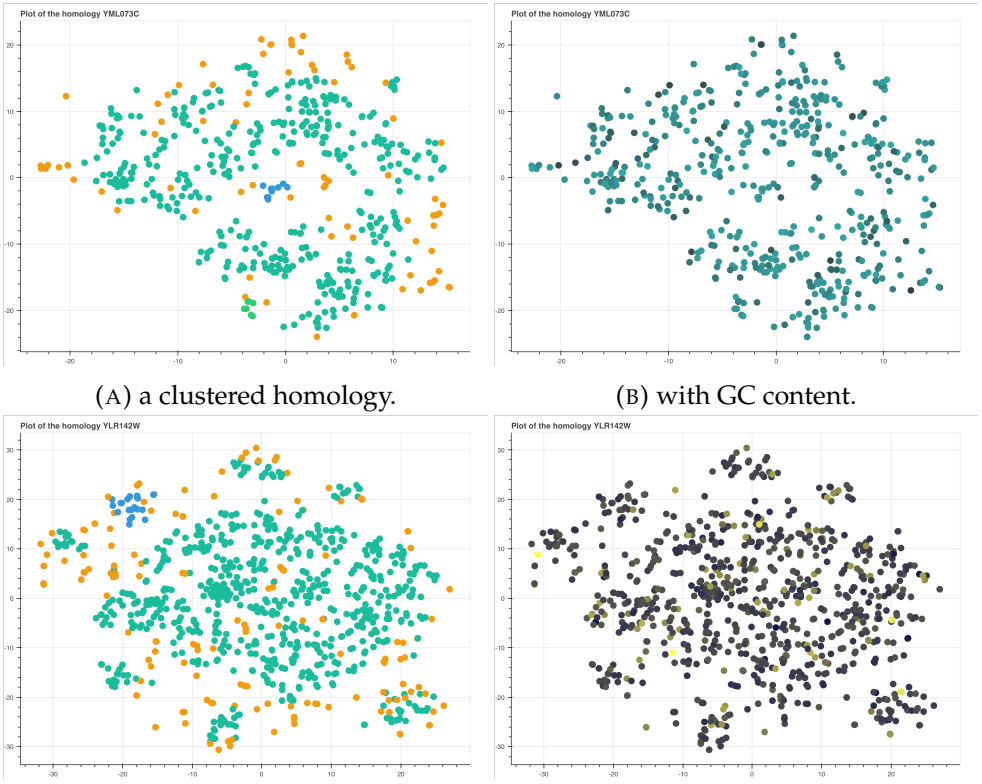


(A) From entropy cost.



(B) From frequency.

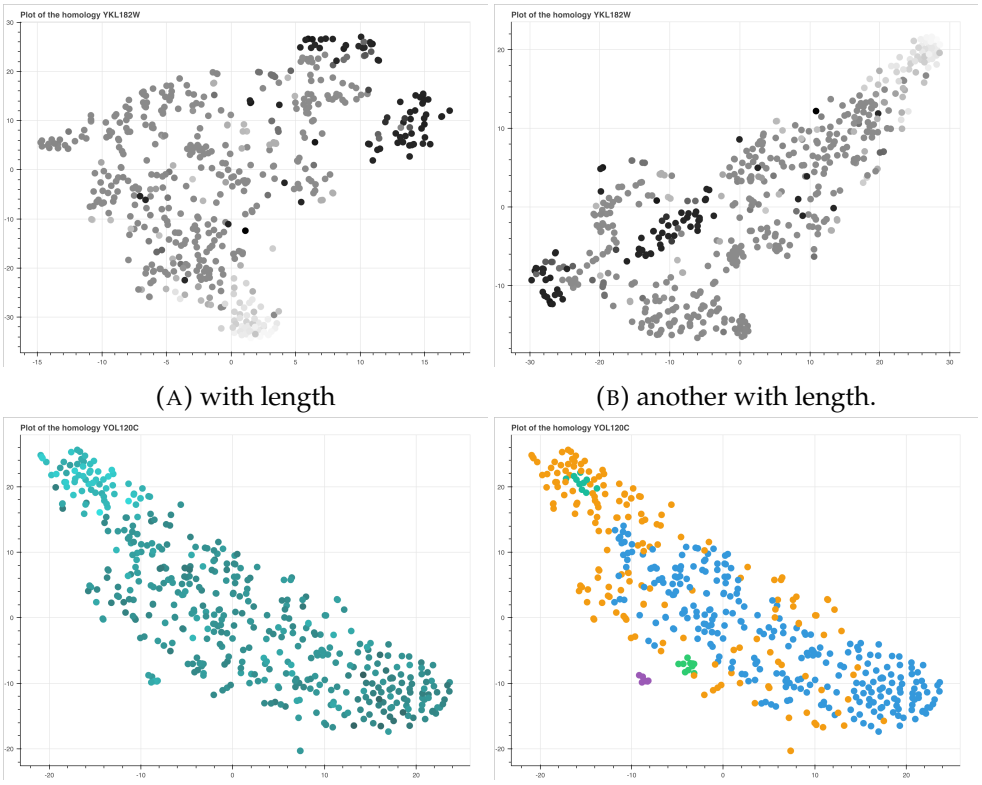
FIGURE B.10: Species CUB with PCA.



(A) a clustered homology. (B) with GC content.

(C) another clustered homology. (D) with sequence similarity.

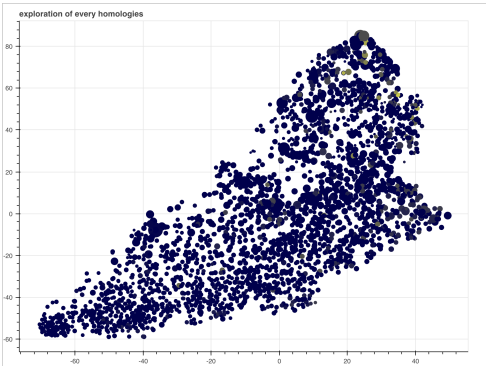
FIGURE B.11: Other homologies with entropy cost of their CUB compared with different statistics



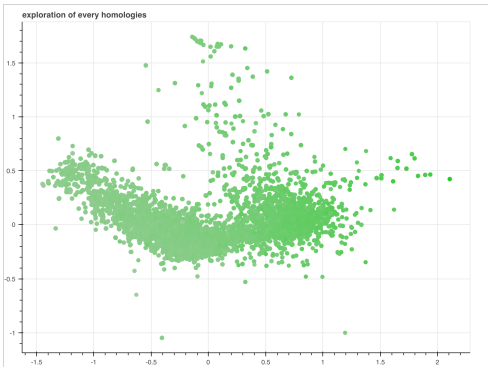
(A) with length (B) another with length.

(C) with CAI. (D) with clustering.

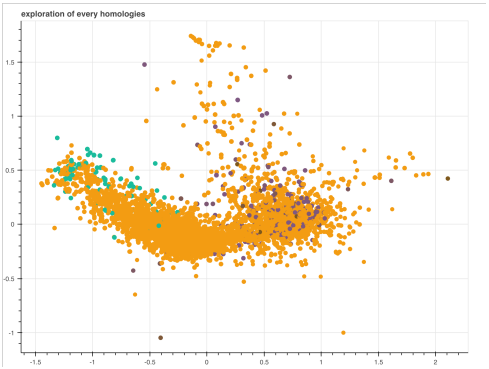
FIGURE B.12: Other homologies with different parameters overlaid.



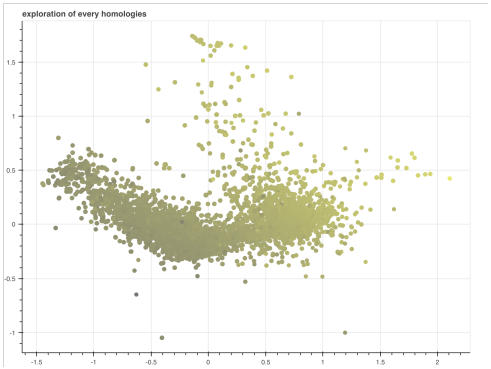
(A) with number of cys regulatory elements



(B) with the GC content



(C) With recent homologies



(D) with the CAI

FIGURE B.13: homology average via PCA on frequency measures.

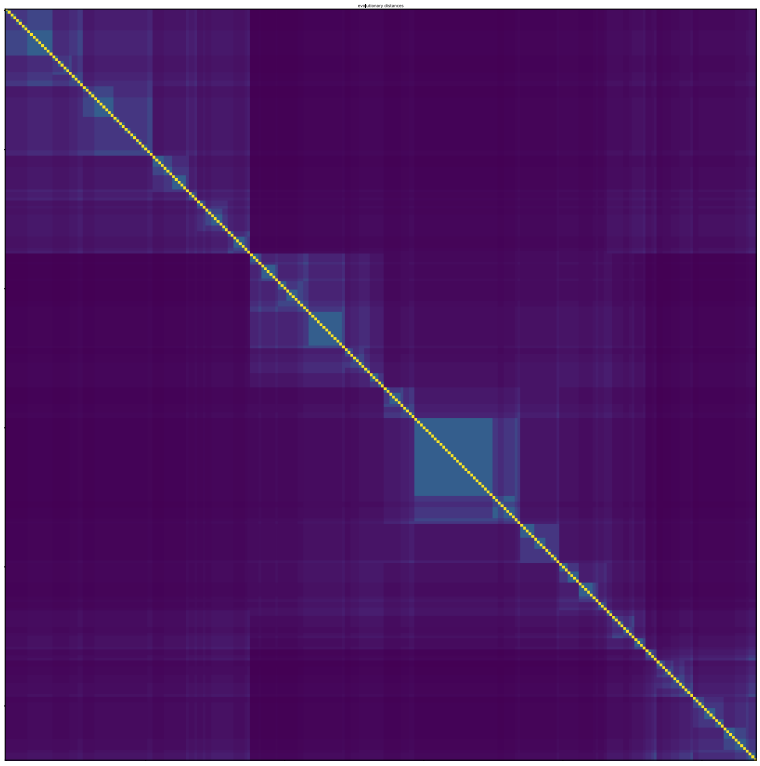


FIGURE B.14: phylogenetic distances amongst 400 of the species of fungi studied -some of them were missing from the taxonomic databases.

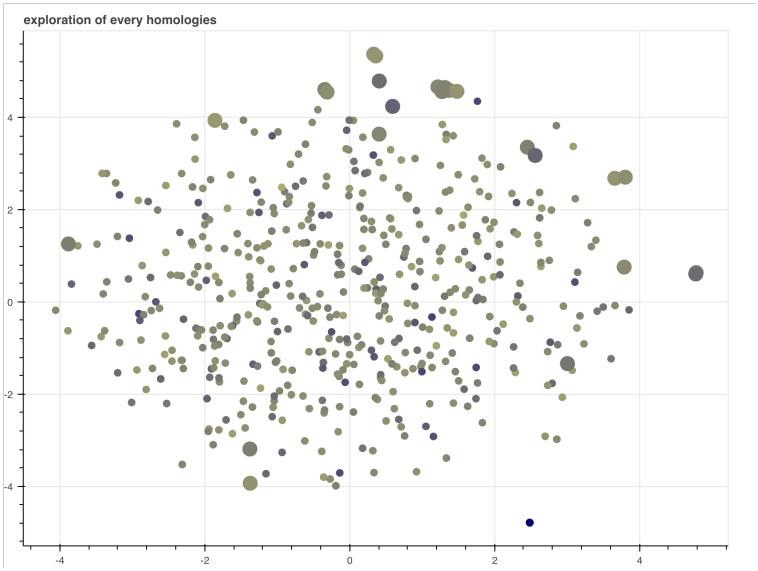


FIGURE B.15: Entropy location averaged over species: we can see that high variances span across all many different dimension -instead of just one for homologies-

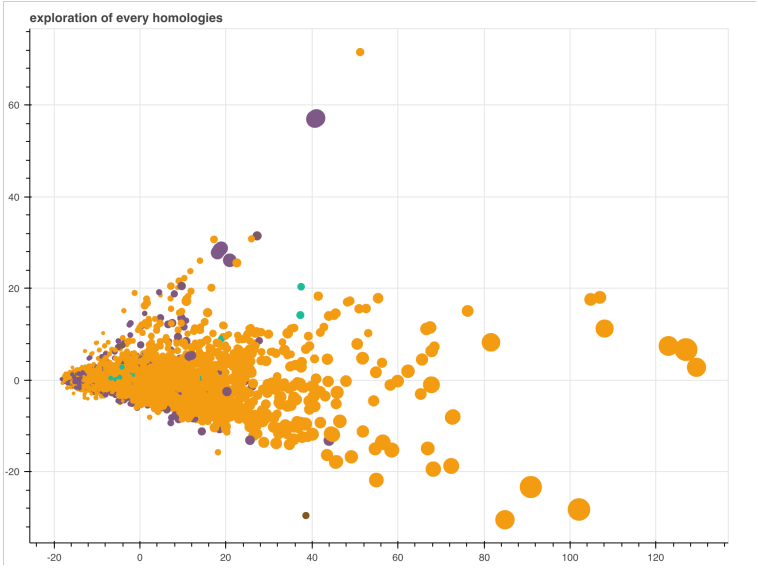


FIGURE B.16: averages over homologies for entropy cost. We can see that compared with what t-SNE outputs there is not much correlations we can find.

(A) from its clusters

(B) From the CUB distances

FIGURE B.17: Here is an example of how to compare homologies together. We compare only their common species. It is very helpful to spot outliers.

Bibliography

Abelson, Sagi et al. (2018). "Prediction of acute myeloid leukaemia risk in healthy individuals". In: *Nature* 559.7714, pp. 400–404. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0317-6. URL: https://doi.org/10.1038/s41586-018-0317-6.

Aguiar, Derek et al. (2018). "Bayesian nonparametric discovery of isoforms and individual specific quantification". In: *Nature Communications* 9.1, p. 1681. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03402-w. URL: https://doi.org/10.1038/s41467-018-03402-w.

Akaike, Hirotugu (2011). "Akaike's information criterion". In: *International encyclopedia of statistical science*. Springer, pp. 25–25. URL: http://www.sortie-nd.org/lme/Statistical\%20Papers/Akaike_1973\%20with\%20commentary.pdf.

Alipanahi, Babak et al. (2015). "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning". In: *Nature Biotechnology* 33, 831 EP –. URL: http://dx.doi.org/10.1038/nbt.3300.

Alpaydin, Ethem (2010). *Introduction to Machine Learning*. 2nd. The MIT Press. ISBN: 026201243X, 9780262012430.

Anfinsen, Christian B. (1973). "Principles that Govern the Folding of Protein Chains". In: *Science* 181.4096, pp. 223–230. ISSN: 0036-8075. DOI: 10.1126/science.181.4096.223. eprint: http://science.sciencemag.org/content/181/4096/223.full.pdf. URL: http://science.sciencemag.org/content/181/4096/223.

Arriel Benis Nissim Harel, Refael Barkan Tomer Sela Becca Feldman (2017). "Identification and Description of Healthcare Customer Communication Patterns Among Individuals with Diabetes in Clalit Health Services: A Retrospective Database Study". In: *Studies in Health Technology and Informatics* 244. Article, pp. 18–22. DOI: 10.3233/978-1-61499-824-2-18. URL: http://ebooks.iospress.nl/volumearticle/47785.

Athey, John et al. (2017). "A new and updated resource for codon usage tables". In: *BMC Bioinformatics* 18.1, p. 391. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1793-7. URL: https://doi.org/10.1186/s12859-017-1793-7.

Baker, Monya (2014). "1,500 scientists lift the lid on reproducibility". In: *Nature* 533.6168, pp. 452–454. DOI: 10.1038/533452a. URL: https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970.

Bali, Vedrana and Zsuzsanna Bebok (2015). "Decoding mechanisms by which silent codon changes influence protein biogenesis and function". In: *The International Journal of Biochemistry & Cell Biology* 64, pp. 58 –74. ISSN: 1357-2725. DOI: https://doi.org/10.1016/j.biocel.2015.03.011. URL: http://www.sciencedirect.com/science/article/pii/S1357272515000801.

Barnes, Josh and Piet Hut (1986). "A hierarchical O(N log N) force-calculation algorithm". In: *Nature* 324, 446 EP –. URL: http://dx.doi.org/10.1038/324446a0.

"Bayesian Information Criteria" (2008). In: *Information Criteria and Statistical Modeling*. New York, NY: Springer New York, pp. 211–237. ISBN: 978-0-387-71887-3. DOI: 10.1007/978-0-387-71887-3_9. URL: https://doi.org/10.1007/978-0-387-71887-3_9.

Beam, A. L. et al. (2018). "Clinical Concept Embeddings Learned from Massive Sources of Multimodal Medical Data". In: *ArXiv e-prints*. arXiv: 1804.01486 [cs.CL].

Belton, Jon-Matthew et al. (2012). "Hi-C: A comprehensive technique to capture the conformation of genomes". In: 58.

Benhenda, M. (2017). "ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity?" In: *ArXiv e-prints*. arXiv: 1708.08227 [stat.ML].

Bertsekas, Dimitri P. and John N. Tsitsiklis (1996). *Neuro-Dynamic Programming*. 1st. Athena Scientific. ISBN: 1886529108.

Buhr, Florian et al. (2016). "Synonymous Codons Direct Cotranslational Folding toward Different Protein Conformations". In: 61, pp. 341–351.

Carter, Andrew V et al. (2002). "Deficiency distance between multinomial and multivariate normal experiments". In: *The Annals of Statistics* 30.3, pp. 708–730. URL: http://www.stat.yale.edu/~pollard/Manuscripts+Notes/Paris2001/Lectures/Multinomial.pdf.

Cengizler, Caglar and M Kerem (2017). "Evaluation of Calinski-Harabasz Criterion as Fitness Measure for Genetic Algorithm Based Segmentation of Cervical Cell Nuclei". In: 22, pp. 1–13. URL: http://www.journalrepository.org/media/journals/BJMCS_6/2017/Jun/Cengizler2262017BJMCS33729.pdf.

Chamary, J. V., Joanna L. Parmley, and Laurence D. Hurst (2006). "Hearing silence: non-neutral evolution at synonymous sites in mammals". In: *Nature Reviews Genetics* 7. Review Article, 98 EP –. URL: http://dx.doi.org/10.1038/nrg1770.

Ching, Travers et al. (2018). "Opportunities And Obstacles For Deep Learning In Biology And Medicine". In: *bioRxiv*. DOI: 10.1101/142760. eprint: https://www.biorxiv.org/content/early/2018/01/19/142760.full.pdf. URL: https://www.biorxiv.org/content/early/2018/01/19/142760.

Choi, Seung-Seok, Sung-Hyuk Cha, and Charles C Tappert (2010). "A survey of binary similarity and distance measures". In: *Journal of Systemics, Cybernetics and Informatics* 8.1, pp. 43–48. URL: http://www.iiisci.org/journal/CV\$/sci/pdfs/GS315JG.pdf.

Chris Olah Arvind Satyanarayan, Ian Johnson Shan Carter Ludwig Schubert Katherine Ye Alexander Mordvintsev (2018). "The Building Blocks of Interpretability". In: *Distill*. DOI: 10.23915/distill.00010.

Church, George M., Yuan Gao, and Sriram Kosuri (2012). "Next-Generation Digital Information Storage in DNA". In: *Science*. ISSN: 0036-8075. DOI: 10.1126/science.1226355. eprint: http://science.sciencemag.org/content/early/2012/08/15/science.1226355.full.pdf. URL: http://science.sciencemag.org/content/early/2012/08/15/science.1226355.

Consortium, International Human Genome Sequencing (2004). "Finishing the euchromatic sequence of the human genome". In: *Nature* 431. Article, 931 EP –. URL: http://dx.doi.org/10.1038/nature03001.

Cun, Yann Le (1988). *A Theoretical Framework for Back-Propagation*.

De Cao, N. and T. Kipf (2018). "MolGAN: An implicit generative model for small molecular graphs". In: *ArXiv e-prints*. arXiv: 1805.11973 [stat.ML].

De Smet, Riet and Kathleen Marchal (2010). "Advantages and limitations of current network inference methods". In: *Nature Reviews Microbiology* 8. Review Article, 717 EP –. URL: http://dx.doi.org/10.1038/nrmicro2419.

Diament, Alon, Ron Y. Pinter, and Tamir Tuller (2014). "Three-dimensional eukaryotic genomic organization is strongly correlated with codon usage expression and function". In: *Nature Communications* 5. Article, 5876 EP –. URL: http://dx.doi.org/10.1038/ncomms6876.

Ding, Z F et al. (2018). "An integrated metabolic consequence of Hepatospora eriocheir infection in the Chinese mitten crab Eriocheir sinensis." In: *Fish & shellfish immunology* 72, pp. 443–451. URL: https://www.ncbi.nlm.nih.gov/pubmed/29146449.

Do, Chuong B (2008). "The multivariate gaussian distribution". In: *Section Notes, Lecture on Machine Learning, CS* 229. URL: http://cs229.stanford.edu/section/gaussians.pdf.

Duan, Zhijun et al. (2010). "A three-dimensional model of the yeast genome". In: *Nature* 465, 363 EP –. URL: http://dx.doi.org/10.1038/nature08973.

Endres, Dominik and Johannes Schindelin (2003). "A new metric for probability distributions". In: 49, pp. 1858 –1860.

Ester, Martin et al. (1996). "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

Filion, Guillaume (2018). *A Tutorial on t-SNE*. eprint: 1404.1100. URL: http://blog.thegrandlocus.com/2018/08/a-tutorial-on-t-sne-1.

Fox, Jesse M. and Ivan Erill (2010). "Relative Codon Adaptation: A Generic Codon Bias Index for Prediction of Gene Expression". In: *DNA Research* 17.3, pp. 185–196. DOI: 10.1093/dnares/dsq012. eprint: /oup/backfile/content_public/journal/dnaresearch/17/3/10.1093/dnares/dsq012/2/dsq012.pdf. URL: http://dx.doi.org/10.1093/dnares/dsq012.

Frey, Brendan (2017). "Why Medicine Needs Deep Learning". In: 14. URL: https://www.youtube.com/watch?v=bYTq4QEDA78.

Giovannucci, Andrea et al. (2018). "CaImAn: An open source tool for scalable Calcium Imaging data Analysis". In: *bioRxiv*. DOI: 10.1101/339564. eprint: https://www.biorxiv.org/content/early/2018/06/05/339564.full.pdf. URL: https://www.biorxiv.org/content/early/2018/06/05/339564.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Haar, Tobias von der (2008). "A quantitative estimation of the global translational activity in logarithmically growing yeast cells". In: *BMC Syst Biol* 2. 1752-0509-2-87[PII], pp. 87–87. ISSN: 1752-0509. DOI: 10.1186/1752-0509-2-87. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2590609/.

Hershberg, Ruth and Dmitri A. Petrov (2008). "Selection on Codon Bias". In: *Annual Review of Genetics* 42.1. PMID: 18983258, pp. 287–299. DOI: 10.1146/annurev.genet.42.110807.091442. eprint: https://doi.org/10.1146/annurev.genet.42.110807.091442. URL: https://doi.org/10.1146/annurev.genet.42.110807.091442.

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: http://science.sciencemag.org/content/313/5786/504.full.pdf. URL: http://science.sciencemag.org/content/313/5786/504.

"IEEE Guide for Developing System Requirements Specifications" (1998). In: *IEEE Std 1233, 1998 Edition*, pp. 1–36. DOI: 10.1109/IEEESTD.1998.88826.

Kalfon, Jérémie (2018). "Data Science and Deep Learning: Overview from the standpoint of Bioinformatics".

Kelley, David R., Jasper Snoek, and John L. Rinn (2016). "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks". In: *Genome Research* 26.7, pp. 990–999. DOI: 10.1101/gr.200535.115. eprint: http://genome.cshlp.org/content/26/7/990.full.pdf+html. URL: http://genome.cshlp.org/content/26/7/990.abstract.

Khan, Javed et al. (2001). "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks". In: *Nat Med* 7.6. 11385503[pmid], pp. 673–679. ISSN: 1078-8956. DOI: 10.1038/89044. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1282521/.

Kohavi, Ron (1995). "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". In: Morgan Kaufmann, pp. 1137–1143.

Kozlowski, Lukasz P. (2016). "IPC - Isoelectric Point Calculator". In: *Biol Direct* 11. 159[PII], p. 55. ISSN: 1745-6150. DOI: 10.1186/s13062-016-0159-9. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5075173/.

Li, Heng et al. (2018). "A synthetic-diploid benchmark for accurate variant-calling evaluation". In: *Nature Methods* 15.8, pp. 595–597. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0054-7. URL: https://doi.org/10.1038/s41592-018-0054-7.

Li, Y. and W. Ping (2018). "Cancer Metastasis Detection With Neural Conditional Random Field". In: *ArXiv e-prints*. arXiv: 1806.07064 [cs.CV].

Libbrecht, Maxwell W. and William Stafford Noble (2015). "Machine learning applications in genetics and genomics". In: *Nature Reviews Genetics* 16. Review Article, 321 EP –. URL: http://dx.doi.org/10.1038/nrg3920.

Lin Liu Yinhu Li, Siliang Li Ni Hu Yimin He Ray Pong Danni Lin Lihua Lu Maggie Law (2012). "Comparison of Next-Generation Sequencing Systems". In: *Journal of Biomedicine and Biotechnology*. DOI: 10.1155/2012/251364. URL: http://dx.doi.org/10.1155/2012/251364.

Liu, Jie et al. (2018). "Unsupervised embedding of single-cell Hi-C data". In: *bioRxiv*. DOI: 10.1101/257048. eprint: https://www.biorxiv.org/content/early/2018/01/30/257048.full.pdf. URL: https://www.biorxiv.org/content/early/2018/01/30/257048.

Luo, Ruibang et al. (2018). "Clairvoyante: a multi-task convolutional deep neural network for variant calling in Single Molecule Sequencing". In: *bioRxiv*. DOI: 10.1101/310458. eprint: https://www.biorxiv.org/content/early/2018/04/28/310458.full.pdf. URL: https://www.biorxiv.org/content/early/2018/04/28/310458.

M. Comeron, Josep and Montserrat Aguadé (1998). "An Evaluation of Measures of Synonymous Codon Usage Bias". In: 47, pp. 268–74.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9, pp. 2579–2605. URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Maheshri, Narendra and Erin K. O'Shea (2007). "Living with Noisy Genes: How Cells Function Reliably with Inherent Variability in Gene Expression". In: *Annual Review of Biophysics and Biomolecular Structure* 36.1. PMID: 17477840, pp. 413–434. DOI: 10.1146/annurev.biophys.36.040306.132705. eprint: https://doi.org/10.1146/annurev.biophys.36.040306.132705. URL: https://doi.org/10.1146/annurev.biophys.36.040306.132705.

McNutt, Marcia (2014). "Reproducibility". In: *Science* 343.6168, pp. 229–229. ISSN: 0036-8075. DOI: 10.1126/science.1250475. eprint: http://science.sciencemag.

org/content/343/6168/229.full.pdf. URL: http://science.sciencemag.org/content/343/6168/229.

Mnih, Volodymyr et al. (2014). "Recurrent Models of Visual Attention". In: *CoRR* abs/1406.6247. arXiv: 1406.6247. URL: http://arxiv.org/abs/1406.6247.

Moore, G. E. (2006). "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff." In: *IEEE Solid-State Circuits Society Newsletter* 11.3, pp. 33–35. ISSN: 1098-4232. DOI: 10.1109/N-SSC.2006.4785860.

Movva, Rajiv et al. (2018). "Deciphering regulatory DNA sequences and noncoding genetic variants using neural network models of massively parallel reporter assays". In: *bioRxiv*. DOI: 10.1101/393926. eprint: https://www.biorxiv.org/content/early/2018/08/17/393926.full.pdf. URL: https://www.biorxiv.org/content/early/2018/08/17/393926.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 2014–2023. URL: http://proceedings.mlr.press/v48/niepert16.html.

O'Brien, Edward P. et al. (2014). "Understanding the Influence of Codon Translation Rates on Cotranslational Protein Folding". In: *Accounts of Chemical Research* 47.5, pp. 1536–1544. ISSN: 0001-4842. DOI: 10.1021/ar5000117. URL: https://doi.org/10.1021/ar5000117.

Osborne, Cameron S. et al. (2004). "Active genes dynamically colocalize to shared sites of ongoing transcription". In: *Nature Genetics* 36. Article, 1065 EP –. URL: http://dx.doi.org/10.1038/ng1423.

Paggi, Joe et al. (2017). "Predicting Transcriptional Regulatory Activities with Deep Convolutional Networks". In: *bioRxiv*. DOI: 10.1101/099879. eprint: https://www.biorxiv.org/content/early/2017/01/12/099879.full.pdf. URL: https://www.biorxiv.org/content/early/2017/01/12/099879.

Parrella, Paola et al. (2014). "Evaluation of microRNA-10b prognostic significance in a prospective cohort of breast cancer patients". In: *Mol Cancer* 13. 1476-4598-13-142[PII], pp. 142–142. ISSN: 1476-4598. DOI: 10.1186/1476-4598-13-142. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4055397/.

Pearson, W R and D J Lipman (1988). "Improved tools for biological sequence comparison". In: *Proceedings of the National Academy of Sciences* 85.8, pp. 2444–2448. ISSN: 0027-8424. DOI: 10.1073/pnas.85.8.2444. eprint: http://www.pnas.org/content/85/8/2444.full.pdf. URL: http://www.pnas.org/content/85/8/2444.

Pervez, Muhammad Tariq et al. (2014). "Evaluating the Accuracy and Efficiency of Multiple Sequence Alignment Methods". In: *Evol Bioinform Online* 10. ebo-10-2014-205[PII], pp. 205–217. ISSN: 1176-9343. DOI: 10.4137/EBO.S19199. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4267518/.

Poyatos, Juan F. and Laurence D. Hurst (2007). "The determinants of gene order conservation in yeasts". In: *Genome Biology* 8.11, R233. ISSN: 1474-760X. DOI: 10.1186/gb-2007-8-11-r233. URL: https://doi.org/10.1186/gb-2007-8-11-r233.

Rafalski, Antoni (2002). "Applications of single nucleotide polymorphisms in crop genetics". In: *Current Opinion in Plant Biology* 5.2, pp. 94 –100. ISSN: 1369-5266. DOI: https://doi.org/10.1016/S1369-5266(02)00240-6. URL: http://www.sciencedirect.com/science/article/pii/S1369526602002406.

Reis, Mario dos, Renos Savva, and Lorenz Wernisch (2004). "Solving the riddle of codon usage preferences: a test for translational selection". In: *Nucleic Acids Research* 32.17, pp. 5036–5044. DOI: 10.1093/nar/gkh834. eprint: /oup/backfile/content_public/journal/nar/32/17/10.1093/nar/gkh834/2/gkh834.pdf. URL: http://dx.doi.org/10.1093/nar/gkh834.

Reynolds, Douglas (2015). "Gaussian mixture models". In: *Encyclopedia of biometrics*, pp. 827–832. URL: http://statweb.stanford.edu/~tibs/stat315a/LECTURES/em.pdf.

Rohlf, F. James and David R. Fisher (1968). "Tests for Hierarchical Structure in Random Data Sets". In: *Systematic Biology* 17.4, pp. 407–412. DOI: 10.1093/sysbio/17.4.407. eprint: /oup/backfile/content_public/journal/sysbio/17/4/10.1093/sysbio/17.4.407/2/17-4-407.pdf. URL: http://dx.doi.org/10.1093/sysbio/17.4.407.

Rousseeuw, Peter J. (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53 –65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

Sander, Ian M., Julie L. Chaney, and Patricia L. Clark (2014). "Expanding Anfinsen's Principle: Contributions of Synonymous Codon Selection to Rational Protein Design". In: *Journal of the American Chemical Society* 136.3, pp. 858–861. ISSN: 0002-7863. DOI: 10.1021/ja411302m. URL: https://doi.org/10.1021/ja411302m.

Schmidt, M. et al. (2015). "The Danish National Patient Registry: a review of content, data quality, and research potential". In: *Clin Epidemiol* 7, pp. 449–490.

Schreiber, Jacob et al. (2018). "Multi-scale deep tensor factorization learns a latent representation of the human epigenome". In: *bioRxiv*. DOI: 10.1101/364976. eprint: https://www.biorxiv.org/content/early/2018/07/08/364976.full.pdf. URL: https://www.biorxiv.org/content/early/2018/07/08/364976.

Sharp, P. M. and W. H. Li (1987). "The codon Adaptation Index–a measure of directional synonymous codon usage bias, and its potential applications." In: *Nucleic Acids Res* 15.3. 3547335[pmid], pp. 1281–1295. ISSN: 0305-1048. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC340524/.

Shin, Young C. et al. (2015). "Importance of codon usage for the temporal regulation of viral gene expression". In: *Proceedings of the National Academy of Sciences* 112.45, pp. 14030–14035. ISSN: 0027-8424. DOI: 10.1073/pnas.1515387112. eprint: http://www.pnas.org/content/112/45/14030.full.pdf. URL: http://www.pnas.org/content/112/45/14030.

Shlens, J. (2014). "A Tutorial on Principal Component Analysis". In: *ArXiv e-prints*. arXiv: 1404.1100.

Singer, G. A. and D. A. Hickey (2003). "Thermophilic prokaryotes have characteristic patterns of codon usage, amino acid composition and nucleotide content". In: *Gene* 317.1-2, pp. 39–47.

Stergachis, Andrew B. et al. (2013). "Exonic Transcription Factor Binding Directs Codon Choice and Affects Protein Evolution". In: *Science* 342.6164, pp. 1367–1372. ISSN: 0036-8075. DOI: 10.1126/science.1243490. eprint: http://science.sciencemag.org/content/342/6164/1367.full.pdf. URL: http://science.sciencemag.org/content/342/6164/1367.

Sudmant, Peter H. et al. (2015). "An integrated map of structural variation in 2,504 human genomes". In: *Nature* 526. Article, 75 EP –. URL: http://dx.doi.org/10.1038/nature15394.

Trotta, E. (2013a). "Selection on codon bias in yeast: a transcriptional hypothesis". In: *Nucleic Acids Res.* 41.20, pp. 9382–9395.

– (2013b). "Selection on codon bias in yeast: a transcriptional hypothesis". In: *Nucleic Acids Res.* 41.20, pp. 9382–9395.

Tuller, Tamir and Hadas Zur (2015). "Multiple roles of the coding sequence 5' end in gene expression regulation". In: *Nucleic Acids Research* 43.1, pp. 13–28. DOI: 10.1093/nar/gku1313. eprint: /oup/backfile/content_public/journal/nar/43/1/10.1093_nar_gku1313/2/gku1313.pdf. URL: http://dx.doi.org/10.1093/nar/gku1313.

Ulyanov, Dmitry (2016). *Multicore-TSNE*. https://github.com/DmitryUlyanov/Multicore-TSNE.

Unknown. *Analysis of discrete data: The Multinomial Distribution*.

Van Der Maaten, Laurens (2014). "Accelerating t-SNE using tree-based algorithms". In: *The Journal of Machine Learning Research* 15.1, pp. 3221–3245. URL: http://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf.

Van Der Malsburg, C. (1986). "Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms". In: *Brain Theory*. Ed. by Günther Palm and Ad Aertsen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 245–248. ISBN: 978-3-642-70911-1.

Wang, Bo et al. (2017). "Vicus: Exploiting local structures to improve network-based analysis of biological data". In: *PLOS Computational Biology* 13.10, pp. 1–18. DOI: 10.1371/journal.pcbi.1005621. URL: https://doi.org/10.1371/journal.pcbi.1005621.

Weatheritt, Robert J. and M. Madan Babu (2013). "The Hidden Codes That Shape Protein Evolution". In: *Science* 342.6164, pp. 1325–1326. ISSN: 0036-8075. DOI: 10.1126/science.1248425. eprint: http://science.sciencemag.org/content/342/6164/1325.full.pdf. URL: http://science.sciencemag.org/content/342/6164/1325.

Wright, Frank (1990). "The 'effective number of codons' used in a gene". In: *Gene* 87.1, pp. 23 –29. ISSN: 0378-1119. DOI: https://doi.org/10.1016/0378-1119(90)90491-9. URL: http://www.sciencedirect.com/science/article/pii/0378111990904919.

Xu, Yao et al. (2013). "Non-optimal codon usage is a mechanism to achieve circadian clock conditionality". In: *Nature* 495, 116 EP –. URL: http://dx.doi.org/10.1038/nature11942.

Yang, Ziheng and Joseph Bielawski (2009). "Statistical Methods for Detecting Molecular Adaptation". In: 15.

Zerbino, Daniel R et al. (2018). "Ensembl 2018". In: *Nucleic Acids Research* 46.D1, pp. D754–D761. DOI: 10.1093/nar/gkx1098. eprint: /oup/backfile/content_public/journal/nar/46/d1/10.1093_nar_gkx1098/2/gkx1098.pdf. URL: http://dx.doi.org/10.1093/nar/gkx1098.

Zhao, Peng and Bin Yu (2006). "On model selection consistency of Lasso". In: *Journal of Machine learning research* 7.11, pp. 2541–2563. URL: http://www.jmlr.org/papers/volume7/zhao06a/zhao06a.pdf.