UNIVERSITY OF CAMBRIDGE
Department of Public Health and Primary Care

Medical Research Council

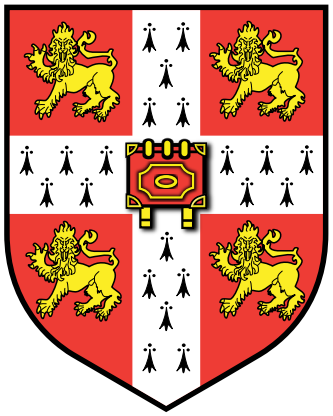
HDRUK
Health Data Research UK

Baker
HEART & DIABETES INSTITUTE

INFERENCE FRAMEWORKS IN COMPUTATIONAL BIOLOGY: FROM PROTEIN-PROTEIN INTERACTION NETWORKS USING MACHINE LEARNING TO CARBON FOOTPRINT ESTIMATION

Loïc Lannelongue

Clare Hall, University of Cambridge



Supervised by Professor Michael Inouye

Submitted in March 2022

This thesis is submitted for the degree of Doctor of Philosophy

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Abstract

Inference frameworks in computational biology: from protein-protein interaction networks using machine learning to carbon footprint estimation.

Loïc Lannelongue

Protein-protein interactions (PPIs) are essential to understanding biological pathways and their roles in development and disease. Computational tools have been successful at predicting PPIs *in silico*, but the lack of consistent and reliable frameworks for this task has led to network models that are difficult to compare and, overall, a low level of trust in the predicted PPIs. To better understand the underlying mechanisms underpinning these models, I designed B4PPI, an open-source framework for benchmarking that accounts for a range of biological and statistical pitfalls while facilitating reproducibility. I use B4PPI to shed light on the impact of network topology and understand how different algorithms deal with highly connected proteins. By studying functional genomics-based and sequence-based models (two of the most popular approaches) on human PPIs, I show their complementarity as the former performs best on lone proteins while the latter specialises in interactions involving hubs. I also show that algorithm design has little impact on performance with functional genomic data. I replicate these results between human and yeast data and demonstrate that models using functional genomics are better suited to PPI prediction across species. These analyses also highlight disparities in computing resources needed to train the prediction tools; some models run within seconds while others need hours. Longer runtimes require more energy and are responsible for more greenhouse gas emissions. Being able to quantify this impact is crucial as climate change profoundly affects nearly all aspects of life on earth, including human societies, economies and health. Various human activities are responsible for significant greenhouse gas emissions, including data centres and other sources of large-scale computation. Although many important scientific milestones have been achieved thanks to the development of high-performance computing, the resultant environmental impact has been underappreciated. I present a methodological framework to estimate the carbon footprint of any computational task in a standardised and reliable way, and metrics to contextualise greenhouse gas emissions are defined. I develop a freely available online tool, Green Algorithms, which enables a user

to estimate and report the carbon footprint of their computation (available at www.green-algorithms.org). The tool easily integrates with computational processes as it requires minimal information and does not interfere with existing code while also accounting for a broad range of hardware configurations. Finally, I quantify the greenhouse gas emissions of algorithms used for particle physics simulations, weather forecasts, natural language processing and a wide range of bioinformatic tools.

With rapidly increasing amounts of sequence and functional genomics data, this work on protein interactions provides a systematic foundation for future construction, comparison and application of PPI networks. It also integrates essential metrics of environmental efficiency developed by the Green Algorithms project, a simple generalisable framework and a freely available tool to quantify the carbon footprint of nearly any computation. This work also elucidates the carbon footprint of common analyses in bioinformatics and provides recommendations to empower scientists to move toward greener research.

Acknowledgements

Like all PhDs, this one took many unexpected turns, and the final body of work is far from what I envisioned at the start. This is the beauty and fun of research, and I am glad I had the chance to explore different topics as I did. The journey would not have been the same without the support I had over the past 3.5 years, and for that, I am infinitely grateful to my family, friends and colleagues. I will probably forget some people here, and I apologise for that (I will blame the frenzy of the last few weeks of a PhD).

Starting with Mike, for his unwavering support throughout the past years. I'm especially grateful that he let me pursue whatever avenue I found interesting, even when it would seem to be a waste of time. He set us off on the Green Algorithms project, and always supported it, even when the project grew bigger and more time consuming. We had a vision of how important green computing was for the research community, and I'm glad uncooperating reviewers and resisting institutions didn't stop us. I have also been lucky to collaborate with many brilliant scientists in the CBSGI and CEU, who have always provided thoughtful feedback and been friendly sounding boards for my ideas. I can't list everyone, but particular mentions go to Sam, Scott, Yu Xu, Jonathan, Guillaume, Petar, and of course Jason with whom I shared the Green Algorithms adventure.

(French version below) Anyone who went through a PhD knows that the support of family is essential, and for that too, I have been extremely lucky. Even though what I do often sounds like gibberish and they live across the Channel (which, with a pandemic, proves to be a bigger hurdle than anticipated), they have always been keen followers of my progress. It would be too long to list everything I am thankful for, but in no particular order, to my mum for pushing me to do the best I can and my dad for lengthy nerd chats about our latest gadgets and technological adventures. To Mamie Gi for having me for four (stressful) years while I made my way through prépa and ENSAE and setting me up so well for what came next. To Gralyne for continuous encouragement and some pre-exams boosts (it was a tortuous journey to biomedicine, but I'm there!). And of course, Papily, who inspired me every step of the way, from endless science talks as a kid to following in his footsteps at St Louis and pursuing a PhD; I couldn't have hoped for a better role model. I also have a special thought for Papi Pierre (I followed the family tradition in computer science!) and Mamilyne who missed the end of the journey but planted the seeds that made it possible. I should also mention Baptiste, who took over brilliantly as family tech support when I left (and doing a much better job than me!), Louise, Audrey, Aline, Muriel and Manu… I'm lucky to be surrounded by so many bright and supportive people.

En français. Tous ceux qui sont passés par un doctorat savent que le support de la famille est essentiel, et là aussi, j'ai eu beaucoup de chance. Ce que je fais au quotidien est souvent du charabia et ils habitent tous de l'autre côté de la Manche (ce qui, pendant une pandémie, s'est révélé être un obstacle plus important qu'anticipé), mais ils ont toujours été des spectateurs attentifs de mes progrès. Il serait trop long de lister tout ce pour quoi je suis

reconnaissant, mais, sans ordre particulier : Maman qui m'a encouragé à faire de mon mieux et à repousser les limites, Papa avec qui j'ai des discussions sans fin à propos de nos derniers gadgets et aventures technologiques. Mamie Gi, qui m'a supporté (dans tous les sens du terme) pendant quatre ans de prépa et d'ENSAE, sans quoi je ne serais certainement pas à Cambridge aujourd'hui. Gralyne, pour ses encouragements et son aide relaxante avant les examens (le chemin vers médecine a été long, mais j'ai fini par y arriver !). Et bien sûr, Papily, qui m'a inspiré à chaque étape, à commencer enfant par de longues discussions à propos du monde et des sciences, jusqu'à suivre ses traces à St Louis puis ce doctorat ; je n'aurais pas pu espérer un meilleur modèle à suivre. J'ai aussi une pensée particulière pour Papi Pierre (l'honneur est sauf, je continue la tradition familiale dans l'informatique !) et Mamilyne, qui ont manqué la fin de l'aventure, mais qui ont semé les graines rendant tout cela possible. Je dois aussi mentionner Baptiste, qui a parfaitement repris le rôle de support informatique familial (bien mieux que moi !), Louise, Audrey, Aline, Muriel et Manu… J'ai de la chance d'avoir autant de personnes brillantes et encourageantes autour de moi.

Emily, of course, who has been by my side for most of the journey (even proofreading some sections as I'm writing these lines) and has been an incredible support throughout. From Pentathlon training camps in Cambridge to Ross-on-Wye, it would have been a very different few years without her, and late-night on-calls have been welcome distractions from research. Cheers to joining the Dr&Dr club! Considering his contribution to my overall sanity, and the numerous bugs I've solved on walks, it seems only fitting to thank Mr Marty McFly as well here, an outstanding four-legged writing companion, feet warmer and pub partner. I'm also grateful to Sue and Ade for helping me decipher some British traditions and taking on pet sitting duties in the final stretch of writing up this thesis.

Pentathlon has been a big part of my life for the past four years (too big a part, some would say), but I am grateful for the people I met over the years, both in OUMPA (yes, I know) and CUMPC. Too many people to list everyone here, but a special shout out for a lot of sporting fun and incredible friendships through the years to Steph, Tads, Pops, Fred, Anna, Lise, Naoki, Tancred, Sam… and many others.

It takes a village, and many more people deserve to be acknowledged here; Kim for many late-night rants about academia, Marie for moral support and some much-needed breaks back home and Philippe, who introduced me to machine learning many years ago, for too rare dinners at Montparnasse talking endlessly about machine learning. Also come to mind my professors at St Louis, old companions from St Louis and ENSAE (shout out to Maxime, Jean-Michel, Aldéric, Morgane…) and all the friends from Oxford and Cambridge.

It has been a long and unexpected journey since I started uni, but I'm grateful to everyone who played a part in it. Now onto the next chapter!

Contributions to collaborative work

My contribution to collaborative work is described in more detail at the start of each chapter. The work on protein-protein interactions (Chapters 2 and 3) is my own work, with feedback and suggestion from Michael Inouye and colleagues. The Green Algorithms project (Chapter 4) is a collaboration with Jason Grealey and Michael Inouye; I led the development of the framework and the online calculator and contributed to the survey on the impact of bioinformatics.

Contents

INTRODUCTION

Robust, open-source and standardised frameworks are essential to modern open science. They ensure that published results can be trusted, are reproducible and that performance can be compared between publications. This enhanced trustworthiness also increases the uptake of published models while reducing the burden of future benchmarks.

Proteins are the functional units of a cell, responsible for numerous operations necessary to survival such as catalysing (i.e. accelerating) chemical reactions, enabling gene expression and facilitating information flows. Because of their central role in an organism's metabolism, malfunctioning proteins are almost always responsible for diseases [1], [2], and are therefore a target of choice for therapeutic treatments [3]–[6]. As demonstrated, for example, by the discovery of the role of *Htt* and *GIT1* in Huntington's disease [7], a complete mapping of proteins' roles would enable unprecedented progress in the treatment of diseases. However, such mapping is made difficult by the complex relationship between proteins and biological functions. Indeed, the proteome is constantly changing and, by combining with different proteins at different times, a protein can be involved in numerous functions. These protein-protein interactions (PPIs) are the cornerstone of biological processes and better understanding them is instrumental to successfully mapping proteins to functions. However, the diversity of PPIs, from permanent complexes to transient associations, make the task difficult. Another challenge is the size of the PPI network. Although the exact size is still unknown today, it is estimated that, in humans, there are between 20,000 protein-coding genes and several millions of proteoforms [8]; even when only including the 20,000 verified proteins in UniProt, it still represents 200,000,000 potential pairwise interactions or up to 2.6×10^{19} 5-way combinations.

Experimental methods exist to observe interacting proteins, for example Yeast Two-Hybrid and mass spectrometry, but these tend to suffer from different biases and even high throughput experiments cannot survey the entire landscape of protein pairs. Computational methods can address the issue of scalability and measurement bias. Based on a dataset of known interactions (obtained from experiments), an algorithm can learn

some patterns connecting what is known about each protein and the probability of interaction. How to curate the gold standard of interactions and the type of information used to characterise each protein is where approaches differ. Some methods are based on functional genomic (FG) information about each protein, while others only use amino acid sequences. There are also methods based on the topology of existing networks, 3D structures or phylogeny.

Despite a wealth of algorithms available, the inference mechanisms are still poorly understood and discrepancies between predictions remain unexplained. It is still unclear where the differences between the various approaches lie and how to find the right method for a task. This inefficient situation can be partly explained by the lack of a unified framework. Models are trained on different datasets, with different training and validation strategies, and different metrics. Unfortunately, the complexity of the biology underlying protein interactions is source of numerous pitfalls which render many approaches unusable because of significant flaws in their training or evaluation. A robust framework would facilitate the development of new models and their comparison to previous tools. It would also ensure that prediction models can be trusted and would facilitate the utilisation of inferred PPIs for downstream analysis.

Such inference frameworks are built on a number of components: the data used for training, how to set aside observations for validation and testing, what features to use as input and a range of appropriate evaluation metrics. The latter in particular warrants special attention. Although properly quantifying accuracy is important, the huge computing requirements of modern algorithms call for more metrics of efficiency, energy usage and carbon footprint [9]. Indeed, the concentration of greenhouse gases (GHG) in the atmosphere dramatically influences climate change with both global and locally focused consequences, such as rising sea levels, devastating wildfires in Australia, extreme typhoons in the Pacific, severe droughts across Africa, as well as repercussions for human health. Every year, it is estimated that 4.2 million deaths are caused by ambient air pollution alone, while 91% of the world's population suffers from air quality below the World Health Organisation standards [10]. Global warming results in further consequences on human health, economy and society: the daily population exposure to wildfires has increased in 77% of countries [11], 133.6 billion potential work hours were lost to high

temperatures in 2018, and with 220 million heatwave exposures, vulnerable populations (aged 65 and older) are affected at an unprecedented level. Power generation, through the associated emissions of GHGs, is one of the causes of outdoor air pollution and climate change, and the electricity needs of data centres contribute to this. Especially for scientists striving to improve human health in the first place, it is our responsibility to factor in the environmental impact of our work. However, no framework or tool existed for that either and as a result, this has been mostly overseen in modern science. In the words of Dr Richard Horton, Editor-in-Chief of The Lancet [12]:

> *"The climate emergency that we're facing today is the most important existential crisis facing the human species, and since medicine is all about protecting and strengthening the human species it should be absolutely foundational to the practice of what we do every single day."*

This thesis presents unified frameworks for PPI prediction and carbon footprint estimation. Chapter 1 is a literature review of these two topics and shows the urgent need for such frameworks. In Chapter 2, I introduce B4PPI, a benchmarking standard for *in silico* prediction of PPIs. In Chapter 3, I work within this framework to dive into popular approaches for PPI prediction to better understand what models learn and how predictions are made. I also show the impact of network topology on predictions and how these results reproduce in different organisms. Chapter 4 is dedicated to the Green Algorithms project that helps quantifying the carbon footprint of computations for tasks such as PPI prediction. I present the online tool and the result of investigations into the impact of algorithms in physics, weather forecasting and bioinformatics. I then explore how this work on green computing could help move the field of computational research towards improved sustainability.

References

[1] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, "Network medicine: a network-based approach to human disease," *Nat Rev Genet*, vol. 12, no. 1, Art. no. 1, Jan. 2011, doi: 10.1038/nrg2918.

[2] M. Y. Hein, K. Sharma, J. Cox, and M. Mann, "Proteomic Analysis of Cellular Systems," in *Handbook of Systems Biology*, Elsevier, 2013, pp. 3–25. doi: 10.1016/B978-0-12-385944-0.00001-0.

[3] H. Ruffner, A. Bauer, and T. Bouwmeester, "Human protein-protein interaction networks and the value for drug discovery," *Drug Discov Today*, vol. 12, no. 17–18, pp. 709–716, Sep. 2007, doi: 10.1016/j.drudis.2007.07.011.

[4] M. R. Arkin, Y. Tang, and J. A. Wells, "Small-molecule inhibitors of protein-protein interactions: progressing toward the reality," *Chem Biol*, vol. 21, no. 9, pp. 1102–1114, Sep. 2014, doi: 10.1016/j.chembiol.2014.09.001.

[5] M. Bakail and F. Ochsenbein, "Targeting protein–protein interactions, a wide open field for drug design," *Comptes Rendus Chimie*, vol. 19, no. 1–2, pp. 19–27, Jan. 2016, doi: 10.1016/j.crci.2015.12.004.

[6] S. Rapposelli, E. Gaudio, F. Bertozzi, and S. Gul, "Editorial: Protein–Protein Interactions: Drug Discovery for the Future," *Front. Chem.*, vol. 9, p. 811190, Nov. 2021, doi: 10.3389/fchem.2021.811190.

[7] H. Goehler *et al.*, "A Protein Interaction Network Links GIT1, an Enhancer of Huntingtin Aggregation, to Huntington's Disease," *Molecular Cell*, vol. 15, no. 6, pp. 853–865, Sep. 2004, doi: 10.1016/j.molcel.2004.09.016.

[8] R. Aebersold *et al.*, "How many human proteoforms are there?," *Nat Chem Biol*, vol. 14, no. 3, pp. 206–214, Mar. 2018, doi: 10.1038/nchembio.2576.

[9] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *arXiv:1907.10597 [cs, stat]*, Aug. 2019, Accessed: Jan. 07, 2020. [Online]. Available: http://arxiv.org/abs/1907.10597

[10] "Air pollution," *World Health Organisation*. https://www.who.int/westernpacific/health-topics/air-pollution (accessed Oct. 17, 2020).

[11] N. Watts *et al.*, "The 2019 report of The Lancet Countdown on health and climate change: ensuring that the health of a child born today is not defined by a changing climate," *The Lancet*, vol. 394, no. 10211, pp. 1836–1878, Nov. 2019, doi: 10.1016/S0140-6736(19)32596-6.

[12] "NEW WORK: Richard Horton / The Lancet – Rubber Republic." https://www.rubberrepublic.com/new-work-richard-horton-the-lancet/ (accessed Mar. 26, 2020).

CHAPTER 1

LITERATURE REVIEW

Contents

Introduction

Machine learning is one of the central themes of this thesis; I explore how it can and should be used for predicting protein-protein interactions, and I estimate the carbon footprint of training such algorithms. To lay the foundations for the following chapters, I start with a brief introduction to machine learning. What is it? What does *training* mean? And how do the most popular algorithms work? Following this, I discuss protein-protein interactions, their role and what tools exist to detect them, both experimental and computational methods. The energy needed to train such computational tools have raised some concerns so I conclude this chapter with the existing literature on the carbon footprint of IT and computational science. This literature review gives a perspective on the state-of-the-art in these different fields and shows the need for unified frameworks, both for PPI inference and for carbon footprint estimation.

[1]

[1] From www.xkcd.com

Some background on machine learning

Despite the futuristic aspect of artificial intelligence (AI) and machine learning, the field is firmly grounded in statistics and most tools have been known for decades. There is an ongoing debate about what belongs to statistics and what belongs to machine learning, but I tend to believe that, considering the overlap, this is mostly a vocabulary issue and the audience or the topic usually motivates the use of one word or the other. Perhaps a better word would be *statistical learning* [1] to acknowledge the important contribution of statistics to the field of AI. Regardless of the boundaries between fields, it may be best to distinguish between statistics and machine learning based on what the model is trying to achieve: statistical models are generally used to explain a relationship between variables and outcomes, while machine learning focuses on making predictions on new data. The tools may be the same, but how they are trained (or *fitted*) and evaluated differ (although even here, a significant overlap exists).

I have mentioned machine learning, artificial intelligence and deep learning, but I haven't explained how they relate. *Artificial intelligence* represents any attempts at mimicking the human decision-making process. Two types of AI can be distinguished, *symbolic AI* and *machine learning*. The former corresponds to human-made decision rules, for example, a decision tree to standardise diagnostics in hospitals. How to come to a conclusion is entirely decided by the individual drawing the tree. Machine learning, on the other hand, relies on rules that are inferred mathematically: based on data, the model finds, or learns, the best decision rule to optimise some performance metrics. Deep learning is a subfield of machine learning that relies on neural networks to handle high-dimensional data (such as images or sounds).

A brief history

The Dartmouth Summer Research Project on Artificial Intelligence in 1956 is often considered the birthplace of AI, and the project proposal submitted by John McCarthy in 1955 shows the unwavering trust they had in computers.

> *"...that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use*

language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."

- McCarty et al., 1955 [2] -

The term *machine learning* is attributed to Arthur Samuel who coined it shortly after the Dartmouth meeting and is reported to have defined it as a "field of study that gives computers the ability to learn without being explicitly programmed". There are no sources for this widely used citation, but Samuel conveys a similar message in a 1959 publication entitled "Some Studies in Machine Learning Using the Game of Checkers" [3].

"…the programming of a digital computer to behave in a way which, if done by human beings or animals, would be described as involving the process of learning."

- Arthur Samuel, 1959 [3] -

A more mathematical formulation was laid out by Tom Mitchell in 1997 and is now used in most introductory courses about AI.

"[Machine learning is] any computer program that improves its performance at some task through experience. Put more precisely, a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

- Tom Mitchell, 1997 [4] –

In 2012, Kevin Murphy offered an alternative definition that focuses more on how to achieve prediction rather than the nature of machine learning. I find this last definition particularly interesting as it demystifies machine learning by presenting it more like a statistical tool and less like a semi-conscious machine.

"To develop methods that can automatically detect patterns in data, and then to use the uncovered patterns to predict future data or other outcomes of interest."

- Kevin Murphy, 2012 [5] –

The different types of machine learning

Machine learning techniques can be divided based on the nature of the data available for training. *Supervised* learning tools are for when a set of labelled data (i.e. with known outcomes) is available; the labelled dataset is usually called gold standard or reference set. In the absence of such labels, *unsupervised* learning techniques can be used; classic examples of unsupervised learning include clustering and dimension reduction. Semi-supervised learning is the intermediary situation where some labelled data exist but not enough to discard unlabelled observations. Other categories exist, such as reinforcement learning [6], but I will focus on supervised learning, which is most relevant for PPI prediction. When the outcome to predict is continuous, *regression* models are used, while *classification* tools are used for categorical labels.

How to train models

Most supervised algorithms are built on the same underlying principles. A dataset comprises features, also called independent variables or regressors, and a label, or dependent variable or regressand in statistics. A model is simply a function f that depends on a set of parameters w and takes as input the features x. The output $\hat{y}$ is the predicted value.

$$\hat{y} = f(x; w) \quad (1)$$

Training or *fitting* a model means (automatically) finding the parameters w that minimise the error between the predicted values $\hat{y}$ and the true labels y on the training set. This error is measured by a *loss function* (also called *cost* or *objective function*); depending on the situation, a wide range of loss functions can be used (Table 1). Parameters that are not inferred by the algorithm, such as the form of the function f, are called *hyperparameters*. The number of parameters can vary greatly between models, from the number of features in the case of linear regressions to ~100 in Support Vector Machines, easily ~10,000 for random forests and even several million for neural networks (more on these algorithms below).

Table 1: Examples of loss functions.

Name	Loss function	Examples of usage
Quadratic/L2 loss	$L(y,\hat{y}) = (y-\hat{y})^2$	Linear regression (regression)
ε-insensitive (Vapnik) loss	$L(y,\hat{y}) = \max\{\lvert y-\hat{y}\rvert - \epsilon, 0\}$	SVM (regression)
0-1 / misclassification loss	$L(y,\hat{y}) = \mathbb{1}_{y\hat{y}>0}$	Naïve Bayes (classification)
Hinge loss	$L(y,\hat{y}) = (1-y\hat{y})^+$	SVM (classification)
Exponential loss	$L(y,\hat{y}) = e^{-y\hat{y}}$	AdaBoost (classification)
Logistics loss	$L(y,\hat{y}) = \log(1+e^{-y\hat{y}})$	Logistic regression (classification)

Although the minimum of the loss function, i.e. the optimal parameters, can sometimes be found analytically for simple examples, it is more often done using the gradient descent algorithm. Gradient descent, sometimes called mountain climbing, finds the minimum of a function iteratively by starting at a random point in the function's domain, calculating the slope, or gradient, at this point and moving in the direction of the greatest decrease.

A classic example is the linear regression. Although a popular statistical tool, it can also be considered a machine learning model when used for prediction. The form of the function is shown below, where $w = (w_0, w_1, \dots, w_p)$ are the learned parameters, which are found using a squared loss (i.e. the well-known least-square estimator).

$$f(x;w) = w_0 + w_1 \cdot x_1 + \cdots + w_p \cdot x_p \qquad (2)$$

Because the goal of machine learning is to make predictions on new data, it is important to make sure that the patterns learned are not specific to one dataset – in this case, the model is said to *overfit* – but rather are generalisable to similar data points. To evaluate generalisability, the dataset is divided between a training set, used to learn the parameters, and a testing set, used to evaluate the model on a previously unseen set of examples.

Classic machine learning algorithms

Linear models have been known for a long time, from linear regressions to linear discriminant analyses proposed in 1936, logistic regressions in the 1940s and generalised linear models in the 1970s [1]. For some time, these were the main prediction tools available due to limitations in computing power. With the improvement of hardware, more complex non-linear models became available, such as classification and regression trees in

the 1980s and support vector machines in the 1990s [1]. I won't detail how each model works but rather give an overview of the most popular ones (using, among other textbooks, [7]).

The well-known linear regression only allows for simple relationships between features and an outcome. Polynomial terms (i.e. combinations of the existing features) can be added to the model to allow for more complex relationships. This added flexibility comes with a risk of overfitting (the curve would fit the training data too closely, at the expense of generalisability). Regularised regressions have been introduced to avoid this. The quadratic loss is modified to include a penalty on the magnitude of the parameters, controlled by a hyperparameter λ **(3)**. Ridge regression uses an L^2 penalty [8], Lasso uses an L^1 penalty that promotes sparsity in the coefficients [9], and ElasticNet combines both penalties [10].

$$L(y, \hat{y}) = \sum_i (\hat{y_i} - y)^2 + \lambda \parallel w \parallel \quad \textbf{(3)}$$

The Naïve Bayes Classifier [11] assumes that the features are independent to facilitate the calculation of probabilities in the Bayes formula (hence "naïve"). Although the probability $P(Y|X)$ of an outcome Y based on some features X is hard to estimate, the probability $P(X|Y)$ can be obtained by counting occurrences in the training set, provided the features are independent. This assumption is unrealistic, but has been shown that the predictor can still be optimal when independence does not hold [12].

Regression and classification trees are simple ways to create highly non-linear decision rules and are the building blocks of many popular machine learning algorithms. A decision tree iteratively divides the population into subgroups, intending to create homogenous categories to help prediction. In the common case of a binary tree (a node has two child nodes), a feature x_i and a value θ_i are chosen at each step and the population is divided based on whether x_i is greater or smaller than θ_i. How the splitting features and values are chosen is what differentiates the different tree algorithms; for example, CART [13] uses the Gini impurity index and C4.5 [14] uses information gain. The rules created by these decision trees are easy to understand and visualise but are also prone to overfitting and have limited predictive power [15], which motivates ensemble tree methods.

Ensemble methods aggregate several (sometimes thousands) weak learners into a single better predictor. One of the limitations of a single tree is the risk of overfitting, which means that these predictors have a high variance. *Bagging* (for bootstrap aggregation) [16] is an efficient way to reduce the variance of a statistical predictor by repeatedly sampling the training set to create k different datasets (i.e. *bootstrapping*) and training a different predictor on each one. Although each predictor only has a partial understanding of the problem, aggregating their predictions creates a more robust model with smaller variance, an approach particularly efficient with trees [15]. The *Random Forest* algorithm [17] improves further on bagged trees by training each individual tree on a random subset of features (i.e. no tree has access to all the features nor all the observations), which further decorrelates the trees and improves the aggregated predictor. For bagging and random forest, all the trees are built independently, and in some unlucky cases, they may all share the same flaws. To address this, *boosting* also creates bootstrap samples but trains the individual learners sequentially, each learner focusing on the shortcomings of the previous one. Schapire showed in 1990 that any set of weak learners could be boosted into a strong learner [18], and with Freund proposed the AdaBoost algorithm in 1999 [19], said to be "the best off-the-shelf classifier in the world" by Breiman. *Gradient Boosting* improves further on this by combining boosting and gradient descent, turning the problem into a numerical optimisation one that minimises the loss function by differentiation, which allows for arbitrary loss functions [20]. *XGBoost* (standing for extreme gradient boosting), a computationally efficient implementation of gradient boosting, was released in 2016 [21] and has rapidly outperformed other models in machine learning competitions [22], [23].

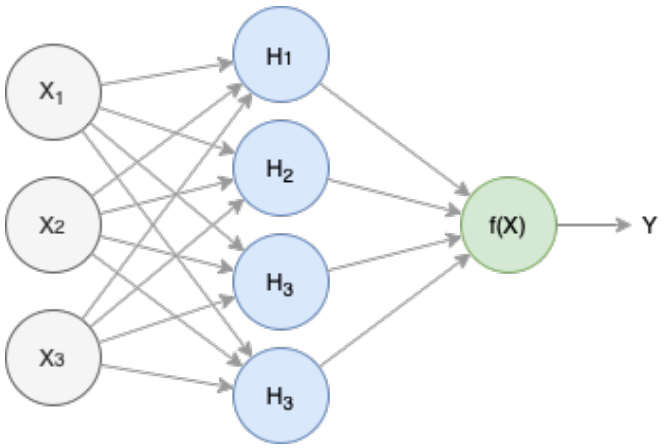
Tree-based models are not the only non-linear models available. Support Vector Machines (SVMs) were initially developed for classification in the 1990s [24]. SVM is built on two key concepts: margin maximisation and the kernel trick. The maximal margin classifier is a linear binary classifier that separates the observations with a separating hyperplane (e.g. a straight line in 2D or a plane in 3D). Provided the data is linearly separable, infinitely many hyperplanes exist, and this classifier chooses the one that maximises the margins between the data points and the decision boundary [25]. To handle problems that are not linearly separable, the support vector classifier allows for misclassifications to keep satisfactory margins (hence its name of soft margin classifier). However, the decision rule remains

linear, which is often not appropriate. SVM introduces non-linearity by expanding the feature space (e.g. considering the squares of all features, cross-products etc.) using *kernels*, which are a computationally efficient way to do so ([25] can be consulted for the detailed mathematical foundations of the kernel trick).

Neural networks are the cornerstone of deep learning; the first model of an artificial neuron, the *perceptron*, was proposed by Rosenblatt in 1958 [26]. However, the lack of computational power rendered these theoretical models impractical and they fell out of favour for over twenty years. It was not until the late 1980s that deep learning became popular again with successes such as handwritten zip codes recognition by Yann Le Cun [27]. The first type of neural network is the *multi-layer perceptron*, similar to the one suggested by Rosenblatt. It is composed of an input layer with one neuron per feature which feeds into one or several hidden layers (Figure 1.a). Each neuron on this hidden layer is a non-linear transformation of a linear combination of the input features (Figure 1.b). This non-linear transformation is called an *activation function*, and while early models were built with the sigmoid function, the *ReLU* (rectified linear unit) is now the preferred option [28]. One of the strengths of neural networks is their flexibility; by adjusting the number of hidden layers and the size of each layer, the algorithm can model arbitrarily complex phenomena. Parameters are found using *backpropagation* [29], an algorithm that calculates the gradient of the loss function with respect to each parameter with the chain rule to propagate gradients and find a minimum. A wide range of different network architectures have been then developed over the years to capture specific phenomena; for example, *Convolutional Neural Networks* (CNN) [27] are particularly suited to image recognition and, to a certain extent, mimic how the human brain processes images by first extracting shapes and edges. A lot of data is sequential, natural language for example, and *Recurrent Neural Networks* (RNN) have been developed to adjust to that. RNNs apply the same weights and activation to each element of a sequence successively while remembering previous elements; this way, these networks can be used with an input of arbitrary length and make connections between elements [30], [31].

To understand why such models, from basic logistic regressions to more sophisticated recurrent neural networks, are key to understanding protein-protein interactions, I should first provide some background on proteins and the tools available to understand them.

(a)



(b)

$$H_k = g\left(w_{k0} + \sum_j w_{kj} X_j\right)$$

Figure 1: Multi-layer perceptron and activation function.
(a) The input layer (i.e. the features) is in grey, the hidden layer in blue and the output layer in green.
(b) g is the non-linear activation function and w are the weights learned by the network.

Protein-protein interactions

The so-called *Central Dogma* of molecular biology describes how information flows between DNA, RNA, and proteins. The general transfers, occurring normally, focus on how DNA can replicate and be used to build messenger RNA (mRNA) through transcription; in turn, mRNA can be used to synthesise proteins (translation). Different mechanisms increase the diversity of gene products. One gene can code for multiple proteins through alternative splicing: by including and excluding different exons (gene's segments), different mRNA molecules can be created from a single gene, resulting in a multitude of proteins. Moreover, proteins can undergo post-translation modifications (PTMs) which also increase the diversity of mature proteins. PTMs can introduce new functional groups or modify existing ones, e.g. phosphorylation to regulate enzymes activity, glycosylation to promote protein folding, and lipidation, related to membrane proteins. Beside mRNA and proteins, other gene products contribute to gene expression, such as tRNA that facilitates translation or miRNA and siRNA that repress it. Our understanding of information flows from DNA to proteins is also complicated by special transfers, occurring only under specific conditions (mainly in viruses or laboratories), e.g. RNA replication or DNA being synthesised from RNA.

System perspectives on biology became increasingly popular to better comprehend the full complexity of living organisms. Modern developments in imaging, computational biology and high-throughput proteomics have provided unprecedented insights into a dynamic view of the different biological elements of a cell, their location, how they physically and functionally interact and how they react to environmental modifications [32].

With the complete sequence of the human genome finally published [33], twenty years after the end of the Human Genome Project, scientists now have a complete inventory of genetic instructions. This would have been sufficient to fully understand human biology under the influential 1941 one gene-one enzyme hypothesis by Beadle et al. [34], [35]. However, this theory was rapidly proven to be too simple [36] and replaced with the now-disputed one gene-one polypeptide model, which highlighted the importance of also understanding the biological pathways triggered and regulated by these genes. System-level analyses were first performed by looking at mRNA to better understand where gene products were expressed (transcriptomics studies). These revealed that, unsurprisingly, the

mapping from genomic sequences to RNA is extremely complex. Besides, the relationship between RNA and biological functions remains intricate as the transcriptome is only an intermediary stage to pass information from the genome to the functional units, the proteins. Proteins – and protein-protein interactions – are the linchpins of systems biology and come in a variety of ways. Some proteins are enzymes catalysing thousands of chemical reactions and orchestrating gene expression, others are antibodies involved in the immune response, transmit signals between cells and tissues, carry other molecules or provide structural support [37], [38].

> *"Clearly, the proteome of the cell is the most complex and functionally most relevant level of cellular regulation and function."*
> *- Hein et al., Handbook of Systems Biology [38] -*

Proteins are composed of a primary structure of amino acids (AA) derived from the nucleotide sequence of the corresponding protein-coding gene. This structure then folds into a 3D tertiary structure thanks to chemical links such as hydrogen bonds and van der Waals forces [39], followed by eventual PTMs. The tertiary structure enables different proteins to bond together temporarily or permanently to form transient interactions and stable protein complexes. By enabling these complexes, PPIs are essential to most biological processes, and disrupted PPIs are both the causes of many diseases and the basis for treatments. Over the years, a lot of efforts have been put into understanding them, first using experimental tools to identify interacting proteins, either *in vitro* or *in vivo*, and more recently using computational models to address the limitations of experimental approaches.

Experimental methods

Experimental methods to identify PPIs can be separated between high and low throughput, or categorised by the type of interactions they identify (e.g. direct, N-ary or functional interactions). Here, I briefly cover the most common approaches, and for a more detailed technical review of each method, Shoemaker et al. and Snider et al. [40], [41] can be consulted.

The most accurate way to characterise PPIs is by observing the interactions at the atomic level, using for example X-ray crystallography [42] or various spectroscopy methods [43],

[44]. Atomic force microscopy [45] can even study a single molecule. These methods can identify both direct binary interactions and protein complexes in a detailed manner but are low throughput.

Phage display, proposed in 1985 [46], uses bacteriophages to connect genotype and phenotype and can identify direct PPIs *in vitro* in a high throughput manner. Yeast two-hybrid (Y2H) [47], [48] improved on it and marked the beginning of large-scale databases on PPIs. It also identifies direct PPIs, using this time specific properties of yeasts' transcription factors and the activation of reporter gene expression [49]. Y2H can identify both stable and transient interactions, and because it is *in vivo*, proteins are more likely to be in their native conformations, which leads to increased accuracy [47], [49]. However, differences in protein folding and post-translational modifications can occur when studying organisms other than yeast [40]. Moreover, Y2H tends to have high precision but low recall (few false positives but many false negatives), resulting in an incomplete map of the network [50]. Y2H is also biased towards non-specific interactions [51] and suffers from drawbacks due to the experiment itself, e.g. targets cannot be proteins that initiate transcription by themselves [40] or proteins confined to specific cellular locations such as membrane proteins [41], [52]. Membrane Yeast two-hybrid (MYTH) [53], [54] was specifically designed to address this, using ubiquitin proteins.

Both phage display and Y2H identify binary interactions, i.e. between just two proteins. Biochemical methods, based on mass-spectrometry in particular, are used to identify multi-protein complexes [55]. Mass spectrometry can identify polypeptide sequences by converting proteins into ions. The purification of protein complexes is an essential step of the process, and Tandem Affinity Purification (TAP-MS) [56], [57] is commonly used for this [38].

Y2H and TAP-MS in particular have led to the first large-scale interactome databases for model organisms such as *Saccharomyces cerevisiae* [58], [59]. The two methods have also been shown to be complementary, with very limited overlap in identified interactions [60], [61]. For example, Y2H can detect weak or transient interactions which are under-represented in AP-MS datasets.

Although Y2H and TAP-MS are the two most common methods, other tools are available. Correlation in gene expression levels has been used to infer PPIs and is particularly relevant for permanent complexes, such as ribosomes and proteasomes [62]. Similarly, synthetic lethality combined with DNA microarrays has been used to study gene interactions and infer PPIs [63], [64].

Several databases have been created to aggregate the information collected. For example, DIP [65], MINT [66], BioGRID [67], STRING [68] and HPRD [69] (focused on human proteins). Most of these databases are regrouped within the IMEx consortium [70] and feed into the IntAct database [71] managed by EMBL-EBI to limit duplication efforts, e.g. MINT and DIP.

These experimental methods suffer from common drawbacks, such as sampling bias (i.e. focusing on well-studied proteins) [72], measurement bias (due to the technical limitations of a technique), and biases towards high abundance proteins or particular cellular localisations [73]. Some progress has been made in this respect, for example with the systematic mapping of PPIs to avoid sampling bias (e.g. HuRI [74] and BioPlex [75]) and advances in biotechnologies to address measurement bias [38]. However, the lack of overlap between different experiments [73], and the continual discovery of new segments of the interactome, indicates that experimental methods are still non-saturating and different tools are needed to apprehend the PPI network fully [76].

Computational methods

A range of computational models has been developed to infer PPIs over the past twenty years. Similarly to experimental methods, different types of interactions can be detected, from direct physical interactions to co-complex and functional ones. This review will focus on direct interactions as these are more relevant to the research presented in the next chapters. For a more exhaustive and detailed review of existing models, Hu et al. [77] can be consulted.

Most PPI prediction models are built on the same paradigm: based on some characteristics of each protein, an algorithm learns patterns indicative of interaction. These patterns are learned on a training set composed of known interacting and non-interacting protein pairs.

What differentiates models, and dictates the form of the predictor, is the type of features used to characterise each protein.

The first and perhaps most intuitive way to predict direct physical PPIs has been to look for similarities between what is known about each protein. For example, as discussed in the previous section, genetic co-expression was already used to build PPI networks in 2002 [62]. Other functional genomics (FG) information has been leveraged, such as tissue or subcellular localisations and biological functions. Then, similarity measures, e.g. colocalisation or co-expression, can be manually calculated for each protein pair. Although colocalisation between two proteins alone is not a proof of interaction, when combined with sharing common functions and gene expression patterns, these weak indicators make interaction more likely [78]. Once this pre-processing of similarity measures is done, the input dimension for each pair of proteins is the number of FG features, usually ~10. Consequently, FG-based methods are of low dimension and therefore relatively easy to implement with traditional machine learning algorithms, such as logistic regression and Naïve Bayes classifier. In 2003, in one of the seminal publications of the field, Jansen et al. followed this principle and used a Bayesian network to integrate experimental data and FG information (biological functions, co-essentiality and mRNA co-expression) to predict PPIs in yeast. More recently, Jain et al. proposed TCSS that leverages Gene Ontology (GO) annotations [79], and in 2016, Bandyopadhyay et al. used FG annotations combined with SVM to predict PPIs in a range of species (*S. cerevisiae*, *Homo Sapiens*, *Escherichia Coli*, and *Drosophila melanogaster*) [80]. Similarity measures simplify the problem greatly but hide the richness of FG information. TCSS mentioned above leveraged the hierarchical structure of GO annotations, and so did Armean et al. to predict yeast PPIs using entropy-based models and SVM [81]. Two years later, Zhong and Rajapaske also used a graph representation of GO terms to identify missing and spurious PPIs, this time in *H. sapiens*, *Mus Musculus* and *S. cerevisiae* [82].

Phylogeny information, such as coevolution or coinheritance, is often included as well. This was motivated by the observation that the phylogenetic trees of interacting proteins were more similar than expected from the standard molecular clock hypothesis [83], pointing at a shared evolutionary history of interactors. Historically, coevolution has been used to predict PPIs by either focusing on structural constraints and residues with Direct Coupling

Analyses [84] or measuring more general sequence similarities, e.g. mutual information between paralogous proteins [85]. Srinivasan et al., for example, combined these phylogenetic features with co-expression and colocalisation into a non-naïve Bayesian network to predict functional interactions in eleven different microbes [86]. More recently, in 2019, Marmier et al. showed that phylogenetic correlations alone successfully predicted interacting proteins [87].

Another approach to PPI prediction not yet mentioned is to use the topology of existing PPI networks, a method that has been used since the early days of PPI prediction to compensate for the lack of data, despite the risks of suffering from sampling bias. This hypothesis of *guilt-by-association* assumes that interacting proteins should have similar interaction partners [88], [89]; *PIPE*, introduced in 2006 by Pitre et al., is based on this principle [90] and in 2021, Wang et al. [89] demonstrated the robustness of topology-based models. Murakami and Mizugucchi's 2014 paper [91] combines this principle with known PPIs between homologous proteins and the AODE model, a variation of Naïve Bayes [92]. In a recent work published in 2021, Becchetti et al. worked on the same assumption to present an improved version of PIPE, *MPS(T&B)* [93]. Despite these successes, Kovács et al. showed that predicting PPIs based on network topology was complex, and not all network tools – e.g. the triadic closure principle from social sciences – could or should be used [88]. Besides, such approaches suffer from the scale-free nature of PPI networks, as they perform best for densely connected proteins [77].

Primary AA sequences have been long thought to hold the keys to PPI prediction without depending on imperfect experimental annotations. This assumption was first motivated by the observation that differences in interface types were reflected in the AA sequences [94], [95]. It has also been shown that protein binding sites are composed of "hot spots" that are structurally conserved [96] and that close homologs interact in a similar manner [40], [97]. However, sequences of varying lengths are extremely challenging to process with standard machine learning tools. Some early works adjusted to the difficulty of extracting information from sequences by also including FG and topology annotations; in 2005, Ben-Hur and Noble proposed a model combining homologous interactions and the topology of the known network to predict yeast PPIs with a kernel algorithm [98].

As a middle ground between FG annotations and AA sequences, annotated domains and motifs were used as early as 2005, combined with random forests, to predict yeast PPIs [99]. Domains were often combined with FG features as well. In 2007, Scott and Barton used a Bayesian framework integrating FG features (shared domains, motifs and post-translational modifications) to predict human PPIs [100]. In 2015, using the same information as well as physicochemical properties, Kotlyar et al. built *FpClass* for human PPIs, a model based on custom scoring schemes that allowed for more complex interactions between features, e.g. a specific domain and cellular compartment [101]. The success of FpClass highlighted the limitations of considering only similarity measures and the need for more complex modelling.

As for many fields dealing with complex high-dimensional data, such as computer vision or natural language processing (NLP), deep learning has enabled a major leap forward and opened many avenues once hardware made it feasible to train large networks. One of the first applications of deep learning came in 2017 with Sun et al.'s stacked autoencoder model based on primary sequences and tested on a range of species (*H. Sapiens*, *E. coli*, *Drosophila*, *Caenorhabditis elegans*) [102]. Around the same time, Extreme Learning Machines [103] were also used for human PPIs [104]. Siamese networks [105], [106], which had successfully handled paired inputs for image recognition, were then a natural tool for PPI prediction. In 2018, Hashemifar et al. published *DPPI* [107] which combined Siamese networks with CNNs. RNNs such as *Long-Short Term Memory* (LSTM) networks [30] and *Gated Recurrent Units* (GRU) [31] have been hugely successful in NLP, and the parallel between primary protein sequences and natural language led to the application of a number of these tools to PPI prediction. At the same time as DPPI, Li et al. published *DNN-PPI*, a model combining CNN and LSTM networks in a Siamese architecture [108] and tested it on a range of species (*H. sapiens*, *E. coli*, *Drosophila*, *C. elegans*, *M. Musculus*). A year later, Guo and Chen released *iPPI*, which additionally fed AA properties into an LSTM network [109] and Chen et al. released *PIPR* combining GRUs and CNNs [110]. In 2017, with the now-famous article "Attention Is All You Need" [111], attention mechanisms started to replace RNNs in NLP. In 2020, Li et al. released *EnAmDNN* based on ensemble deep learning, CNNs and attention heads to integrate physicochemical properties of AA and primary sequences [112].

After the success of models based on primary sequences, the next natural step was to leverage known tertiary (3D) structures of proteins to infer interactions. However, whereas primary sequences of proteins are well known, tertiary structures are only referenced for a limited subset of the proteome (~35% of confirmed human proteins in UniProt have a PDB structure as of March 2022), although this is starting to change with the AlphaFold database [113]. *PrePPI* was presented in 2012 by Zhang et al. [114] and combined 3D protein structures with FG and phylogeny features into a Naïve Bayes model to predict interactions in *S. cerevisiae* and *H. sapiens*. This was made possible by extensive pre-processing of the protein structures to reduce their dimension. In 2017, Mirabello and Wallner proposed *InterPred* which uses a Random Forest classifier to predict PPIs based on structural data and molecular docking [115]. Around the same time, Zhao et al. predicted interactions between HIV-1 and human proteins by adapting the guilt-by-association hypothesis to protein structures [116].

Accurate PPI prediction models can fill the gaps in existing databases to provide insights into biological pathways and enrich downstream analysis. They can deepen our understanding of biological and genetic processes, and highlight the contribution of PPIs to disease mechanisms. For example, PPI databases have been used to identify ways microbiome bacteria can impact the host health (by targeting human proteins) in relation to inflammatory bowel disease, colorectal cancer, obesity and type 2 diabetes [117]. More exhaustive maps of the interactome can be used to prioritise drug targets and anticipate unwanted side effects. PPIs can even be targeted directly, and drugs acting as PPI modulator have been the subject of intense research efforts in recent years [118]. The potential benefits of imputed PPIs are clear, as demonstrated by databases collating them, such as the PIPs database [119] and STRING [68]. However, for these models to fully reach their potential, their predictions need to be trusted by the community, results have to be reproducible, and it should be clear what approach performs best in each situation. Unfortunately, the lack of standardised best practices prevents this. The models discussed above are trained on different datasets, with different performance metrics, and some suffer from various evaluation biases. Consequently, it is still unknown how these models really perform against one another, which greatly limits their impact on proteomic research.

There has been limited research into addressing this issue. Most publications proposing a new algorithm include some form of comparison, but these benchmarks all include a small (and always different) subset of the literature and a variety of evaluation methods. Recent efforts in this direction have been published, such as the assessment of the International Network Medicine Consortium [89] and a 2022 study by Dunham and Ganapathiraju [120]. The latter benchmarked several models in a standardised way by retraining them on the authors' own datasets to limit discrepancies. They found that most models overestimated their performance greatly in their original publications and a number of algorithms performed worse than random guesses. To ensure that the latest developments in *in silico* PPI prediction are leveraged and inform our understanding of biology and therapeutic targets, it is therefore crucial to expand the efforts towards unified and reliable inference frameworks. Such a framework is the subject of Chapter 2. Besides, to ensure that the community can use the right algorithm for each situation, the mechanistic differences between the different types of prediction models need to be better understood, which is the subject of Chapter 3.

The differences between the prediction models discussed above go beyond accuracy or precision. Some models rely on manually calculated scores and features, and the training part can be performed within minutes on a laptop, while other predictors require training large deep learning architectures for hours on dedicated GPU servers. Most benchmarks overlook these discrepancies, as the quality of the predictions is the main concern. However, with the ongoing climate change crisis, energy is a valuable commodity that should not be wasted; notwithstanding that energy production can have adverse consequences on human health through GHG emissions, negating some of the benefits of bioinformatic discoveries. Intuitively, if there is a large performance gap between a simple tool and an energy-hungry model, the trade-off will likely result in a net benefit for society. To formalise this cost-benefit analysis, we need to reliably estimate the energy needs and carbon footprints of scientific computations; the next section dives into existing studies on this topic.

The environmental impact of computational science

The environmental impact of science is a topic too broad to be covered extensively here; I will therefore focus on computational science, which is most relevant to my work on protein interactions. This review first discusses the evidence supporting the significant carbon footprint of new technologies and data centres and then presents existing studies around the impact of computational research specifically.

The metric *CO_2-equivalent* (CO_2e) is commonly used to summarise the various environmental impacts of an activity and represents the quantity of carbon dioxide with the same global warming impact.

The carbon footprint of IT

The carbon footprint of Information and Communication Technologies (ICT) is a widely studied topic, but the diversity of devices and technologies makes it difficult to obtain exact figures. In 2020, the electricity needs of the ICT sector were estimated to represent up to 7% of the global electricity usage [121], and most studies agree that although this figure is high – due to the digitalisation of all aspects of society – it remains manageable thanks to technological improvements over the past decade that have partially offset increased usage [122]. However, this share is still growing rapidly – it was around 1% in 2018 [122] – which points to unsustainable energy needs in the future when hardware improvements will not be able to match the increasing demand. Moreover, it can be feared that this controlled increase has given an overly optimistic image of the road ahead and the early 2020s seems to be a pivotal point for ICT technologies [121], [123].

Data centres are only responsible for a portion of the impact of ICTs – and scientific computations are only a fraction of this – yet, the magnitude of the figures means that their emissions cannot be ignored. A 2018 News Feature in *Nature* by Nicola Jones wondered "How to stop data centres from gobbling up the world's electricity" [122]. In this widely cited piece, Jones looked at the global electricity usage of data centres and questioned the sustainability of the current trends. In 2018, globally, data centres were estimated to use between 200 TWh [122] and 400 TWh [124], and the expected usage for 2030 is as high as 974 TWh [121]. From these 2018 figures, I estimated a lower bound of the corresponding

GHG emissions to be around 100 $MtCO_2e$ per year, similar to the impact of all intra-European passenger flights in the same year [125]. I obtained this estimate using two methods that agreed on the order of magnitude. First, the 200 TWh estimate and the 2019 world average carbon intensity of 475 gCO2e/kWh [126] resulted in a footprint of 95 x 10^6 tCO_2e. To corroborate this, I used the estimated contribution of data centres to global emissions, 0.3% [122], and the estimation for global emissions, 36 x 10^9 tCO_2e, which yielded a total footprint of 108 x 10^6 tCO_2e.

Although outside the scope of this work, cryptocurrencies are often mentioned when discussing the carbon footprint of data centres. Rightfully so, as cryptocurrencies and their so-called *mining farms* have also seen their environmental impact increase exponentially in recent years, with several reports casting doubts on their sustainability [127]–[129]. A 2018 study estimated the yearly energy usage of Bitcoin to be 46 TWh, resulting in 22 Mt of CO2 released into the atmosphere [128]. In March 2022, Bitcoin's annual usage was estimated to be 138 TWh[2], which, if Bitcoin was a country, would rank its energy usage in the 27th highest position in the world, ahead of countries such as Norway, the United Arab Emirates and Argentina [130]. Even though crypto-mining relies on dedicated hardware (application-specific integrated circuits, ASIC) instead of usual processors and therefore does not compete directly with scientific computing, the magnitude of its carbon footprint needs to be addressed urgently.

The impact of (scientific) computing

Whereas many analyses of the environmental impact of ICT have been published, it is not the case for the impact of High-Performance Computing (HPC). One of the first studies on the topic was the 2019 work of Emma Strubell et al. investigating the carbon footprint of training popular models for NLP [131], which was largely relayed in the media.

> *"Training a single AI model can emit as much carbon as five cars in their lifetimes. Deep learning has a terrible carbon footprint."*
> *- MIT Technology Review [132] -*

[2] Crucially, the annual energy consumption of Bitcoin could power all the tea kettles in the UK for 30 years or the University of Cambridge for close to 1,000 years [130].

Strubell et al. indeed reported that in some extreme cases, when looking at larger models and including hyperparameters tuning, over 280,000 $kgCO_2e$ was emitted. Perhaps more importantly, they showed that most popular deep learning models have non-negligible carbon footprints that warrant further investigations. The method used for these estimates required monitoring the power usage of different hardware components, such as Central and Graphics Processing Units (CPUs and GPUs). Although a landmark study, the estimation method was impractical for most scientists with computers based in the institution's data centre. Later that year, Lacoste et al. published a more user-friendly online calculator [133]. Despite being called the *Machine Learning Emissions Calculator*, it was designed exclusively for deep learning using GPUs and cloud computing, therefore unsuitable for the vast majority of computational research which uses CPUs on local servers. The option of a Python library such as the *experiment-impact-tracker* [134] or *Code Carbon* [135] avoids having to monitor the hardware manually but restricts the audience to Python users, notwithstanding the compatibility issues that arise when adding new libraries to an analysis pipeline.

Environmental impact is part of a broader sustainability challenge for AI. To tend towards what Schwartz et al. called *Green AI* [136], the field will have to address the issue of accessibility as models' sizes and costs have increased exponentially over the past years [137]. Accessibility is particularly critical as marginalised communities tend to be penalised the most by these increasing costs (both environmental and financial ones). These communities are simultaneously least likely to benefit from these advances and most likely to suffer from climate change, while often not being able to participate in the research and development of these tools [138].

There is a broad consensus in the green computing community that the carbon footprint of AI is a growing source of concern. However, the research teams from Google notably stand out and claim that this impact is largely exaggerated. In 2021, as a direct reaction to the Bender et al. article "*On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?*" [138], they defended the carbon footprint of their language models [139] and now claim that "The carbon footprint of machine learning training will plateau, then shrink" [140]. Yann Le Cun, now Chief AI scientist at Meta (formerly Facebook), unsurprisingly

agrees[3]. There are indeed some positive developments in the field: large language models can be trained once and made available to thousands of researchers who otherwise would have to duplicate computations, and dedicated AI processing cores become more and more energy-efficient. However, Google came to this conclusion by only looking at the largest models trained and optimised for Google's processors and with full control over the data centres. They fail to consider the vast majority of modern deep learning research, the attempts at replicating or outperforming these tools, smaller architectures which run thousands of times each day etc. Besides, considering Google's incentive to promote its cloud-based deep learning platform, it is difficult to fully trust this analysis.

So far, I have only discussed deep learning, which has concentrated almost all the investigations into the topic despite representing only a small part of computational research. Notably, however, astronomers have investigated the environmental impact of their work quite extensively since 2019. Australian scientists found that, on average, each astronomer emits more than 37 tCO_2e per year, including 22 tCO_2e just for supercomputers and ~5 tCO_2e to operate observatories [141]. German astronomers broke down the emissions of the Max Planck Institute to promote greener practices [142] and found emissions per researcher for electricity usage (mostly computing) to be five times less than Australian astronomers, demonstrating the impact of electricity generation methods (Australia relies more on coal than Germany, more on this in Chapter 4). However, the total impact remains high, 18.1 tCO_2e per researcher, 2.6 times the 2030 German pledge of 6.8 tCO2e per capita per year [142]. These figures are in line with the survey of Dutch universities that found an average carbon footprint of 4.7 tCO_2e/year per member of staff [143]. In an unusual but very positive move, international researchers estimated the carbon footprint of the *Giant Array for Neutrino Detection* (GRAND), a worldwide network of 200,000 antennas. They estimated the carbon footprint of the full-scale project to be 13,407 tCO_2e/year, 45% of which being for digital technologies only (simulations, data analysis, data storage and transfer, communication) [144]. To further reduce emissions, Zwart released coding best practices for astronomy and discussed the pros and cons of different programming languages [145].

[3] https://twitter.com/ylecun/status/1496705645172727808

Unfortunately, this is where the literature ends. Outside of the fields of deep learning and astronomy, very little work has been conducted on the carbon footprint of computational science, and the calculators available are unsuitable for the vast majority of analyses. Besides, while efforts have been undertaken to reduce energy needs, the focus has been on hardware efficiency and cloud technologies [146]–[148] rather than controlling computing use. The Green Algorithms project presented in **Chapter 4** addresses this.

Finally, it is worth noting that other than computations, different environmental impacts of science have been investigated [149]. Conference travel has been particularly scrutinised, especially since the COVID pandemic; a study found that, ironically, climate-change researchers flew more often than other scientists [150], while a survey of the University of Montreal's staff estimated that professors had twice the carbon footprint of students [151]. As a case study, the 2019 Fall Meeting of the American Geophysical Union in California was shown to be responsible for the emission of 80,000 tCO_2e, the average weekly emissions of the city of Edinburgh [152]. As discussed above, Australian and German astronomers investigated the different impacts of their work, including flights (respectively ~6 tCO_2e and 8.5 tCO_2e per person and per year). Nathans and Sterling also investigated the carbon footprint of life science conferences and estimated that the 2014 annual meeting of the Society for Neuroscience emitted 20,000 tCO2e, equivalent to the annual footprint of 1,000 medium-sized laboratories [153]. A range of initiatives has tried to tackle the environmental impact of life science laboratories, such as the LEAF initiative driven by Martin Farley at UCL [154] and the Green Lab grassroots movements in the Netherlands [155].

References

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani, 'Introduction', in *An Introduction to Statistical Learning*, New York, NY: Springer US, 2021, pp. 1–14. doi: 10.1007/978-1-0716-1418-1_1.

[2] J. McCarthy, M. L. Minsky, N. Rochester, I. B. M. Corporation, and C. E. Shannon, 'A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence', p. 13, 1955.

[3] A. L. Samuel, 'Some studies in machine learning using the game of checkers', *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, juillet 1959, doi: 10.1147/rd.33.0210.

[4] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[6] R. S. Sutton and A. G. Barto, 'Introduction to reinforcement learning', 1998.

[7] É. Biernat and M. Lutz, *Data science: fondamentaux et études de cas machine learning avec Python et R*. Paris: Eyrolles, 2015.

[8] D. E. Hilt and D. W. Seegrist, *Ridge, a computer program for calculating ridge regression estimates*. Department of Agriculture, Forest Service, Northeastern Forest Experiment Station, 1977.

[9] R. Tibshirani, 'Regression Shrinkage and Selection via the Lasso', *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[10] H. Zou and T. Hastie, 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005, doi: 10.1111/j.1467-9868.2005.00503.x.

[11] M. E. Maron, 'Automatic Indexing: An Experimental Inquiry', *J. ACM*, vol. 8, no. 3, pp. 404–417, Jul. 1961, doi: 10.1145/321075.321084.

[12] P. Domingos and M. Pazzani, 'On the Optimality of the Simple Bayesian Classifier under Zero-One Loss', *Machine Learning*, vol. 29, no. 2, pp. 103–130, Nov. 1997, doi: 10.1023/A:1007413511361.

[13] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984. [Online]. Available: https://books.google.fr/books?id=JwQx-WOmSyQC

[14] S. L. Salzberg, 'C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993', *Mach Learn*, vol. 16, no. 3, pp. 235–240, Sep. 1994, doi: 10.1007/BF00993309.

[15] G. James, D. Witten, T. Hastie, and R. Tibshirani, 'Tree-Based Methods', in *An Introduction to Statistical Learning*, New York, NY: Springer US, 2021, pp. 327–365. doi: 10.1007/978-1-0716-1418-1_8.

[16] L. Breiman, 'Bagging predictors', *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.

[17] L. Breiman, 'Random Forests', *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[18] R. E. Schapire, 'The strength of weak learnability', *Mach Learn*, vol. 5, no. 2, pp. 197–227, Jun. 1990, doi: 10.1007/BF00116037.

[19] Y. Freund and R. E. Schapire, 'A Short Introduction to Boosting', *Journal of Japanese Society for Artificial Intelligence*, p. 14, 1999.

[20] J. H. Friedman, 'Greedy function approximation: A gradient boosting machine.', *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, doi: 10.1214/aos/1013203451.

[21] T. Chen and C. Guestrin, 'XGBoost: A Scalable Tree Boosting System', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, San Francisco, California, USA, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[22] ‘Machine Learning Wins the Higgs Challenge | ATLAS Experiment at CERN’. https://atlas.cern/updates/news/machine-learning-wins-higgs-challenge (accessed Mar. 24, 2022).

[23] ‘xgboost/demo at master · dmlc/xgboost’, *GitHub*. https://github.com/dmlc/xgboost (accessed Mar. 24, 2022).

[24] C. Cortes and V. Vapnik, ‘Support-vector networks’, *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[25] G. James, D. Witten, T. Hastie, and R. Tibshirani, ‘Support Vector Machines’, in *An Introduction to Statistical Learning: with Applications in R*, G. James, D. Witten, T. Hastie, and R. Tibshirani, Eds. New York, NY: Springer US, 2021, pp. 367–402. doi: 10.1007/978-1-0716-1418-1_9.

[26] F. Rosenblatt, ‘The perceptron: A probabilistic model for information storage and organization in the brain’, *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, doi: 10.1037/h0042519.

[27] Y. LeCun *et al.*, ‘Handwritten Digit Recognition with a Back-Propagation Network’, in *Advances in Neural Information Processing Systems*, 1990, vol. 2. Accessed: Feb. 04, 2022. [Online]. Available: https://papers.nips.cc/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html

[28] G. James, D. Witten, T. Hastie, and R. Tibshirani, ‘Deep Learning’, in *An Introduction to Statistical Learning*, New York, NY: Springer US, 2021, pp. 403–460. doi: 10.1007/978-1-0716-1418-1_10.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, ‘Learning representations by back-propagating errors’, *Nature*, vol. 323, no. 6088, Art. no. 6088, Oct. 1986, doi: 10.1038/323533a0.

[30] S. Hochreiter and J. Schmidhuber, ‘Long Short-Term Memory’, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[31] K. Cho *et al.*, ‘Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation’, *arXiv:1406.1078 [cs, stat]*, Sep. 2014, Accessed: Nov. 29, 2021. [Online]. Available: http://arxiv.org/abs/1406.1078

[32] *Handbook of Systems Biology*. Elsevier, 2013. doi: 10.1016/C2010-0-67190-2.

[33] S. Nurk *et al.*, ‘The complete sequence of a human genome’. bioRxiv, p. 2021.05.26.445798, May 27, 2021. doi: 10.1101/2021.05.26.445798.

[34] G. W. Beadle and E. L. Tatum, ‘Genetic Control of Biochemical Reactions in Neurospora*’, *Proceedings of the National Academy of Sciences*, vol. 27, no. 11, pp. 499–506, Nov. 1941, doi: 10.1073/pnas.27.11.499.

[35] ‘George W. Beadle’s One Gene-One Enzyme Hypothesis | The Embryo Project Encyclopedia’. https://embryo.asu.edu/pages/george-w-beadles-one-gene-one-enzyme-hypothesis (accessed Mar. 26, 2022).

[36] V. M. Ingram, ‘Gene Mutations in Human Hæmoglobin: the Chemical Difference Between Normal and Sickle Cell Hæmoglobin’, *Nature*, vol. 180, no. 4581, Art. no. 4581, Aug. 1957, doi: 10.1038/180326a0.

[37] ‘What are proteins and what do they do?: MedlinePlus Genetics’. https://medlineplus.gov/genetics/understanding/howgeneswork/protein/ (accessed Mar. 26, 2022).

[38] M. Y. Hein, K. Sharma, J. Cox, and M. Mann, ‘Proteomic Analysis of Cellular Systems’, in *Handbook of Systems Biology*, Elsevier, 2013, pp. 3–25. doi: 10.1016/B978-0-12-385944-0.00001-0.

[39] *A dictionary of biology*, 8th edition. New York, NY: Oxford University Press, 2019.

[40] B. A. Shoemaker and A. R. Panchenko, ‘Deciphering Protein–Protein Interactions. Part I. Experimental Techniques and Databases’, *PLOS Computational Biology*, vol. 3, no. 3, p. e42, Mar. 2007, doi: 10.1371/journal.pcbi.0030042.

[41] J. Snider, M. Kotlyar, P. Saraon, Z. Yao, I. Jurisica, and I. Stagljar, ‘Fundamentals of protein interaction network mapping’, *Molecular Systems Biology*, vol. 11, no. 12, pp. 848–848, Dec. 2015, doi: 10.15252/msb.20156351.

[42] R. B. Russell *et al.*, 'A structural perspective on protein–protein interactions', *Current Opinion in Structural Biology*, vol. 14, no. 3, pp. 313–324, Jun. 2004, doi: 10.1016/j.sbi.2004.04.006.

[43] J. Lippincott-Schwartz and G. H. Patterson, 'Development and use of fluorescent protein markers in living cells', *Science*, vol. 300, no. 5616, pp. 87–91, Apr. 2003, doi: 10.1126/science.1082520.

[44] J. Piehler, 'New methodologies for measuring protein interactions in vivo and in vitro', *Curr Opin Struct Biol*, vol. 15, no. 1, pp. 4–14, Feb. 2005, doi: 10.1016/j.sbi.2005.01.008.

[45] Y. Yang, H. Wang, and D. A. Erie, 'Quantitative characterization of biomolecular assemblies and interactions using atomic force microscopy', *Methods*, vol. 29, no. 2, pp. 175–187, Feb. 2003, doi: 10.1016/s1046-2023(02)00308-0.

[46] G. P. Smith, 'Filamentous Fusion Phage: Novel Expression Vectors That Display Cloned Antigens on the Virion Surface', *Science*, vol. 228, no. 4705, pp. 1315–1317, Jun. 1985, doi: 10.1126/science.4001944.

[47] S. Fields and O. Song, 'A novel genetic system to detect protein-protein interactions', p. 2, 1989.

[48] M. Dreze *et al.*, 'High-Quality Binary Interactome Mapping', in *Methods in Enzymology*, vol. 470, Elsevier, 2010, pp. 281–315. doi: 10.1016/S0076-6879(10)70012-4.

[49] J.-S. Lin and E.-M. Lai, 'Protein–Protein Interactions: Yeast Two-Hybrid System', in *Bacterial Protein Secretion Systems*, vol. 1615, L. Journet and E. Cascales, Eds. New York, NY: Springer New York, 2017, pp. 177–187. doi: 10.1007/978-1-4939-7033-9_14.

[50] A.-R. Carvunis, F. P. Roth, M. A. Calderwood, M. E. Cusick, G. Superti-Furga, and M. Vidal, 'Interactome Networks', in *Handbook of Systems Biology*, Elsevier, 2013, pp. 45–63. doi: 10.1016/B978-0-12-385944-0.00003-4.

[51] E. J. Deeds, O. Ashenberg, and E. I. Shakhnovich, 'A simple physical model for scaling in protein-protein interaction networks', *PNAS*, vol. 103, no. 2, pp. 311–316, Jan. 2006, doi: 10.1073/pnas.0509715102.

[52] N. Pržulj, Ed., *Analyzing Network Data in Biology and Medicine: An Interdisciplinary Textbook for Biological, Medical and Computational Scientists*. Cambridge: Cambridge University Press, 2019. doi: 10.1017/9781108377706.

[53] I. Stagljar, C. Korostensky, N. Johnsson, and S. te Heesen, 'A genetic system based on split-ubiquitin for the analysis of interactions between membrane proteins in vivo', *Proceedings of the National Academy of Sciences*, vol. 95, no. 9, pp. 5187–5192, Apr. 1998, doi: 10.1073/pnas.95.9.5187.

[54] J. Snider, S. Kittanakom, D. Damjanovic, J. Curak, V. Wong, and I. Stagljar, 'Detecting interactions with membrane proteins using a membrane two-hybrid assay in yeast', *Nat Protoc*, vol. 5, no. 7, pp. 1281–1293, Jul. 2010, doi: 10.1038/nprot.2010.83.

[55] R. Aebersold and M. Mann, 'Mass spectrometry-based proteomics', *Nature*, vol. 422, p. 10, 2003.

[56] G. Rigaut, A. Shevchenko, B. Rutz, M. Wilm, M. Mann, and B. Séraphin, 'A generic protein purification method for protein complex characterization and proteome exploration', *Nat Biotechnol*, vol. 17, no. 10, Art. no. 10, Oct. 1999, doi: 10.1038/13732.

[57] O. Puig *et al.*, 'The Tandem Affinity Purification (TAP) Method: A General Procedure of Protein Complex Purification', *Methods*, vol. 24, no. 3, pp. 218–229, Jul. 2001, doi: 10.1006/meth.2001.1183.

[58] A.-C. Gavin *et al.*, 'Functional organization of the yeast proteome by systematic analysis of protein complexes', *Nature*, vol. 415, no. 6868, pp. 141–147, Jan. 2002, doi: 10.1038/415141a.

[59] Y. Ho *et al.*, 'Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry', *Nature*, vol. 415, no. 6868, pp. 180–183, Jan. 2002, doi: 10.1038/415180a.

[60] P. Uetz *et al.*, 'A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae', *Nature*, vol. 403, no. 6770, pp. 623–627, Feb. 2000, doi: 10.1038/35001009.

[61] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, 'A comprehensive two-hybrid analysis to explore the yeast protein interactome', *Proceedings of the National Academy of Sciences*, vol. 98, no. 8, pp. 4569–4574, Apr. 2001, doi: 10.1073/pnas.061034498.

[62] R. Jansen, D. Greenbaum, and M. Gerstein, 'Relating whole-genome expression data with protein-protein interactions', *Genome Res*, vol. 12, no. 1, pp. 37–46, Jan. 2002, doi: 10.1101/gr.205602.

[63] H. B. Fraser, A. E. Hirsh, D. P. Wall, and M. B. Eisen, 'Coevolution of gene expression among interacting proteins', *PNAS*, vol. 101, no. 24, pp. 9033–9038, Jun. 2004, doi: 10.1073/pnas.0402591101.

[64] S. L. Ooi *et al.*, 'Global synthetic-lethality analysis and yeast functional profiling', *Trends Genet*, vol. 22, no. 1, pp. 56–63, Jan. 2006, doi: 10.1016/j.tig.2005.11.003.

[65] I. Xenarios, Ł. Salwínski, X. J. Duan, P. Higney, S.-M. Kim, and D. Eisenberg, 'DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions', *Nucleic Acids Res*, vol. 30, no. 1, pp. 303–305, Jan. 2002.

[66] L. Licata *et al.*, 'MINT, the molecular interaction database: 2012 update', *Nucleic Acids Research*, vol. 40, no. D1, pp. D857–D861, Jan. 2012, doi: 10.1093/nar/gkr930.

[67] R. Oughtred *et al.*, 'The BioGRID interaction database: 2019 update', *Nucleic Acids Res*, vol. 47, no. D1, pp. D529–D541, Jan. 2019, doi: 10.1093/nar/gky1079.

[68] D. Szklarczyk *et al.*, 'The STRING database in 2021: customizable protein-protein networks, and functional characterization of user-uploaded gene/measurement sets', *Nucleic Acids Res*, vol. 49, no. D1, pp. D605–D612, Jan. 2021, doi: 10.1093/nar/gkaa1074.

[69] T. S. Keshava Prasad *et al.*, 'Human Protein Reference Database--2009 update', *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D767-772, Jan. 2009, doi: 10.1093/nar/gkn892.

[70] S. Orchard *et al.*, 'Protein interaction data curation: the International Molecular Exchange (IMEx) consortium', *Nature Methods*, vol. 9, no. 4, pp. 345–350, Apr. 2012, doi: 10.1038/nmeth.1931.

[71] S. Orchard *et al.*, 'The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases.', *Nucleic Acids Res*, vol. 42, no. Database issue, pp. D358-63, Jan. 2014, doi: 10.1093/nar/gkt1115.

[72] K. H. Vousden and D. P. Lane, 'p53 in health and disease', *Nat Rev Mol Cell Biol*, vol. 8, no. 4, pp. 275–283, Apr. 2007, doi: 10.1038/nrm2147.

[73] C. von Mering *et al.*, 'Comparative assessment of large-scale data sets of protein–protein interactions', *Nature*, vol. 417, no. 6887, pp. 399–403, May 2002, doi: 10.1038/nature750.

[74] K. Luck *et al.*, 'A reference map of the human binary protein interactome', *Nature*, vol. 580, no. 7803, pp. 402–408, Apr. 2020, doi: 10.1038/s41586-020-2188-x.

[75] E. L. Huttlin *et al.*, 'Dual proteome-scale networks reveal cell-specific remodeling of the human interactome', *Cell*, vol. 184, no. 11, pp. 3022-3040.e28, May 2021, doi: 10.1016/j.cell.2021.04.011.

[76] G. D. Bader and C. W. V. Hogue, 'Analyzing yeast protein–protein interaction data obtained from different sources', *Nat Biotechnol*, vol. 20, no. 10, pp. 991–997, Oct. 2002, doi: 10.1038/nbt1002-991.

[77] L. Hu, X. Wang, Y.-A. Huang, P. Hu, and Z.-H. You, 'A survey on computational models for predicting protein–protein interactions', *Briefings in Bioinformatics*, vol. 22, no. 5, Sep. 2021, doi: 10.1093/bib/bbab036.

[78] R. Jansen, 'A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data', *Science*, vol. 302, no. 5644, pp. 449–453, Oct. 2003, doi: 10.1126/science.1087361.

[79] S. Jain and G. D. Bader, 'An improved method for scoring protein-protein interactions using semantic similarity within the gene ontology', *BMC Bioinformatics*, vol. 11, no. 1, p. 562, Nov. 2010, doi: 10.1186/1471-2105-11-562.

[80] S. Bandyopadhyay and K. Mallick, 'A New Feature Vector Based on Gene Ontology Terms for Protein-Protein Interaction Prediction', *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 4, pp. 762–770, Jul. 2017, doi: 10.1109/TCBB.2016.2555304.

[81] I. M. Armean, K. S. Lilley, M. W. B. Trotter, N. C. V. Pilkington, and S. B. Holden, 'Co-complex protein membership evaluation using Maximum Entropy on GO ontology and InterPro annotation', *Bioinformatics*, vol. 34, no. 11, pp. 1884–1892, Jun. 2018, doi: 10.1093/bioinformatics/btx803.

[82] X. Zhong and J. C. Rajapakse, 'Graph embeddings on gene ontology annotations for protein–protein interaction prediction', *BMC Bioinformatics*, vol. 21, no. Suppl 16, p. 560, Dec. 2020, doi: 10.1186/s12859-020-03816-8.

[83] F. Pazos and A. Valencia, 'Similarity of phylogenetic trees as indicator of protein–protein interaction', *Protein Engineering, Design and Selection*, vol. 14, no. 9, pp. 609–614, Sep. 2001, doi: 10.1093/protein/14.9.609.

[84] M. Weigt, R. A. White, H. Szurmant, J. A. Hoch, and T. Hwa, 'Identification of direct residue contacts in protein–protein interaction by message passing', *Proc. Natl. Acad. Sci. U.S.A.*, vol. 106, no. 1, pp. 67–72, Jan. 2009, doi: 10.1073/pnas.0805923106.

[85] A.-F. Bitbol, 'Inferring interaction partners from protein sequences using mutual information', *PLoS Comput Biol*, vol. 14, no. 11, p. e1006401, Nov. 2018, doi: 10.1371/journal.pcbi.1006401.

[86] B. S. Srinivasan, A. F. Novak, J. A. Flannick, S. Batzoglou, and H. H. McAdams, 'Integrated Protein Interaction Networks for 11 Microbes', in *Research in Computational Molecular Biology*, 2006, pp. 1–14.

[87] G. Marmier, M. Weigt, and A.-F. Bitbol, 'Phylogenetic correlations can suffice to infer protein partners from sequences', *PLoS Comput Biol*, vol. 15, no. 10, p. e1007179, Oct. 2019, doi: 10.1371/journal.pcbi.1007179.

[88] I. A. Kovács *et al.*, 'Network-based prediction of protein interactions', *Nat Commun*, vol. 10, no. 1, p. 1240, Dec. 2019, doi: 10.1038/s41467-019-09177-y.

[89] X.-W. Wang *et al.*, 'Assessment of community efforts to advance computational prediction of protein-protein interactions', Systems Biology, preprint, Sep. 2021. doi: 10.1101/2021.09.22.461292.

[90] S. Pitre *et al.*, 'PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs', *BMC Bioinformatics*, vol. 7, no. 1, p. 365, Jul. 2006, doi: 10.1186/1471-2105-7-365.

[91] Y. Murakami and K. Mizuguchi, 'Homology-based prediction of interactions between proteins using Averaged One-Dependence Estimators', *BMC Bioinformatics*, vol. 15, no. 1, p. 213, Jun. 2014, doi: 10.1186/1471-2105-15-213.

[92] G. I. Webb, J. R. Boughton, and Z. Wang, 'Not So Naive Bayes: Aggregating One-Dependence Estimators', *Mach Learn*, vol. 58, no. 1, pp. 5–24, Jan. 2005, doi: 10.1007/s10994-005-4258-6.

[93] L. Becchetti, A. Fazzone, and L. Martini, 'Network and Sequence-Based Prediction of Protein-Protein Interactions', *arXiv:2107.03694 [cs, q-bio]*, Jul. 2021, Accessed: Aug. 16, 2021. [Online]. Available: http://arxiv.org/abs/2107.03694

[94] Y. Ofran and B. Rost, 'Analysing Six Types of Protein–Protein Interfaces', *Journal of Molecular Biology*, vol. 325, no. 2, pp. 377–387, Jan. 2003, doi: 10.1016/S0022-2836(02)01223-8.

[95] E. Sprinzak and H. Margalit, 'Correlated sequence-signatures as markers of protein-protein interaction1 1Edited by G. von Heijne', *Journal of Molecular Biology*, vol. 311, no. 4, pp. 681–692, Aug. 2001, doi: 10.1006/jmbi.2001.4920.

[96] B. Ma, T. Elkayam, H. Wolfson, and R. Nussinov, 'Protein-protein interactions: Structurally conserved residues distinguish between binding sites and exposed protein surfaces', *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5772–5777, May 2003, doi: 10.1073/pnas.1030237100.

[97] W. S. Valdar and J. M. Thornton, 'Protein-protein interfaces: analysis of amino acid conservation in homodimers', *Proteins*, vol. 42, no. 1, pp. 108–124, Jan. 2001.

[98] A. Ben-Hur and W. S. Noble, 'Kernel methods for predicting protein-protein interactions', *Bioinformatics*, vol. 21, no. Suppl 1, pp. i38–i46, Jun. 2005, doi: 10.1093/bioinformatics/bti1016.

[99] X.-W. Chen and M. Liu, 'Prediction of protein–protein interactions using random decision forest framework', *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, Dec. 2005, doi: 10.1093/bioinformatics/bti721.

[100] M. S. Scott and G. J. Barton, 'Probabilistic prediction and ranking of human protein-protein interactions', *BMC Bioinformatics*, vol. 8, no. 1, p. 239, Jul. 2007, doi: 10.1186/1471-2105-8-239.

[101] M. Kotlyar *et al.*, '*In silico* prediction of physical protein interactions and characterization of interactome orphans', *Nature Methods*, vol. 12, no. 1, pp. 79–84, Jan. 2015, doi: 10.1038/nmeth.3178.

[102] T. Sun, B. Zhou, L. Lai, and J. Pei, 'Sequence-based prediction of protein protein interaction using a deep-learning algorithm', *BMC Bioinformatics*, vol. 18, no. 1, p. 277, May 2017, doi: 10.1186/s12859-017-1700-2.

[103] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, 'Extreme Learning Machine for Regression and Multiclass Classification', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, Apr. 2012, doi: 10.1109/TSMCB.2011.2168604.

[104] Z.-H. You, M. Zhou, X. Luo, and S. Li, 'Highly Efficient Framework for Predicting Interactions Between Proteins', *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 731–743, Mar. 2017, doi: 10.1109/TCYB.2016.2524994.

[105] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, 'Signature Verification using a "Siamese" Time Delay Neural Network', in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan-Kaufmann, 1994, pp. 737–744. Accessed: Oct. 21, 2019. [Online]. Available: http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf

[106] S. Chopra, R. Hadsell, and Y. LeCun, 'Learning a Similarity Metric Discriminatively, with Application to Face Verification', in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, vol. 1, pp. 539–546. doi: 10.1109/CVPR.2005.202.

[107] S. Hashemifar, B. Neyshabur, A. A. Khan, and J. Xu, 'Predicting protein–protein interactions through sequence-based deep learning', *Bioinformatics*, vol. 34, no. 17, pp. i802–i810, Sep. 2018, doi: 10.1093/bioinformatics/bty573.

[108] H. Li, X.-J. Gong, H. Yu, and C. Zhou, 'Deep Neural Network Based Predictions of Protein Interactions Using Primary Sequences', *Molecules*, vol. 23, no. 8, p. 1923, Aug. 2018, doi: 10.3390/molecules23081923.

[109] Y. Guo and X. Chen, 'A deep learning framework for improving protein interaction prediction using sequence properties', Bioinformatics, preprint, Nov. 2019. doi: 10.1101/843755.

[110] M. Chen *et al.*, 'Multifaceted protein–protein interaction prediction based on Siamese residual RCNN', *Bioinformatics*, vol. 35, no. 14, pp. i305–i314, Jul. 2019, doi: 10.1093/bioinformatics/btz328.

[111] A. Vaswani *et al.*, 'Attention Is All You Need', *arXiv:1706.03762 [cs]*, Dec. 2017, Accessed: Feb. 17, 2022. [Online]. Available: http://arxiv.org/abs/1706.03762

[112] F. Li, F. Zhu, X. Ling, and Q. Liu, 'Protein Interaction Network Reconstruction Through Ensemble Deep Learning With Attention Mechanism', *Front Bioeng Biotechnol*, vol. 8, p. 390, May 2020, doi: 10.3389/fbioe.2020.00390.

[113] M. Varadi *et al.*, 'AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models', *Nucleic Acids Research*, vol. 50, no. D1, pp. D439–D444, Jan. 2022, doi: 10.1093/nar/gkab1061.

[114] Q. C. Zhang *et al.*, 'Structure-based prediction of protein–protein interactions on a genome-wide scale', *Nature*, vol. 490, no. 7421, pp. 556–560, Oct. 2012, doi: 10.1038/nature11503.

[115] C. Mirabello and B. Wallner, 'InterPred: A pipeline to identify and model protein–protein interactions', *Proteins: Structure, Function, and Bioinformatics*, vol. 85, no. 6, pp. 1159–1170, 2017, doi: 10.1002/prot.25280.

[116] C. Zhao, Y. Zang, W. Quan, X. Hu, and A. Sacan, 'HIV1-human protein-protein interaction prediction based on interface architecture similarity', in *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Nov. 2017, pp. 97–100. doi: 10.1109/BIBM.2017.8217632.

[117] H. Zhou, J. F. Beltrán, and I. L. Brito, 'Host-microbiome protein-protein interactions capture disease-relevant pathways', *Genome Biol*, vol. 23, no. 1, p. 72, Mar. 2022, doi: 10.1186/s13059-022-02643-9.

[118] H. Lu *et al.*, 'Recent advances in the development of protein–protein interactions modulators: mechanisms and clinical trials', *Sig Transduct Target Ther*, vol. 5, no. 1, p. 213, Sep. 2020, doi: 10.1038/s41392-020-00315-3.

[119] M. D. McDowall, M. S. Scott, and G. J. Barton, 'PIPs: human protein–protein interaction prediction database', *Nucleic Acids Research*, vol. 37, no. suppl_1, pp. D651–D656, Jan. 2009, doi: 10.1093/nar/gkn870.

[120] B. Dunham and M. K. Ganapathiraju, 'Benchmark Evaluation of Protein–Protein Interaction Prediction Algorithms', *Molecules*, vol. 27, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/molecules27010041.

[121] A. S. G. Andrae, 'New perspectives on internet electricity use in 2030', *Engineering and Applied Science Letter*, vol. 3, p. 13, 2020, doi: 10.30538/psrp-easl2020.0038.

[122] N. Jones, 'How to stop data centres from gobbling up the world's electricity', *Nature*, vol. 561, no. 7722, pp. 163–166, Sep. 2018, doi: 10.1038/d41586-018-06610-y.

[123] J. Lorincz, A. Capone, and J. Wu, 'Greener, Energy-Efficient and Sustainable Networks: State-Of-The-Art and New Trends', *Sensors*, vol. 19, no. 22, Art. no. 22, Jan. 2019, doi: 10.3390/s19224864.

[124] R. Hintemann, 'Data centers 2018. Efficiency gains are not enough: Data center energy consumption continues to rise significantly - Cloud computing boosts growth', 2020, doi: 10.13140/RG.2.2.26033.40800.

[125] B. Graver, 'CO2 emissions from commercial aviation: 2013, 2018, and 2019', p. 36, 2013.

[126] 'Emissions – Global Energy & CO2 Status Report 2019 – Analysis', *IEA*. https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions (accessed Feb. 10, 2020).

[127] C. Mora *et al.*, 'Bitcoin emissions alone could push global warming above 2°C', *Nature Clim Change*, vol. 8, no. 11, Art. no. 11, Nov. 2018, doi: 10.1038/s41558-018-0321-8.

[128] C. Stoll, L. Klaaßen, and U. Gallersdörfer, 'The Carbon Footprint of Bitcoin', *Joule*, vol. 3, no. 7, pp. 1647–1661, Jul. 2019, doi: 10.1016/j.joule.2019.05.012.

[129] S. Jiang *et al.*, 'Policy assessments for the carbon emission flows and sustainability of Bitcoin blockchain operation in China', *Nat Commun*, vol. 12, no. 1, Art. no. 1, Apr. 2021, doi: 10.1038/s41467-021-22256-3.

[130] 'Cambridge Bitcoin Electricity Consumption Index (CBECI)'. https://cbeci.org/contact/ (accessed Jan. 13, 2022).

[131] E. Strubell, A. Ganesh, and A. McCallum, 'Energy and Policy Considerations for Deep Learning in NLP', *arXiv:1906.02243 [cs]*, Jun. 2019, Accessed: Jan. 01, 2020. [Online]. Available: http://arxiv.org/abs/1906.02243

[132] 'Training a single AI model can emit as much carbon as five cars in their lifetimes', *MIT Technology Review*. https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/ (accessed Mar. 28, 2022).

[133] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, 'Quantifying the Carbon Emissions of Machine Learning', *arXiv:1910.09700 [cs]*, Nov. 2019, Accessed: Jan. 07, 2020. [Online]. Available: http://arxiv.org/abs/1910.09700

[134] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, 'Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning', *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020.

[135] 'CodeCarbon.io'. https://codecarbon.io/ (accessed Mar. 25, 2022).

[136] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, 'Green AI', *arXiv:1907.10597 [cs, stat]*, Aug. 2019, Accessed: Jan. 07, 2020. [Online]. Available: http://arxiv.org/abs/1907.10597

[137] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[138] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, 'On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?', in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, New York, NY, USA, Mar. 2021, pp. 610–623. doi: 10.1145/3442188.3445922.

[139] D. Patterson *et al.*, 'Carbon Emissions and Large Neural Network Training', *arXiv:2104.10350 [cs]*, Apr. 2021, Accessed: Mar. 04, 2022. [Online]. Available: http://arxiv.org/abs/2104.10350

[140] D. Patterson *et al.*, 'The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink', preprint, Mar. 2022. doi: 10.36227/techrxiv.19139645.v3.

[141] A. R. H. Stevens, S. Bellstedt, P. J. Elahi, and M. T. Murphy, 'The imperative to reduce carbon emissions in astronomy', *Nature Astronomy*, vol. 4, no. 9, Art. no. 9, Sep. 2020, doi: 10.1038/s41550-020-1169-1.

[142] K. Jahnke *et al.*, 'An astronomical institute's perspective on meeting the challenges of the climate crisis', *Nature Astronomy*, vol. 4, no. 9, Art. no. 9, Sep. 2020, doi: 10.1038/s41550-020-1202-4.

[143] F. van der Tak *et al.*, 'The carbon footprint of astronomy research in the Netherlands', *Nat Astron*, vol. 5, no. 12, pp. 1195–1198, Dec. 2021, doi: 10.1038/s41550-021-01552-4.

[144] C. Aujoux, K. Kotera, and O. Blanchard, 'Estimating the carbon footprint of the GRAND project, a multi-decade astrophysics experiment', *Astroparticle Physics*, vol. 131, p. 102587, Sep. 2021, doi: 10.1016/j.astropartphys.2021.102587.

[145] S. Portegies Zwart, 'The ecological impact of high-performance computing in astrophysics', *Nature Astronomy*, vol. 4, no. 9, Art. no. 9, Sep. 2020, doi: 10.1038/s41550-020-1208-y.

[146] 'Towards Green ICT Strategies: Assessing Policies and Programmes on ICT and the Environment', OECD Digital Economy Papers 155, May 2009. doi: 10.1787/222431651031.

[147] S. Sarkar and S. Misra, 'Theoretical modelling of fog computing: a green computing paradigm to support IoT applications', *IET Networks*, vol. 5, no. 2, pp. 23–29, Mar. 2016, doi: 10.1049/iet-net.2015.0034.

[148] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, 'Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing', *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, Jan. 2016, doi: 10.1016/j.jnca.2015.05.016.

[149] 'The huge carbon footprint of large-scale computing', *Physics World*, Mar. 16, 2022. https://physicsworld.com/the-huge-carbon-footprint-of-large-scale-computing/ (accessed Mar. 25, 2022).

[150] L. Whitmarsh, S. Capstick, I. Moore, J. Köhler, and C. Le Quéré, 'Use of aviation by climate change researchers: Structural influences, personal attitudes, and information provision', *Global Environmental Change*, vol. 65, p. 102184, Nov. 2020, doi: 10.1016/j.gloenvcha.2020.102184.

[151] J. Arsenault, J. Talbot, L. Boustani, R. Gonzalès, and K. Manaugh, 'The environmental footprint of academic and student mobility in a large research-oriented university', *Environ. Res. Lett.*, vol. 14, no. 9, p. 095001, Aug. 2019, doi: 10.1088/1748-9326/ab33e6.

[152] M. Klöwer, D. Hopkins, M. Allen, and J. Higham, 'An analysis of ways to decarbonize conference travel after COVID-19', *Nature*, vol. 583, no. 7816, pp. 356–359, Jul. 2020, doi: 10.1038/d41586-020-02057-2.

[153] J. Nathans and P. Sterling, 'How scientists can reduce their carbon footprint', *eLife*, vol. 5, p. e15928, Mar. 2016, doi: 10.7554/eLife.15928.

[154] UCL, 'Take part in LEAF', *Sustainable UCL*, Jan. 23, 2019. https://www.ucl.ac.uk/sustainable/staff/labs/take-part-leaf (accessed Mar. 25, 2022).

[155] 'About Green Labs NL – Green Labs – NL'. https://www.greenlabs-nl.eu/about/ (accessed Mar. 25, 2022).

CHAPTER 2

B4PPI: A UNIFIED FRAMEWORK FOR *IN SILICO* PREDICTION OF PROTEIN-PROTEIN INTERACTIONS

Contents

Introduction

Protein-protein interactions (PPIs) are central to protein functions and inform a wide range of biomedical applications, from mechanistic studies [1], [2] to drug development [3], [4]. Better understanding these interactions is critical to successfully map biological pathways, but the diversity of PPIs and the scale of the network make this a difficult task. Experimental methods to map PPIs exist, but, despite progress in systematic mapping [5], [6], even high-throughput ones are not yet able to investigate all possible interaction.

As discussed in Chapter 1, computational methods can address the issue of scalability and measurement bias; given a pair of proteins and some characteristics of each one, machine learning models can learn to predict the likelihood of interaction. Numerous methods have been developed for this, using the full range of machine learning models, from early work on *S. cerevisiae* [7]–[10] to algorithms dedicated to human PPIs [11]–[15]. Yet, despite a wealth of tools, reported performance scores often cannot be compared or replicated due to proprietary data and inconsistent or flawed assessment methods [16]. The lack of a consistent way to assess PPI prediction algorithms has hindered the development of these tools and reduced their impact by making it difficult to reuse models for downstream analyses [17]. As a consequence, there are multiple issues for *in silico* PPIs: it is unclear what the state-of-the-art is, analyses are difficult to reconcile, the development of new models is inefficient, follow-up mechanisms studies are likely undermined, and ultimately, there are different versions of the underlying molecular networks that describe protein functions.

A unified framework for PPI inference would improve the development and reliable assessment of new models [18] and would facilitate the overdue widespread adoption of PPI predictions for downstream analysis. Replicable, trustworthy and generalisable high-performing models can capture more causal biology and enhance many aspects of biological research, such as experimental designs and drug development.

I designed the Benchmarking Pipeline for the Prediction of Protein-Protein Interactions (B4PPI), a robust and standardised approach to *in silico* PPI prediction that accounts for both biological and statistical pitfalls and leverages the strengths of large, open-source and

professionally curated databases. I made publicly available benchmarking standards for human PPIs to accelerate future discoveries and laid the foundations for similar datasets for yeast and other organisms. All this, including the pre-processing steps and relevant guidelines, is made available online[1] and can be easily downloaded alongside minimal examples. An example of a reporting sheet is in **Figure 1**. This work provides robust foundations for future developments in PPI predictions and enables deeper analyses to understand the learning mechanisms of prediction models (see **Chapter 3**).

Publication and contribution

The work presented in this chapter has been released initially as a preprint on *bioRxiv* [19].

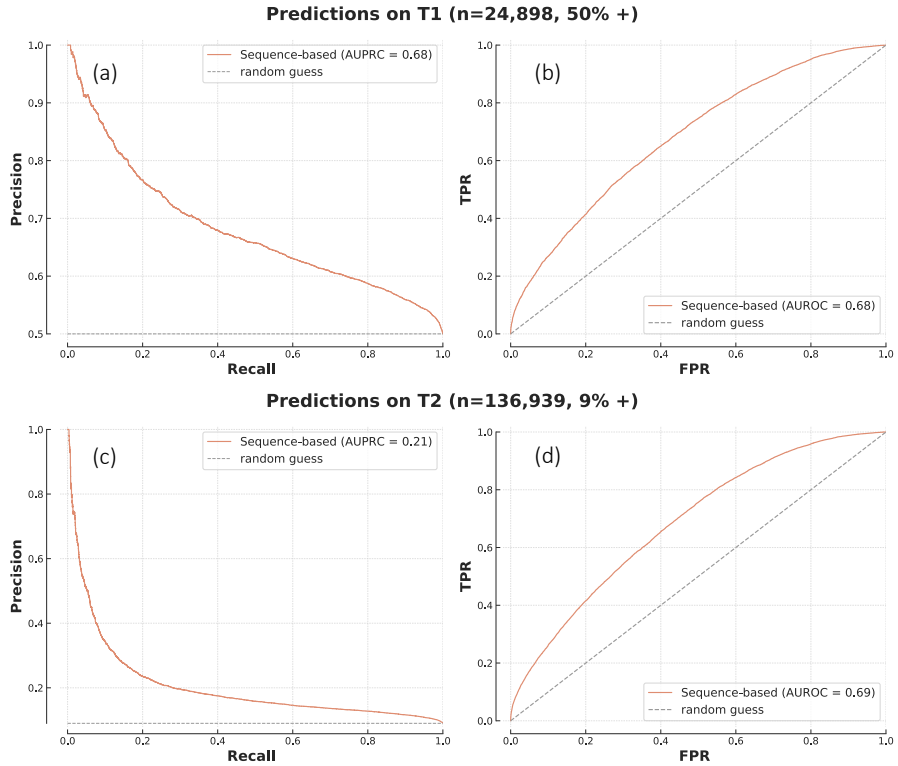
L. Lannelongue and M. Inouye,
'Construction of *in silico* protein-protein interaction networks across different topologies using machine learning'.
bioRxiv, p. 2022.02.07.479382, Feb. 09, 2022.
doi: 10.1101/2022.02.07.479382.

Contribution: I designed and conducted the analyses presented here, with feedback and suggestions from Prof. Michael Inouye and colleagues.

[1] https://github.com/Llannelongue/B4PPI

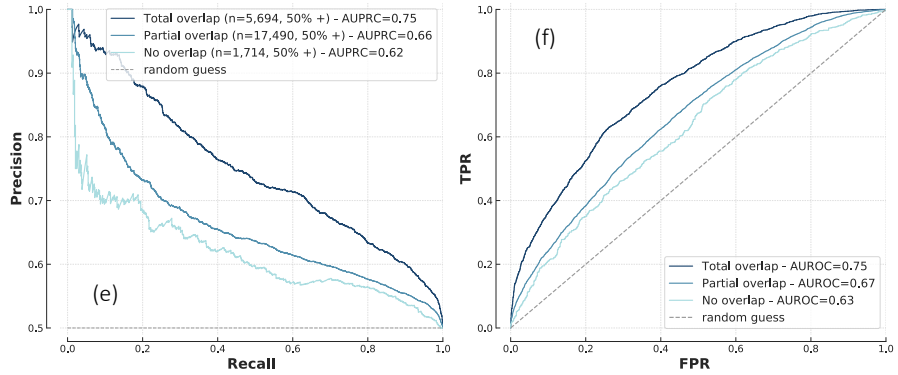
Reporting sheet B4PPI-Human: Sequence-based

PR and ROC curves on T1* and T2**



* First testing set, used to compare models on an independent set and investigate protein-level overlap.
** Second testing set, used to assess generalisation on an imbalanced dataset (10 times more negative examples than positive ones).

Impact of protein-level overlap



	Running time	Memory	Energy used	Carbon footprint (UK)
Training once	1h10	15 GB	0.75 kWh	189 gCO_2e
Training incl. hyperparameters tuning	>100h	>1.5 TB	> 75 kWh	> 18.9 $kgCO_2e$
Inference	1min46	6 GB	0.01 kWh	4 gCO_2e

Number of (trainable) parameters: 1.6m

Figure 1: Example of reporting sheet for B4PPI-Human.
Panels (a) and (b) show, respectively, the PR and ROC curves on the first testing set T1. (c) and (d) show the PR and ROC curves for the second testing set, T2. Panels (e) and (f) show the PR and ROC curves broken down by protein-level overlap.

Challenges and opportunities for a PPI benchmarking standard

The complexity of the underlying biological mechanisms of PPIs introduces pitfalls that need to be carefully considered when evaluating models. First, the way non-interacting proteins are selected for training is important. While some efforts, both early on and more recently, have used proteins known to be localised in different parts of the cell [7], [14], [20], [21], this has been shown to be unreliable and a source of significant bias that overestimates accuracy [22]. An alternative is to use a database of experimentally tested non-interacting proteins, but leading resources such as Negatome have only ~1,300 pairs (1,321 non-interacting pairs between 972 proteins for Negatome 2.0) and thus offer limited coverage [20], [23]. Considering the scarcity of PPIs, randomly sampling pairs of proteins has a very low risk of false negative and limits sampling bias (i.e. focusing on already well studied proteins) [22], [24]. Moreover, the associated imbalance between interacting and non-interacting proteins should be considered when training models on balanced datasets (i.e. with 50% of positive examples) [25]. Lastly, each observation is itself a pair of proteins. Even when ensuring that training and testing sets don't have pairs in common, there can be individual proteins present in both sets. This protein-level overlap, often overseen, has been shown to significantly affect the performance of an algorithm and should therefore be properly assessed [26]. Despite being documented in the literature, these pitfalls are still unevenly accounted for in published works. These issues, alongside inconsistencies in the choice of testing sets and performance metrics, explains why, despite the number of algorithms released in recent years, there are still no simple ways to compare a new approach to the state-of-the-art, or even know what the state-of-the-art is.

Although it is a complex problem, modern resources offer new opportunities for a reproducible PPI framework. Historically, PPI models were built on data generated by a specific experiment, which is easy to process but the experiment's measurement biases are directly passed onto the model. In recent years, aggregated databases have progressed immensely; for example, UniProt from EMBL-EBI went from 25 million entries ten years ago to almost 350m at the start of 2020 [27]. Large and professionally curated databases also leverage experts' knowledge in the curation process to ensure high-quality data and

limit measurement bias, as each annotation is based on several experiments. UniProt's protein IDs also provide standardised and consistent protein identifiers alongside matching tables to ensure maximum compatibilities with other databases.

The essential aspects of training and assessment that should be systematically accounted for are (1) the quality of the positive examples (i.e. the interacting proteins), (2) how non-interacting proteins are selected for the gold standard, (3) a suitable split between training and testing sets, in particular regarding individual proteins, and (4) the metrics to evaluate and compare models. Here I seek to address these four aspects of benchmarking.

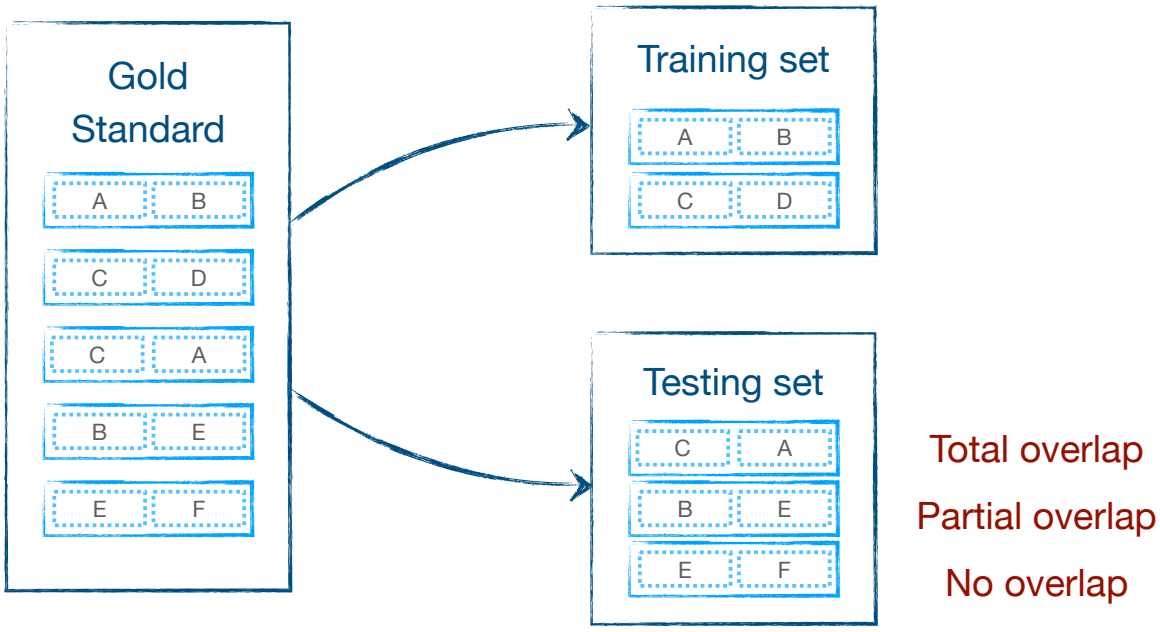


Figure 2: Example of protein-level overlap.
In this example, (A, B, C … F) are proteins, and the gold standard is composed of five pairs (A-B, C-D, C-A etc.). The first two pairs are allocated to the training set, while the rest is used for testing. In this case, both proteins A and C are in the training set, and although the pairing C-A is new, both proteins are already known (total overlap). Conversely, neither E nor F is used for training, so the pair E-F is completely new to testing (no overlap). Protein B is in the training set while E is not: B-E is an example of partial overlap.

Creation of a gold standard

Positive examples: PPIs

When building a gold standard for machine learning, quality and representativity are the most important aspects to consider, making IntAct [28] a database of choice for interacting proteins. It aggregates reliable evidence of molecular interactions from over 20,000 publications, which are manually curated, and includes data from other interaction databases such as the ones in the IMEx consortium [29]. It also quantifies the reliability of each interaction with the MIscore [30], a quality score that considers the detection method, the interaction type and the number of publications reporting it (each with a confidence coefficient). The reliability of the detection method and interaction type is based on the assessment of the HUPO PSI-MI consortium, each method or interaction type ontology term is assigned a confidence score [31]. For example, PPIs detected by colocalisation have a method score of 0.33 while biochemical methods (coimmunoprecipitation, TAP etc.) have a score of 1.0 and post-transcriptional interference techniques only 0.1. Independent detections from different methods increase the final score and the MIscore rates experimental evidence higher compared to predictive algorithms or literature text-mining methods. The number of publications contributes to the score on a log scale.

I downloaded data from the EMBLE-EBI FTP server (timestamp: 15/10/2021) and restricted the observations to human interactions between proteins with UniProt IDs. I also removed homodimeric interactions. Then, to limit the noise in the dataset, I focused on high confidence interactions. One source of errors is how complexes are represented. Indeed, when interaction data are based on experimental methods detecting complexes (mass spectrometry for example), this information is converted to a set of interaction pairs. There are several ways to do so, the two most common methods are the matrix and spoke expansion models; in the matrix model, it is assumed that all the proteins in the complex interact with one another, while in the spoke model, only one protein (usually the bait) interacts with all the other ones. The spoke expansion has been shown to be threefold more accurate than the matrix model [32] but remains a potential source of errors and

results in lower confidence scores (**Figure 4.a**). Therefore, I removed spoke complex expansions from the dataset.

Interestingly, interactions detected by coimmunoprecipitation or TAP tend to be scored lower than interactions detected by Y2H (**Figure 4.b**), while other parameters such as host organisms did not show differences (**Figure 4.c**). In case of PPIs with multiple entries, the highest MIscore was used. This quality control step left 128,790 PPIs, covering 15,506 proteins (out of 20,386 in UniProt).

Chapter 3 will discuss the particular importance of highly connected proteins (hubs). Based on this cleaned dataset, I created an index of the number of recorded interactions per protein and made a list of hubs. In line with the literature, hubs are defined as the 20% of proteins with the most interactions [33], here equivalent to proteins with more than 21 partners.

Finally, when looking at the distribution of the MIscores in the dataset (**Figure 3**), a natural threshold of 0.47 was visible, which restricted the final dataset to 78,229 interactions, covering 12,026 proteins. In **Chapter 3**, I also discuss the impact that this filter has on the analyses, and show that the results presented there still stand when keeping the whole dataset.

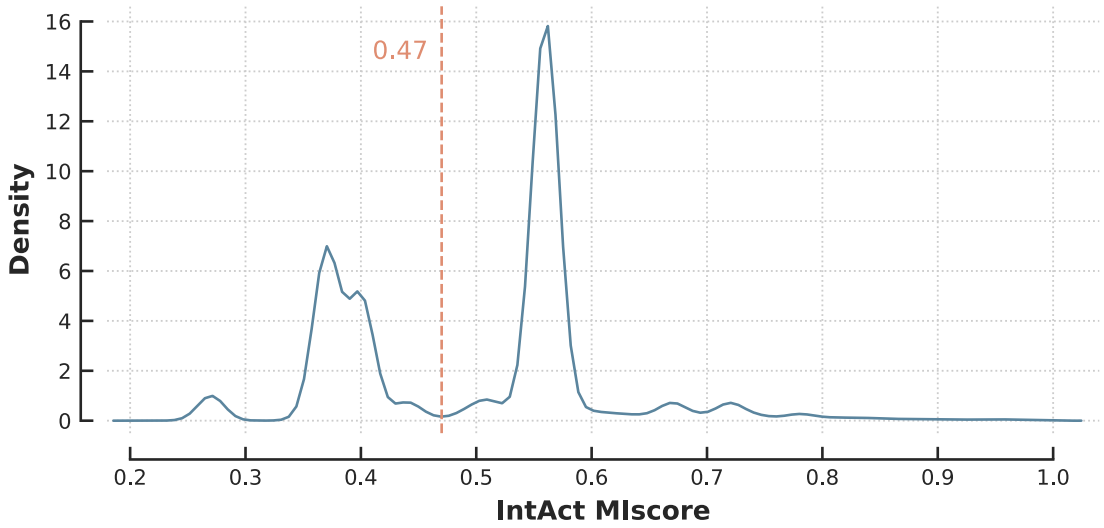


Figure 3: Distribution of the MIscore in IntAct.

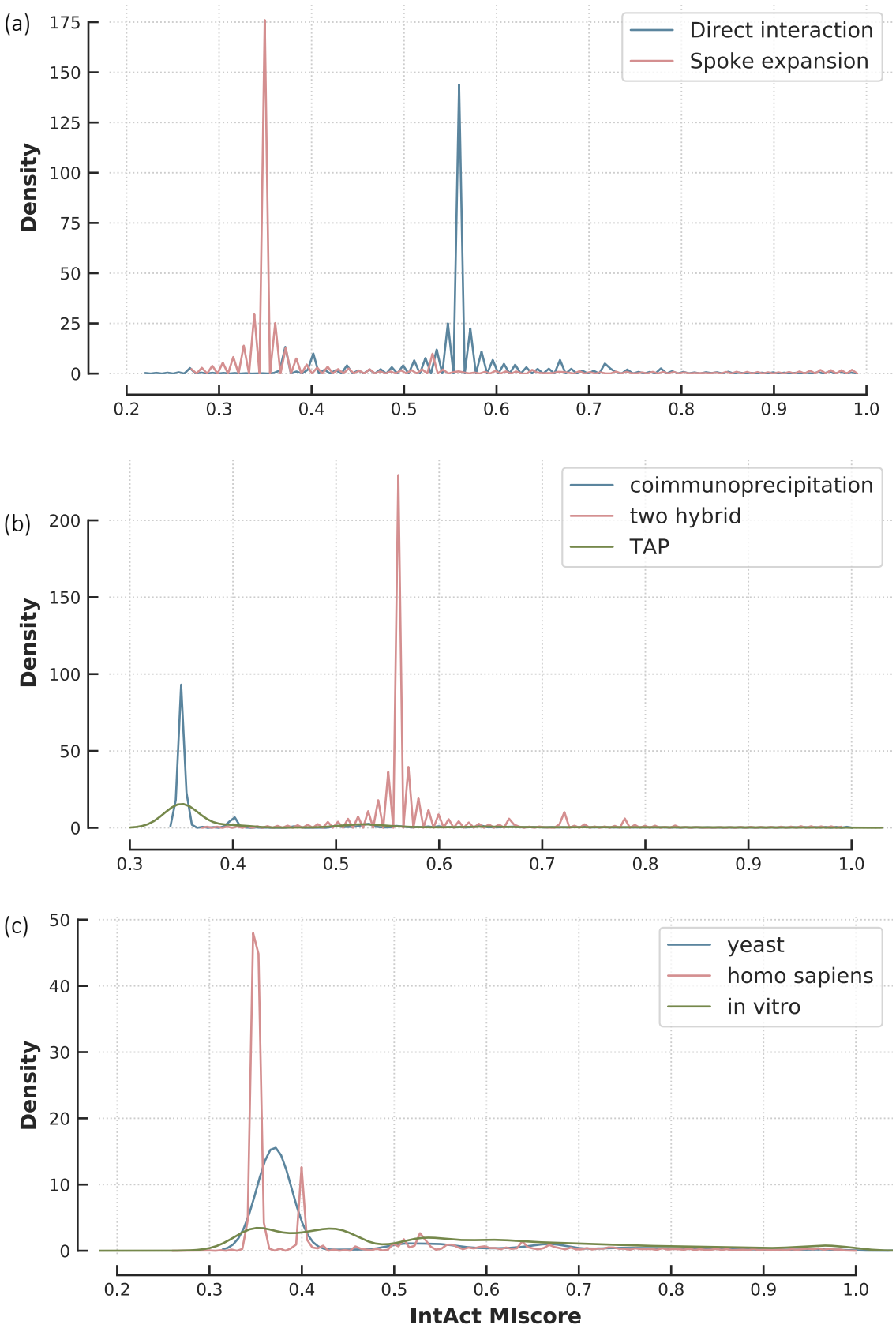


Figure 4: Distribution of IntAct's MIscore depending in different subgroups. (a) Direct interactions vs Spoke expansions, (b) different detection methods and (c) different host organisms.

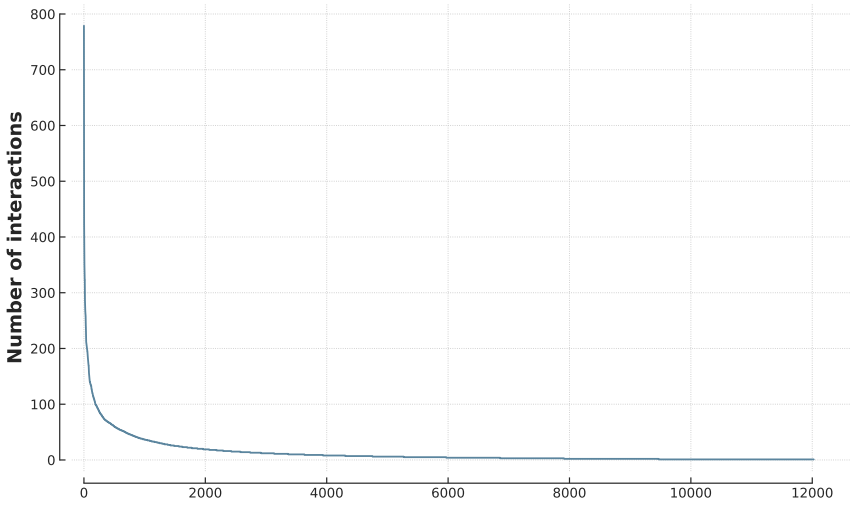


Figure 5: Distribution of proteins' degree in IntAct.
The exponential decrease is characteristic of a scale-free network.

Negative examples: non-interacting proteins

Considering that selecting non-interacting proteins based on their (supposed) cellular locations has been proved unreliable and the limited size of databases of non-interactions, randomly sampling protein pairs remains the approach with the lowest probability of error [34]. Indeed, the PPI network is extremely scarce; it has been estimated that only ~0.15% of the possible pair-wise interactions in humans has been observed [34], and even when accounting for missing data, the probability of false negative remains extremely low.

Non-interacting proteins can be sampled using a uniform distribution, i.e. all proteins have an equal probability of being selected, which leads to an unbiased set representative of the general population of protein pairs. However, PPI networks are known to be similar to scale-free networks, i.e. composed of a few highly connected nodes, called *hubs*, and numerous *lone* proteins with few interactions [35] (Figure 5). Consequently, hubs are over-represented in a set of PPIs. For example, in the curated set from IntAct, the top 20% of proteins by number of interactions were involved in 94% of PPIs. But when uniformly sampling protein pairs, the same top 20% were only involved in 37% of non-interacting proteins. This discrepancy can be an issue for machine learning algorithms that would recognise hubs and systematically predict a positive interaction when a hub is involved. Such a strategy would maximise accuracy on the training set but lead to a majority of false positives when making predictions on new pairs. To mitigate this, balanced sampling [36]

uses a probability of sampling a protein for the negative set proportionate to its frequency in the positive set. It has been shown that each strategy serves a different purpose [24]; balanced sampling is beneficial for training models but should not be used for evaluating them, as the induced sampling bias makes metrics less meaningful. This strategy was implemented here, where non-interacting proteins were selected with balanced sampling for the training set and uniform sampling for the testing sets.

Train/test split

In the presence of limited data, the division of the gold standard between training and testing sets is critical to simultaneously optimise learning and obtain meaningful generalisation metrics. Here, the testing set should achieve several objectives, (1) provide performance metrics on a new, independent set, (2) measure the impact, or absence of impact, of protein-level overlap, (3) demonstrate how the model can generalise to real-world data. Since a single set cannot achieve simultaneously (2) and (3), as the careful selection of proteins to measure overlap biases the dataset, I designed two testing sets *T1* and *T2*. *T1* should be used to compare different approaches with an independent set and investigate protein-level overlap, and *T2* should be used to assess generalisation. *T2*, with ten times more negative examples than positive ones (Table 1), can then be used to assess how models perform in an imbalanced setting where positive interactions are rare compared to non-interacting proteins.

T1 was built by purposefully leaving some proteins out of the training set. To demonstrate the importance of doing this, I first divided training and testing sets conventionally using the popular *scikit-learn* library [37], which resulted in 95% of pairs in the testing set sharing both proteins with the training set (Figure 6), and 99.9% sharing at least one proteins, which may lead to overestimating performances [26]. Then I measured the protein-level overlap when setting aside some proteins for testing, which resulted in 77% of pairs having at most one protein in common (Figure 6).

To achieve this better split, 1,562 proteins (13%) were randomly set aside for the testing set. This percentage was chosen as it gives a train/test ratio of ~70%/30%, which is standard when data is limited. In Chapter 3, I also show that perturbating this ratio slightly does not affect the conclusions. The dataset was then randomly divided under this constraint and

included 53,331 PPIs in the training set, 12,449 in *T1* and the same in *T2* (Table 1). Table 2 shows the detection methods that produced the PPIs in the different datasets. The training set and *T1* both have 50% of positive examples, while *T2* has ten times more non-interacting proteins than interacting ones.

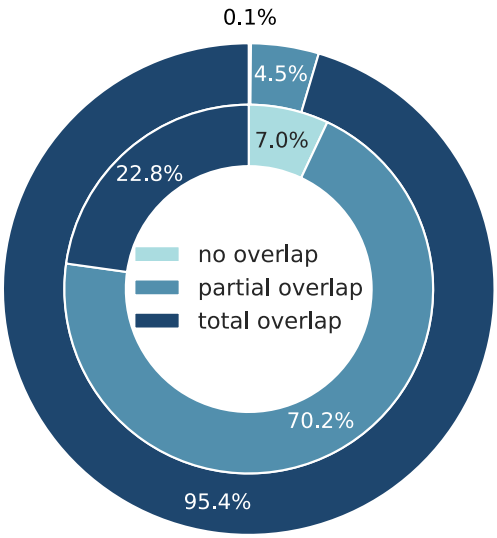


Figure 6: The impact of train/test splitting strategies on protein-level overlap.
The common splitting strategy is to allocate pairs randomly (outer ring), while here, I set aside proteins for testing (inner ring).

Table 1: Sample sizes in B4PPI

Set	Number of examples (% of positive)	
	B4PPI-Human	Yeast dataset
Training	106,662 (50%)	60,738 (50%)
T1	24,898 (50%)	25,398 (50%)
T2	136,939 (9%)	N/A

Table 2: Detection methods used for PPIs in the training and testing sets.
(When a PPI was reported by multiple sources, only the method that resulted in the highest MIscore was counted in the table below)

Detection method	Dataset		
	Training	T1	T2
Two-Hybrid	49,917 (86%)	10,644 (86%)	10,629 (86%)
Coimmunoprecipitation	3,045 (5.7%)	786 (6.3%)	835 (6.7%)
Pull down	1,328 (2.5%)	319 (2.6%)	309 (2.5%)
TAP	127 (0.24%)	30 (0.24%)	30 (0.25%)
Other	2,792 (5.2%)	657 (5.3%)	628 (5.0%)

Choice of metrics

The choice of metrics is a crucial element of a benchmark. It has been argued for a long time that summarising the results by a single number (such as accuracy or AUROC) can be misleading [38], especially here where the use cases are so varied; for some applications, the risk of false positive should be kept to a minimum while for others a balance between precision and recall is important. The Receiver Operating Characteristic (ROC) and the Precision-Recall (PR) curves are complementary options for PPI prediction (e.g. **Figure 1**, panels a and b). The ROC curve is unaffected by the prevalence of interacting proteins, a benefit as the true prevalence of PPIs is mostly unknown; however, it also means that both classes are considered equally, whereas often, PPIs are more interesting than non-interacting proteins. This is addressed by the PR curve, where precision emphasizes positive examples. It is also worth noting that the two metrics can sometimes disagree, e.g. in cases with a low prevalence of PPIs, where a model can have simultaneously high specificity and low precision (good ROC curve but poor PR curve). In some cases, slightly different plots may be useful, for example comparing true positives to false positives (instead of the rates) for the most confident predictions (**Figure 7.a**), or zooming in on the top left corner of a PR curve (high precision/low recall area) (**Figure 7.b**).

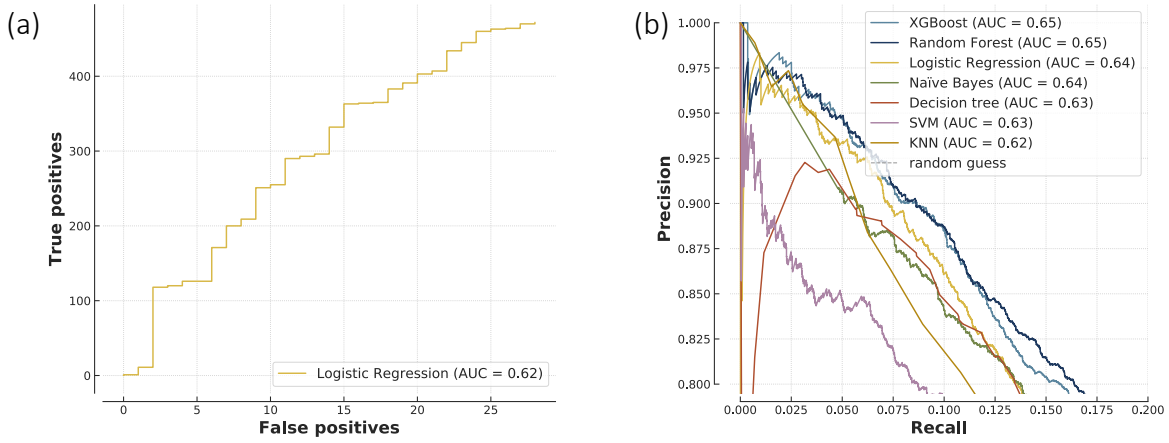


Figure 7: Additional performance plots.
(a) shows true positive vs false positives for the 500 most confident predictions and (b) zooms on the high-performance/low-recall area of a PR curve (the corresponding results are described in Chapter 3).

The importance and difficulty of including metrics of computational efficiency

Although metrics measuring accuracy (in the general sense of performance, not necessarily the statistical definition of accuracy) are essential, it is a common mistake to ignore other aspects of performance, such as computational efficiency [18]. As we will see in Chapter 3, some PPI prediction algorithms train in a couple of minutes while others need several hours. With the current and urgent threat of climate change, acknowledging and reducing the carbon footprint of algorithms is paramount. However, contrary to other metrics presented here, there was no tool available that could be included in B4PPI to estimate carbon footprints and I had to build one first; I detail this in Chapter 4.

In a nutshell, the Green Algorithms online calculator (www.green-algorithms.org) provides estimations for the carbon footprint of computations in terms of carbon dioxide equivalent (CO_2e) and the number of months it would take a tree to sequester the GHG emissions (tree-months). With this tool, carbon footprints can be included in benchmarking results alongside ROC and PR curves (Figure 1).

Pre-computed features for fast development

The elements described above represent the minimum needed for reproducible benchmarks [18], and researchers who wish to use their own input features can evaluate their models on these partitions. However, to rapidly test a new model, it is useful to have access to carefully selected and highly accurate protein properties. In the absence of such datasets, testing a new idea may take weeks to simply pull the data together.

The two main categories of features used are amino acid sequences and functional genomics (FG) annotations, such as subcellular localisation and biological functions. These are available with B4PPI from the professionally curated databases UniProt, the Human Protein Atlas (HPA) [39], [40] and Bgee [41] (Table 3).

Table 3: Features available in B4PPI-Human.

Feature	Number of different annotations	Missing values (/20,386)	Source
Biological processes (GO*)	12,248	3,338	UniProt [27]
Cellular components (GO)	1,754	1,765	UniProt
Molecular functions (GO)	4,346	4,552	UniProt
Domains	2,313	11,815	UniProt
Motifs	819	18,103	UniProt
Sequence	N/A	0	UniProt
Gene expression profile	1,147	1,296	Bgee [41]
Tissue IHC data	62	9,536	HPA [39], [40]
Tissue and cell type	189	9,536	HPA
RNAseq	61	1,448	HPA
Subcellular location	33	7,820	HPA

*Gene ontology

Raw features

Protein sequences in humans are well documented and can be obtained from UniProt, but FG features can be more challenging as they should be diverse (i.e. cover a wide range of properties), of high-quality and have high coverage (i.e. few missing proteins). For the same reasons described above, aggregated, manually curated and professionally reviewed databases are preferred. Based on features that have been successfully used for the task before, I chose to include information about cellular and tissue localisation, biological functions, domain annotations and gene expression patterns [7], [10], [12], [42].

One of the main databases on proteins is UniProt [27] and its knowledgebase, UniProtKB. Swiss-Prot, the section of UniProtKB that is reviewed and manually curated, was used in this work to ensure optimal quality. The data from Swiss-Prot was downloaded through their API by restricting to reviewed, non-obsolete human proteins (last download: 09/11/2021). The different columns were then cleaned to extract the information of interest in a standardised format, and I used UniProt IDs throughout. There was information for 20,386 proteins and more details about each feature are in Table 3. UniProt's API was also used to map UniProt IDs between different databases and map outdated identifiers. In particular, I extracted amino acids sequences for each protein, more than 95% of which came from the translation of coding sequences submitted to the

International Nucleotide Sequence Database Collaboration [43]. Annotated domains and motifs were also extracted, alongside gene ontology (GO) annotations of biological processes, cellular components and molecular functions. For a protein, each of the FG features was represented as a bag-of-words, i.e. a sparse vector of length equal to the number of annotations in the database. Some features, such as domain and motifs, showed a high rate of missing values, the impact of which needs to be assessed on a case-by-case basis. In Chapter 3, I showed that in this case, it didn't influence the results.

When working with gene expression data, biological and technical noise need to be accounted for correctly. The Bgee public repository [41] does that by regrouping curated healthy wild-type standardised gene expression patterns. The human data is mainly from GTEx v6 (phs000424.v6.p1), with an added layer of manual curation to remove unhealthy subjects. For a gene, the Bgee data provides binary calls of presence or absence of expression for each combination of anatomical entity and developmental stage. I downloaded the database from their FTP server (version 14.2) and obtained information for 59,777 genes, 320 anatomical entities and 33 developmental stages, which led to 1,147 stage/entity combinations. The Bgee entries were matched to the UniProt IDs using UniProt's mapping table.

The Human Protein Atlas (HPA) [39], [40] provides data mapping human proteins to tissues and cells. In particular, I used the Tissue Atlas [39] which presents the distribution of proteins in tissues and cell types and the Cell Atlas [40] which contains the distribution across subcellular locations. The Tissue Atlas contains data similar to Bgee, but the overlap is likely limited (Figure 8) as the two databases only share GTEx RNAseq data. While Bgee has a more thorough curation process, HPA contains a lot of original in-house experimental results, which justifies the inclusion of both data sources. I downloaded the HPA data from their website (release 20.1, Ensembl version 92.38). The data in HPA is identified by Ensembl gene IDs, which were mapped to UniProt IDs using UniProt's API. I restricted the dataset to the reviewed proteins present in Swiss-Prot and, to ensure the quality of annotations, I discarded the entries HPA's curators annotated as "uncertain". For the tissue IHC data, I mapped expression levels to numerical values (high=3, medium=2, low=1 and "not detected"=-1), with untested tissues being mapped to 0. Similar pre-processing was used for the consensus RNAseq data and the subcellular location.

Pre-processing to measure features similarity

Both sequence data and FG annotations are encoded by features of high dimension (either the length of a sequence or the total number of annotations in the ontology), which comes with modelling challenges. For FG annotations, however, the dimension can be easily reduced by computing feature-wise similarities between proteins.

The inspiration for how to do so comes from the fields of information retrieval and natural language processing, where the comparison between bags-of-words is common. I chose to use cosine similarity [44], which measures the distance between two vectors, as it can be computed for 200 million pairs in a reasonable time. For two vectors $A = (A_i)$ and $B = (B_i)$, their cosine similarity $CS(A, B)$ is:

$$CS(A,B) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \sqrt{\sum_i B_i^2}} \quad (1)$$

As a result, for each of the 207,784,305 possible pairs of proteins, I obtained 12 similarity features: biological processes, cell components, molecular function, domains and motifs from UniProt, gene expression from Bgee, tissue/cell expression, tissue expression, RNAseq expression and subcellular locations from the Human Protein Atlas (Table 3).

A correlation map of the features created on the training set (Figure 8) shows that few features have a very high correlation coefficient, confirming that there is limited redundancy. As anticipated, the correlations between Bgee and HPA data were elevated but not close to 1 either (between 0.45 and 0.69). Interestingly, despite representing the same information, the correlation between UniProt and HPA for subcellular compartments was only 0.27. The high correlation between the different HPA features (0.50 to 0.96) is also worth noting. However, as shown in Chapter 3, this high correlation has little impact on the predictive models. Finally, with coefficients between 0.03 and 0.17, these features are only weakly correlated with the label indicating whether the two proteins interact. This confirms that no single feature can prove interaction, but rather several weak indicators are needed for prediction models.

Figure 8: Pearson correlation between the features (and the label) of the training set of B4PPI-Human.

S. cerevisiae data

B4PPI is not species-specific and, provided the data is available, can be easily used to create benchmarking gold standards for different organisms. For example, I built a similar benchmarking set for yeast to study inter-species variations (Chapter 3), following the same pipeline described above for B4PPI-Human.

UniProt lists 6,721 *S. cerevisiae* proteins and the same information as for humans (Table 4), but HPA and Bgee do not include data for this organism. PPIs were obtained from IntAct following the same procedure, although no selection based on MIscores was made

considering the absence of an obvious choice when looking at the distribution (**Figure 9**). The final PPI dataset comprised 43,068 interactions covering 5,679 proteins.

The split between training and testing sets was done similarly by setting aside 737 proteins for testing and then randomly allocating PPIs to keep 30,369 PPIs for training (70% of the gold standard) (**Table 1**). Because there are fewer data on yeast, and only one testing set was needed to replicate the analysis conducted on humans, dividing the remaining 12,699 further between *T1* and *T2* was not suitable here. But if the goal was to measure the generalisability of a yeast model with *T2*, this could easily be done.

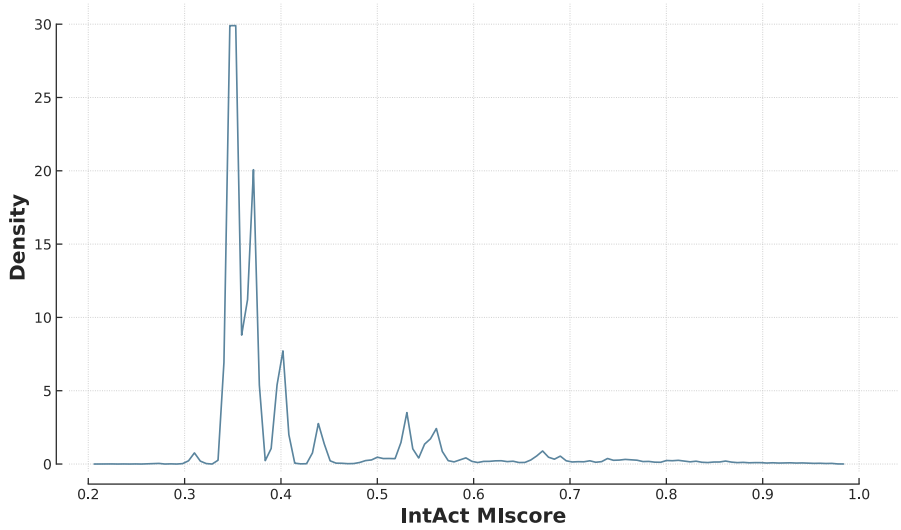


Figure 9: Distribution of IntAct's MIscore in the yeast dataset.

Table 4: Details of the features used for B4PPI-Yeast.

Feature	Number of different annotations	Missing values (/6,721)	Source
Biological processes (GO)	3,114	1,510	UniProt [27]
Cellular components (GO)	820	839	UniProt
Molecular functions (GO)	2,079	2,242	UniProt
Domains	606	5,135	UniProt
Motifs	181	6,273	UniProt
Sequence	N/A	0	UniProt

Discussion

Analyses of PPI prediction models require a robust and reliable benchmarking pipeline. I designed the first one of its kind, B4PPI, which accounts for a range of biological and statistical pitfalls. By being freely accessible, open-source and using standard identifiers for proteins, B4PPI can be used by any researchers working on *in silico* PPI prediction to assess performance and compare their approach to the state-of-the-art. An example reporting sheet includes relevant metrics, from PR and ROC curves to runtime and carbon footprint, to ensure the models released can be trusted and to encourage wider use of PPI imputation for downstream analyses. B4PPI also comes with pre-processed features to enable the rapid development of new ideas through trial and error. B4PPI can also be combined with recent efforts to create alternate benchmarking sets such as HiPPIP [16].

While a benchmarking standard for PPI prediction is needed, it is important to remember the downsides of benchmarks, as demonstrated in computer vision or natural language processing. A fixed set of metrics can motivate the community to overly focus on those at the expense of applicability and usefulness. To limit this, B4PPI includes a range of metrics, but the relevant indicators for each use case should nonetheless be carefully considered.

Some limitations remain. For example, although I addressed the issue of protein-level overlap (i.e. shared proteins between the training and testing sets), similarities in protein sequences or their biological properties could still impact performance metrics [45]. Besides, while relying on large aggregated databases reduces measurement bias, sampling bias remains as the set of interacting proteins remains largely made of popular (overstudied) proteins, and the impact on modelling has been underestimated [46]. The selection of pre-computed features is also not exhaustive; some information such as phylogenetic profiles or chemical properties of amino acids could be added in the future.

The size and complexity of the PPI network make *in silico* prediction tools indispensable. However, it is important to ensure that the models developed are reliable and readily available to the community for downstream analysis and give insights into biological pathways. The inference mechanisms underpinning these models also need to be better understood, which takes us to the next chapter.

References

[1] H. Goehler *et al.*, 'A Protein Interaction Network Links GIT1, an Enhancer of Huntingtin Aggregation, to Huntington's Disease', *Molecular Cell*, vol. 15, no. 6, pp. 853–865, Sep. 2004, doi: 10.1016/j.molcel.2004.09.016.

[2] A. Vinayagam *et al.*, 'A Directed Protein Interaction Network for Investigating Intracellular Signal Transduction', *Science Signaling*, vol. 4, no. 189, pp. rs8–rs8, Sep. 2011, doi: 10.1126/scisignal.2001699.

[3] M. Bakail and F. Ochsenbein, 'Targeting protein–protein interactions, a wide open field for drug design', *Comptes Rendus Chimie*, vol. 19, no. 1–2, pp. 19–27, Jan. 2016, doi: 10.1016/j.crci.2015.12.004.

[4] S. Rapposelli, E. Gaudio, F. Bertozzi, and S. Gul, 'Editorial: Protein–Protein Interactions: Drug Discovery for the Future', *Front. Chem.*, vol. 9, p. 811190, Nov. 2021, doi: 10.3389/fchem.2021.811190.

[5] K. Luck *et al.*, 'A reference map of the human binary protein interactome', *Nature*, vol. 580, no. 7803, pp. 402–408, Apr. 2020, doi: 10.1038/s41586-020-2188-x.

[6] E. L. Huttlin *et al.*, 'Dual proteome-scale networks reveal cell-specific remodeling of the human interactome', *Cell*, vol. 184, no. 11, pp. 3022-3040.e28, May 2021, doi: 10.1016/j.cell.2021.04.011.

[7] R. Jansen, 'A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data', *Science*, vol. 302, no. 5644, pp. 449–453, Oct. 2003, doi: 10.1126/science.1087361.

[8] L. V. Zhang, S. L. Wong, O. D. King, and F. P. Roth, 'Predicting co-complexed protein pairs using genomic and proteomic data integration', *BMC Bioinformatics*, p. 15, 2004.

[9] X.-W. Chen and M. Liu, 'Prediction of protein–protein interactions using random decision forest framework', *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, Dec. 2005, doi: 10.1093/bioinformatics/bti721.

[10] A. Ben-Hur and W. S. Noble, 'Kernel methods for predicting protein-protein interactions', *Bioinformatics*, vol. 21, no. Suppl 1, pp. i38–i46, Jun. 2005, doi: 10.1093/bioinformatics/bti1016.

[11] M. S. Scott and G. J. Barton, 'Probabilistic prediction and ranking of human protein-protein interactions', *BMC Bioinformatics*, vol. 8, no. 1, p. 239, Jul. 2007, doi: 10.1186/1471-2105-8-239.

[12] M. Kotlyar *et al.*, '*In silico* prediction of physical protein interactions and characterization of interactome orphans', *Nature Methods*, vol. 12, no. 1, pp. 79–84, Jan. 2015, doi: 10.1038/nmeth.3178.

[13] Y. Murakami and K. Mizuguchi, 'Homology-based prediction of interactions between proteins using Averaged One-Dependence Estimators', *BMC Bioinformatics*, vol. 15, no. 1, p. 213, Jun. 2014, doi: 10.1186/1471-2105-15-213.

[14] Z.-H. You, M. Zhou, X. Luo, and S. Li, 'Highly Efficient Framework for Predicting Interactions Between Proteins', *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 731–743, Mar. 2017, doi: 10.1109/TCYB.2016.2524994.

[15] M. Chen *et al.*, 'Multifaceted protein–protein interaction prediction based on Siamese residual RCNN', *Bioinformatics*, vol. 35, no. 14, pp. i305–i314, Jul. 2019, doi: 10.1093/bioinformatics/btz328.

[16] B. Dunham and M. K. Ganapathiraju, 'Benchmark Evaluation of Protein–Protein Interaction Prediction Algorithms', *Molecules*, vol. 27, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/molecules27010041.

[17] M. Kotlyar, A. E. M. Rossos, and I. Jurisica, 'Prediction of Protein-Protein Interactions', *Current Protocols in Bioinformatics*, vol. 60, no. 1, p. 8.2.1-8.2.14, 2017, doi: 10.1002/cpbi.38.

[18] S. Mangul *et al.*, 'Systematic benchmarking of omics computational tools', *Nat Commun*, vol. 10, no. 1, p. 1393, Dec. 2019, doi: 10.1038/s41467-019-09406-4.

[19] L. Lannelongue and M. Inouye, 'Construction of in silico protein-protein interaction networks across different topologies using machine learning', *bioRxiv*. p. 2022.02.07.479382, Feb. 09, 2022. doi: 10.1101/2022.02.07.479382.

[20] T. Sun, B. Zhou, L. Lai, and J. Pei, 'Sequence-based prediction of protein protein interaction using a deep-learning algorithm', *BMC Bioinformatics*, vol. 18, no. 1, p. 277, May 2017, doi: 10.1186/s12859-017-1700-2.

[21] F. Li, F. Zhu, X. Ling, and Q. Liu, 'Protein Interaction Network Reconstruction Through Ensemble Deep Learning With Attention Mechanism', *Front Bioeng Biotechnol*, vol. 8, p. 390, May 2020, doi: 10.3389/fbioe.2020.00390.

[22] A. Ben-Hur and W. Noble, 'Choosing negative examples for the prediction of protein-protein interactions', *BMC Bioinformatics*, vol. 7, no. Suppl 1, p. S2, 2006, doi: 10.1186/1471-2105-7-S1-S2.

[23] P. Blohm *et al.*, 'Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis', *Nucleic Acids Res*, vol. 42, no. Database issue, pp. D396–D400, Jan. 2014, doi: 10.1093/nar/gkt1079.

[24] Y. Park and E. M. Marcotte, 'Revisiting the negative example sampling problem for predicting protein–protein interactions', *Bioinformatics*, vol. 27, no. 21, pp. 3024–3028, Nov. 2011, doi: 10.1093/bioinformatics/btr514.

[25] L. Hu, X. Wang, Y.-A. Huang, P. Hu, and Z.-H. You, 'A survey on computational models for predicting protein–protein interactions', *Briefings in Bioinformatics*, vol. 22, no. 5, Sep. 2021, doi: 10.1093/bib/bbab036.

[26] Y. Park and E. M. Marcotte, 'Flaws in evaluation schemes for pair-input computational predictions', *Nature Methods*, vol. 9, no. 12, pp. 1134–1136, Dec. 2012, doi: 10.1038/nmeth.2259.

[27] The UniProt Consortium, 'UniProt: the universal protein knowledgebase in 2021', *Nucleic Acids Research*, vol. 49, no. D1, pp. D480–D489, Jan. 2021, doi: 10.1093/nar/gkaa1100.

[28] S. Orchard *et al.*, 'The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases.', *Nucleic Acids Res*, vol. 42, no. Database issue, pp. D358-63, Jan. 2014, doi: 10.1093/nar/gkt1115.

[29] S. Orchard *et al.*, 'Protein interaction data curation: the International Molecular Exchange (IMEx) consortium', *Nature Methods*, vol. 9, no. 4, pp. 345–350, Apr. 2012, doi: 10.1038/nmeth.1931.

[30] B. Aranda *et al.*, 'PSICQUIC and PSISCORE: accessing and scoring molecular interactions', *Nat Methods*, vol. 8, no. 7, pp. 528–529, Jun. 2011, doi: 10.1038/nmeth.1637.

[31] 'IntAct - User Guide'. https://www.ebi.ac.uk/intact/documentation/user-guide#interaction_scoring (accessed Dec. 30, 2022).

[32] G. D. Bader and C. W. V. Hogue, 'Analyzing yeast protein–protein interaction data obtained from different sources', *Nat Biotechnol*, vol. 20, no. 10, pp. 991–997, Oct. 2002, doi: 10.1038/nbt1002-991.

[33] G. Jin, S. Zhang, X.-S. Zhang, and L. Chen, 'Hubs with Network Motifs Organize Modularity Dynamically in the Protein-Protein Interaction Network of Yeast', *PLoS ONE*, vol. 2, no. 11, p. e1207, Nov. 2007, doi: 10.1371/journal.pone.0001207.

[34] M. Agrawal, M. Zitnik, and J. Leskovec, 'Large-scale analysis of disease pathways in the human interactome', *Pac Symp Biocomput*, vol. 23, pp. 111–122, 2018.

[35] EMBL-EBI, 'Properties of PPINs: scale-free networks | Network analysis of protein interaction data'. https://www.ebi.ac.uk/training/online/courses/network-analysis-of-protein-interaction-data-an-introduction/protein-protein-interaction-networks/properties-of-ppins-scale-free-networks/ (accessed Nov. 29, 2021).

[36] J. Yu, M. Guo, C. J. Needham, Y. Huang, L. Cai, and D. R. Westhead, 'Simple sequence-based kernels do not predict protein-protein interactions', *Bioinformatics*, vol. 26, no. 20, pp. 2610–2614, Oct. 2010, doi: 10.1093/bioinformatics/btq483.

[37] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[38] F. J. Provost, T. Fawcett, and R. Kohavi, 'The Case against Accuracy Estimation for Comparing Induction Algorithms', in *Proceedings of the Fifteenth International Conference on Machine Learning*, San Francisco, CA, USA, Jul. 1998, pp. 445–453.

[39] M. Uhlén *et al.*, 'Tissue-based map of the human proteome', *Science*, vol. 347, no. 6220, Jan. 2015, doi: 10.1126/science.1260419.

[40] P. J. Thul *et al.*, 'A subcellular map of the human proteome', *Science*, vol. 356, no. 6340, May 2017, doi: 10.1126/science.aal3321.

[41] F. B. Bastian *et al.*, 'The Bgee suite: integrated curated expression atlas and comparative transcriptomics in animals', *Nucleic Acids Research*, vol. 49, no. D1, pp. D831–D847, Jan. 2021, doi: 10.1093/nar/gkaa793.

[42] Q. C. Zhang *et al.*, 'Structure-based prediction of protein–protein interactions on a genome-wide scale', *Nature*, vol. 490, no. 7421, pp. 556–560, Oct. 2012, doi: 10.1038/nature11503.

[43] M. Arita, I. Karsch-Mizrachi, G. Cochrane, and on behalf of the International Nucleotide Sequence Database Collaboration, 'The international nucleotide sequence database collaboration', *Nucleic Acids Research*, vol. 49, no. D1, pp. D121–D124, Jan. 2021, doi: 10.1093/nar/gkaa967.

[44] A. Singhal, 'Modern Information Retrieval: A Brief Overview', p. 9, 2001.

[45] T. Hamp and B. Rost, 'More challenges for machine-learning protein interactions', *Bioinformatics*, vol. 31, no. 10, pp. 1521–1525, May 2015, doi: 10.1093/bioinformatics/btu857.

[46] Z. Wang, N. R. Clark, and A. Ma'ayan, 'Dynamics of the discovery process of protein-protein interactions from low content studies', *BMC Syst Biol*, vol. 9, p. 26, Jun. 2015, doi: 10.1186/s12918-015-0173-z.

CHAPTER 3

UNDERSTANDING PPI PREDICTION MODELS: THE ROLE OF NETWORKS TOPOLOGY

Contents

Introduction

In the previous two chapters, I discussed the importance of machine learning tools for PPI prediction and the wide range of models published. However, it remained unclear why models with similar performance make vastly different predictions. Part of the issue was the lack of a standardised framework to train and evaluate PPI prediction algorithms, and I designed B4PPI to address this. In addition to being a benchmarking tool to compare new algorithms to the state-of-the-art, B4PPI can be used to systematically investigate the poorly understood inference mechanisms.

In a nutshell, B4PPI comprises interacting and non-interacting protein pairs divided between a training set and two testing sets (*T1* and *T2*). *T1* contains 50% of positive examples and is used to compare models and investigate the impact of protein-level overlap (i.e. individual proteins present in both the training and testing sets). *T2*, with ten times more negative examples than positive ones, presents an imbalanced situation closer to the proteome and can be used to show generalisability. B4PPI also comes with recommended metrics, namely precision-recall (PR) and receiver operating characteristic (ROC) curves, runtime and carbon footprint.

Within the B4PPI framework, I studied and compared the main approaches to PPI prediction in humans based on functional genomic (FG) information or amino acid sequences alone. Doing so consistently enabled me to better understand what aspects of the underlying biology are captured by each method. I also investigated several ways to combine these approaches to leverage their complementarity.

This chapter highlights why both perspectives are still relevant today and how each one adapts to the topology of PPI networks. In particular, I show that the presence of highly connected proteins has a drastic impact on prediction models and is an area where FG and sequence models diverge. I also replicate these results between humans and yeasts (*S. cerevisiae*) and show which tools are most suitable for cross-species predictions. These results give critical insight into which models can and should be used in different situations.

Publication and contribution

The majority of the analyses presented in this chapter (except for the section "Combining FG annotations and deep learning tools" and "Sensitivity to modelling choices") have been released as a preprint on *bioRxiv* [1].

L. Lannelongue and M. Inouye,
'Construction of *in silico* protein-protein interaction networks
across different topologies using machine learning'.
bioRxiv, p. 2022.02.07.479382, Feb. 09, 2022.
doi: 10.1101/2022.02.07.479382.

Contribution: I designed and conducted the analyses presented here, with feedback and suggestions from Prof. Michael Inouye and colleagues.

FG-based linear models achieved top performance

I first compared a wide range of algorithms to predict PPIs based on FG annotations. For most FG-based models, features are pre-processed to compute similarity measures between proteins, such as colocalisation (Chapter 2). The low dimensionality of the transformed problem explains the success of standard machine learning algorithms; in particular, Naïve Bayes Classifiers [2], decision trees [3] and Random Forests [4] have been the most popular choices [5]–[8]. Other classic machine learning models are worthy of investigation, such as logistic regression, Support Vector Machine (SVM) [9] and K-Nearest Neighbours (KNN) [10]. Despite the proven track record of such tools, the more recent XGBoost algorithm [11] has been shown to outperform them in other situations like kidney disease diagnostic [12], which motivated its inclusion in this analysis.

The full list of FG features is in Table 1. FG-based machine learning models were trained using the scikit-learn library [13]. Mean imputation was used for models that could not handle missing data (Table 2). Hyperparameter search was done using Weight-and-Bias's Bayesian method [14] to find the optimal settings of each algorithm in a reasonable time (a few minutes). All hyperparameter choices are in Table 2. I used a DeLong nonparametric test [15] to statistically compare ROC curves for a same testing set and reported the p-value. I corrected for multiple testing using a conservative significance threshold of 5×10^{-4}, corresponding to a Bonferroni correction for 100 pairwise comparisons [16].

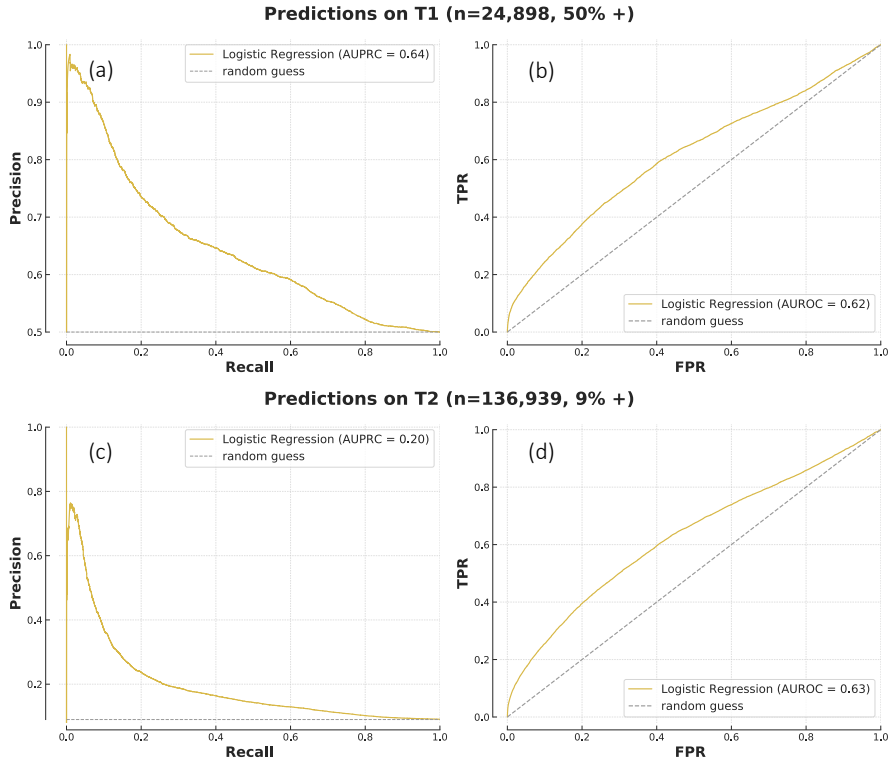
Using logistic regression as a baseline, I reported PR and ROC curves on the two test sets (Figure 1, panels a to d). A list of coordinates for these two curves was made available[1] so that future models can be compared without unnecessary re-training. I also reported the training time[2], 6 seconds, and the carbon footprint, close to 0 gCO_2e. I then ran the same analysis for the other algorithms listed above and produced similar performance sheets and data (Figure 2).

[1] https://github.com/Llannelongue/B4PPI/tree/main/4.%20Results

[2] Models were ran on Cambridge's CSD3 platform [17, p. 3] using 1 NVIDIA A100, 32 cores Xeon Gold 6142, 15GB of memory and a PUE of 1.15.

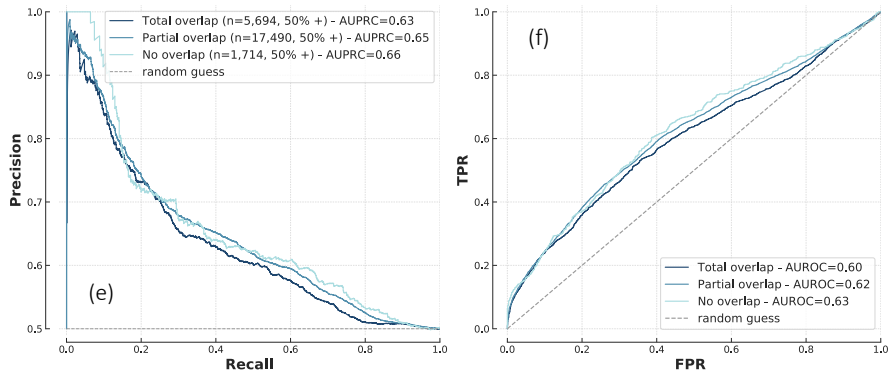
Reporting sheet B4PPI-Human: Logistic Regression

PR and ROC curves on T1* and T2**



* First testing set, used to compare models on an independent set and investigate protein-level overlap.
** Second testing set, used to assess generalisation on an imbalanced dataset (10 times more negative examples than positive ones).

Impact of protein-level overlap



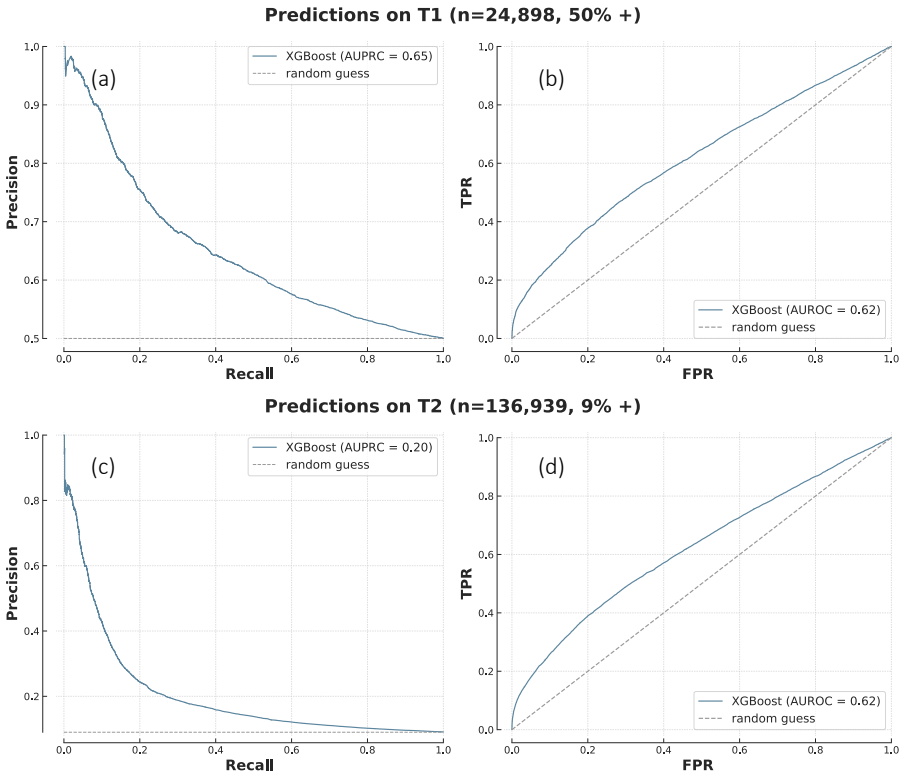
	Running time	Memory	Energy used	Carbon footprint (UK)
Training once	6s	Negligible	< 0.01 kWh	0.007 gCO_2e
Training incl. hyperparameters tuning	same*	Negligible	same	same
Inference	<1s	Negligible	~0	~0

* *No hyperparameters to tune*

Figure 1: Performance sheet of logistic regression on B4PPI-Human.
Panels (a) and (b) show, respectively, the PR and ROC curves on the first testing set *T1*. (c) and (d) show the PR and ROC curves for the second testing set, *T2*. Panels (e) and (f) show the PR and ROC curves broken down by protein-level overlap.

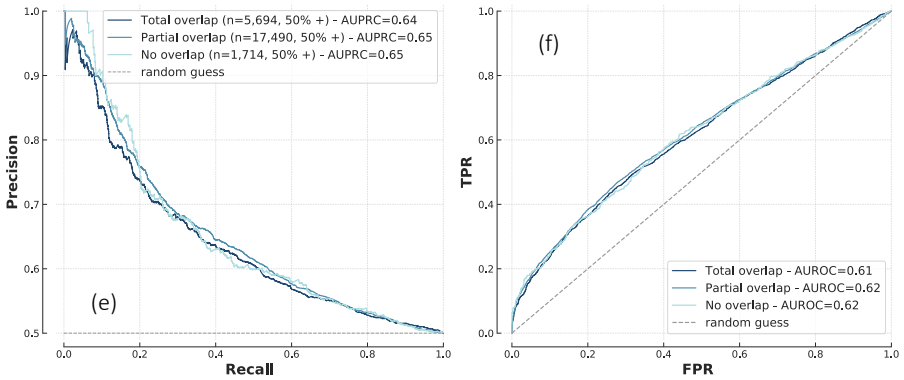
Reporting sheet B4PPI-Human: XGBoost

PR and ROC curves on T1* and T2**



* First testing set, used to compare models on an independent set and investigate protein-level overlap.
** Second testing set, used to assess generalisation on an imbalanced dataset (10 times more negative examples than positive ones).

Impact of protein-level overlap



	Running time	Memory	Energy used	Carbon footprint (UK)
Training once	30s	Negligible	< 0.01 kWh	0.002 gCO_2e
Training incl. hyperparameters tuning	22min	Negligible	< 0.01 kWh	1 gCO_2e
Inference	<1s	Negligible	~0	~0

Figure 2: Performance sheet of XGBoost on B4PPI-Human.

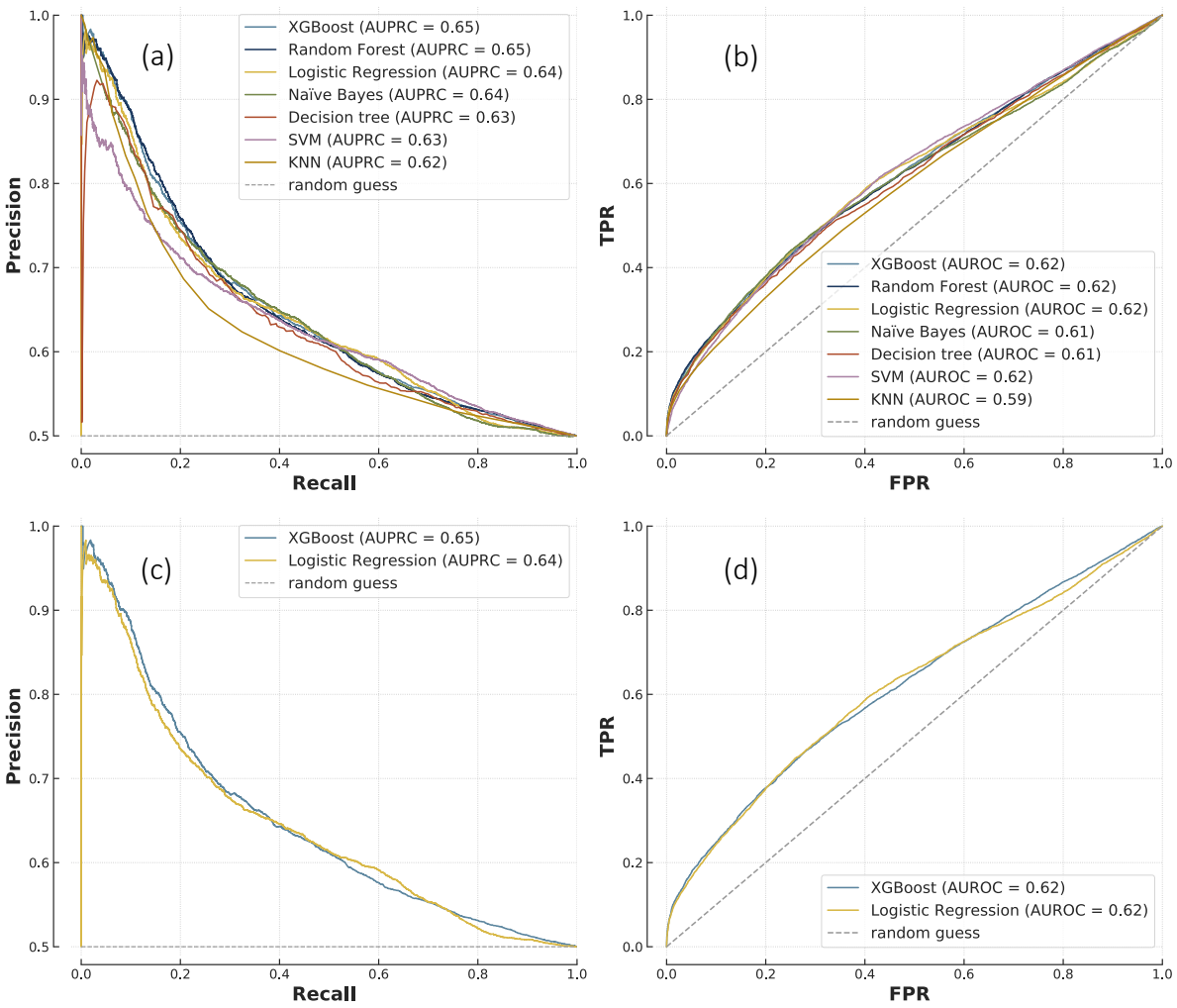


Figure 3: Comparison of FG-based models on *T1* (Human).
PR curves (a, c) and ROC curves (b, d) for all the models tested (a, b) and then only XGBoost and logistic regression for clarity (c, d) (n=24,898, 50% positive).

Table 1: Features available in B4PPI-Human.

Feature	Number of different annotations	Missing values (/20,386)	Source
Biological processes (GO*)	12,248	3,338	UniProt
Cellular components (GO)	1,754	1,765	UniProt
Molecular functions (GO)	4,346	4,552	UniProt
Domains	2,313	11,815	UniProt
Motifs	819	18,103	UniProt
Gene expression profile	1,147	1,296	Bgee
Tissue IHC data	62	9,536	Human Protein Atlas
Tissue and cell type	189	9,536	HPA
RNAseq	61	1,448	HPA
Subcellular location	33	7,820	HPA

When comparing these different models, I found that complex algorithms brought little improvement over logistic regression, as most models performed similarly on *T1* (Figure 3). XGBoost and Random Forest showed minor improvement in AUROC and AUPRC, but the difference between the ROC curves of the logistic regression and XGBoost was non-significant (p=0.27) (Figure 3.b). Moreover, XGBoost was more efficient than Random Forest as it had nearly half the runtime (30s vs 54s). When studying the coefficients of the linear regression, I found that most decisions are based on common biological processes, colocalisation (cellular compartment) and common domains, all three coefficients being significant (p<0.001) (Figure 4).

The reporting standard also enabled me to look at finer performance metrics, broken down by protein-level overlap (i.e. individual proteins common to the training and testing sets). Comparing PR and ROC curves showed that both logistic regression and XGBoost were unaffected by the level of overlap (Figure 1 and Figure 2, panels e and f) and can therefore transfer effectively to new proteins.

	coef	std err	z	P>|z|	[0.025	0.975]
RNAseqHPA	0.0196	0.008	2.526	0.012	0.004	0.035
tissueHPA	-0.0380	0.024	-1.576	0.115	-0.085	0.009
tissueCellHPA	0.0617	0.024	2.614	0.009	0.015	0.108
subcellularLocationHPA	0.0144	0.006	2.235	0.025	0.002	0.027
bioProcessUniprot	0.3038	0.011	27.377	0.000	0.282	0.326
cellCompUniprot	0.2650	0.007	37.125	0.000	0.251	0.279
molFuncUniprot	0.0329	0.008	4.353	0.000	0.018	0.048
domainUniprot	0.1696	0.012	14.470	0.000	0.147	0.193
motifUniprot	0.0235	0.007	3.249	0.001	0.009	0.038
Bgee	0.0196	0.007	2.735	0.006	0.006	0.034

Figure 4: Output of the logistic regression on the training set (B4PPI-Human).

Table 2: Optimal hyperparameters for the models trained on B4PPI-Human.

Algorithm	Missing data imputation	Scaling	Optimal hyperparameters
Logistic Regression	Yes (mean)	Yes	Penalty = none, tol = 0.0001
XGBoost	No	No	colsample_bytree = 0.8059, learning_rate = 0.00002186, max_depth = 29, min_child_weight = 25, n_estimators = 116, subsample = 0.4595
Decision Tree	Yes (mean)	No	Criterion = entropy, min_samples_split = 895, splitter = random
SVM	Yes (mean)	No	C = 1, degree = 3, gamma = scale, kernel = rbf (default values were used due to long runtime)
Random Forest	Yes (mean)	No	Criterion = gini, max_features = log2, min_sample_split = 487, n_estimators = 336
KNN	Yes (mean)	No	Algorithm = brute, leaf_size = 53, n_neighbors = 35, p = 2, weights = uniform
Naïve Bayes	Yes (mean)	No	N/A
Sequence-based Siamese architecture	N/A	N/A	Batch size = 200, gradient_clip_val = 10, RNN = bidirectional GRU, output = linear, hidden size = 512, n_layers = 1, learning rate = 0.001 (GRU) and 0.0001 (output)

Sequence models outperformed FG-based algorithms on known proteins

An alternative to FG-based models is to use amino acids sequences as the input for a PPI prediction algorithm. However, while for FG-based methods, a protein pair could be summarised by just a few numbers thanks to similarity measures, the same approach could not be used here. The inspiration for how to handle sequence pairs comes from computer vision. For automated border controls, the algorithm's goal is not to identify who is at the gate but rather to ensure that the person at the gate matches their passport. As a machine learning problem, this is a lot simpler: while for the former, thousands of images of each citizen would be needed to create personalised prediction models, for the latter, a single algorithm can learn to compare two images, the same way that a custom officer does not need to recognise someone to tell whether they match their passport.

Siamese networks [18], [19] do so with a duplicated embedding step which creates a 1D representation of each input. The distance between the two representations is measured and a prediction is made based on this distance (**Figure 5**). The name comes from the duplicated embedding step, which can be any deep learning architecture. This flexibility makes these methods suitable to handle various types of input. For example for images, Convolutional Neural Networks (CNN) [20] are usually involved. Protein sequences can be

assimilated to sentences, where each amino acid is a word. Like sentences, sequences can be of different lengths and be made of a different combination of amino acids. Recurrent neural networks are the chosen tool for such input, and more specifically, Long-Short Term Memory networks [21] or Gated Recurrent Units (GRU) [22], which have been successful at tasks like machine translation [22] and drug design [23]. CNNs have also been used for PPI predictions by truncating protein sequences [24], but the wide range of sequence lengths makes this approach unsatisfactory. Indeed, the average length of verified human proteins is 558 amino acids, but the standard deviation is 597, with titin being made of as many as 34,350 amino acids [25].

I compared several embedding architectures and found that using GRUs for embedding was preferable because of runtime efficiency and its ability to account for proteins of various lengths. I tested a range of hyperparameters (number of hidden layers, size of the layers, learning rate etc.) and found that networks with a single bidirectional hidden layer were optimal, which is in line with previous findings [26], [27]. Full parameters are in **Table 2** and the open-source code[3]. Models were trained using PyTorch Lightning [28], [29] and tracked with Weight&Bias.

I reported the performance of the optimised Siamese neural network following the B4PPI-Human reporting standard (**Figure 7**). I then compared it to the best performing FG-based model, XGBoost (**Figure 6**). Despite having access to no functional information about the proteins, the sequence model outperformed XGBoost in most situations, except at low recall and high precision (AUPRC=0.68 vs 0.65 and ROC curves significantly different, $p=9x10^{-47}$). However, while XGBoost was trained in only 30s with less than 0.01 kWh of energy, the deep learning approach trained for 1h10 with 0.62 kWh, emitting 22,000 times more greenhouse gases. In addition, the performance of the sequence model was heavily affected by the choice of deep learning architecture and its hyperparameters, such as number of layers and learning rate, which require extensive (and expensive) optimisation. Moreover, while protein-level overlap had no impact on FG-based models, it greatly affected this sequence-based algorithm. The sequence model had an AUPRC of 0.68 on

[3] https://github.com/Llannelongue/B4PPI/tree/main/3.%20Training

average, but 0.75 when restricted to proteins present in both training and testing sets, and only 0.62 when there was no overlap (**Figure 7.e**). This demonstrates that (1) in the absence of specific adjustments, such deep learning models are poorly suited to make predictions on previously unseen proteins and (2) in-depth benchmarks like B4PPI are important to reliably measure performances. While **Figure 6** could indicate that deep learning is the best approach to PPI prediction, it could also be the consequence of unaccounted-for biological properties of PPIs.

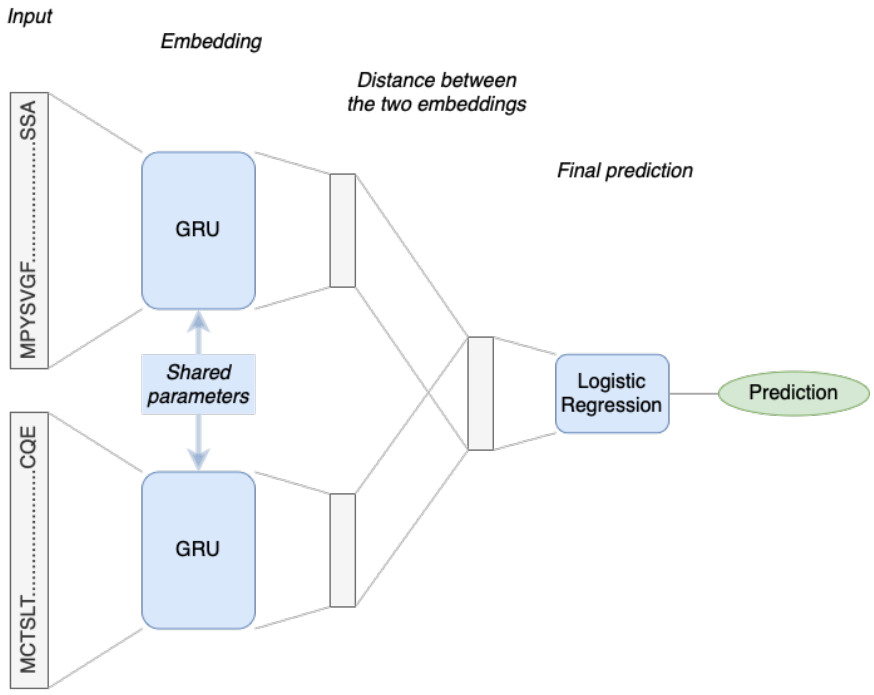


Figure 5: Siamese deep learning architecture to predict PPIs from sequences.

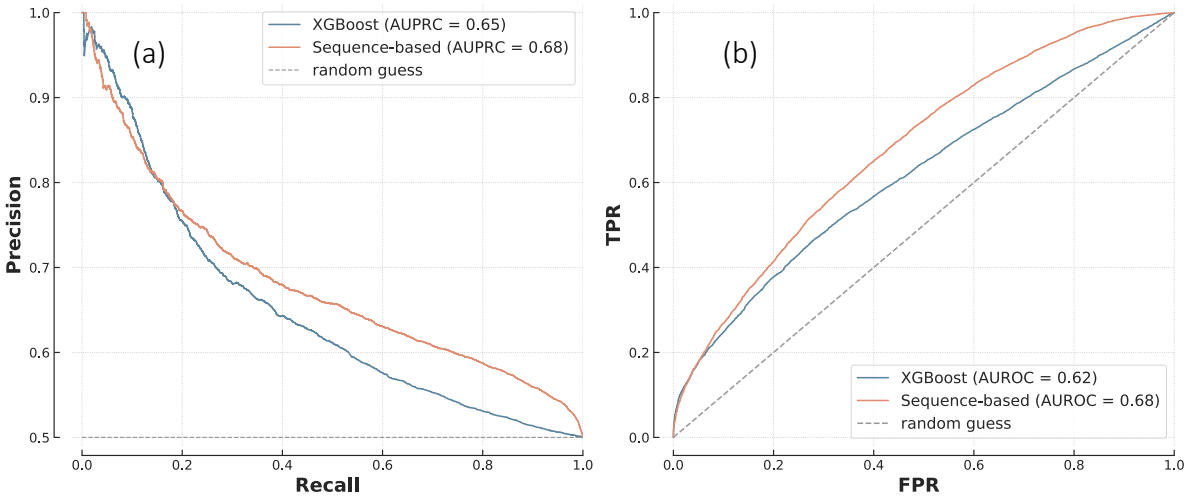
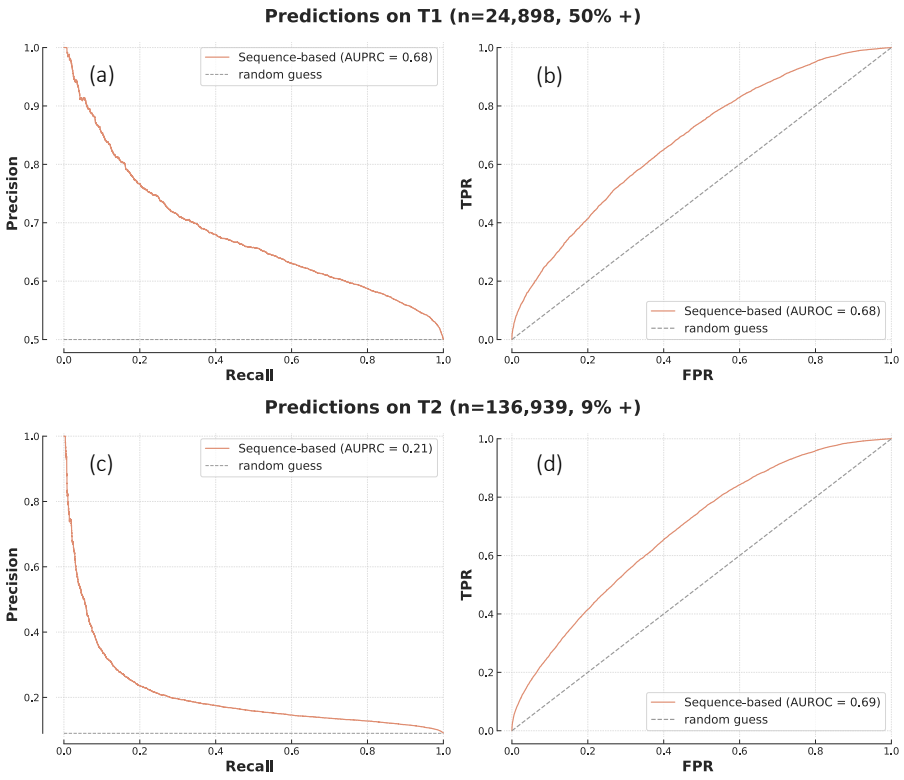


Figure 6: Siamese network vs XGBoost on *T1*.
The difference between the ROC curves (b) was statistically significant (p = 9x10-47).

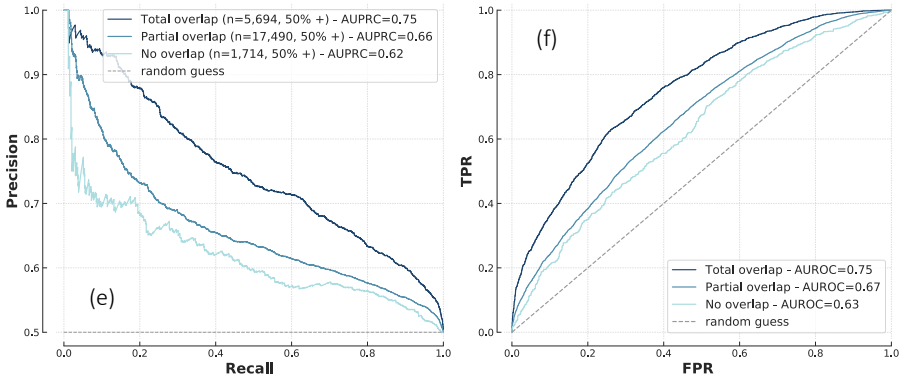
Reporting sheet B4PPI-Human: Sequence-based

PR and ROC curves on T1* and T2**



* First testing set, used to compare models on an independent set and investigate protein-level overlap.
** Second testing set, used to assess generalisation on an imbalanced dataset (10 times more negative examples than positive ones).

Impact of protein-level overlap



	Running time	Memory	Energy used	Carbon footprint (UK)
Training once	1h10	15 GB	0.75 kWh	189 gCO_2e
Training incl. hyperparameters tuning	>100h	>1.5 TB	> 75 kWh	> 18.9 $kgCO_2e$
Inference	1min46	6 GB	0.01 kWh	4 gCO_2e

Number of (trainable) parameters: 1.6m

Figure 7: Performance sheet of the sequence-based model on B4PPI-Human.

Combining FG annotations and deep learning tools

Despite some similarities between the PR curves of XGBoost and the sequence-based model (Figure 6.a), the two tools make different predictions for a large number of protein pairs. For example, for a recall of 17%, both tools have a precision of 80% but make different predictions for 18% of protein pairs. This seems to indicate that the two approaches uncover complementary aspects of PPI predictions. FG-based and sequence-based methods differ in both the input features and the algorithms (traditional machine learning vs deep learning), so I explored different ways to combine them.

Replacing similarity measures with deep learning for FG-based prediction

FG-based models are based on pre-computed similarity measures such as colocalisation and co-expression. Using these similarity measures reduces the dimensionality of the problem to only ~10 features, which enables traditional machine learning algorithms but also hides potential complex biological interactions between the features. As described in Chapter 2, each FG feature was first converted to a long vector (a bag-of-words for example, the lengths of which are in Table 1) which were then compared using cosine similarity.

First, to highlight the need for these similarity measures with traditional machine learning tools, I trained logistic regressions on the long vectors representing the FG features without lowering the dimension. The input is then of size 45,944 (two vectors of length 22,972). I tested an unregularised regression, an L^1 regression (Lasso) [30] and an ElasticNet regression [31], and as shown in Figure 8, such models fail to learn any reliable patterns due to the high dimensionality of the problem.

Considering the successes of deep learning and Siamese architectures for sequence models, I designed a similar architecture that replaces similarity measures with deep learning to better incorporate the richness of GO annotations and expression profiles (Figure 9). The input features were then the long vectors of FG annotations. These vectors were all concatenated to form a long one-dimensional vector of length 22,972. Similarly to protein sequences, these two long vectors (one for each protein) then went through a

Siamese network. Because the data was not sequential anymore, I replaced the GRU in the embedding step with a simpler multi-layer perceptron (**Figure 9** and **Table 3**). As shown on its performance sheet (**Figure 11**), this model, called Deep Learning (DL) FG-based, performed similarly to the sequence-based approach. In particular, protein-level overlap also impacted performance, which seems to indicate that this sensitivity to protein-level overlap is a property of the technology rather than the nature of the input features used. When directly comparing performance curves to the previous models tested (**Figure 10**), I found that the ROC curves of the two deep learning algorithms are almost identical (p=0.1), but the PR curves reveal that the DL FG-based approach doesn't perform as well as the sequence one at low recall/high performance. I therefore went on to explore other ways to combine sequences and FG information.

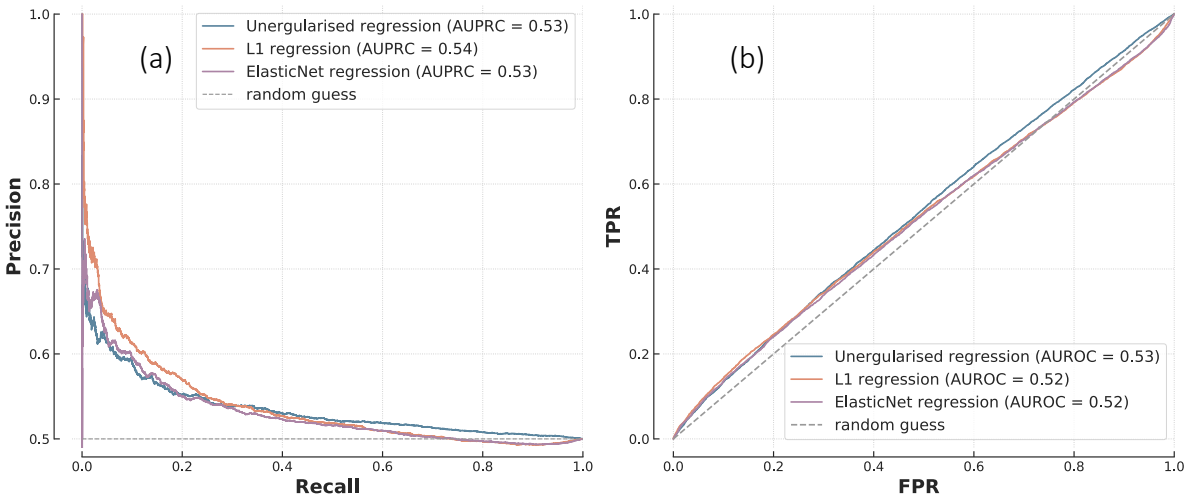


Figure 8: Performance of three logistic regressions using the long FG vectors as input.

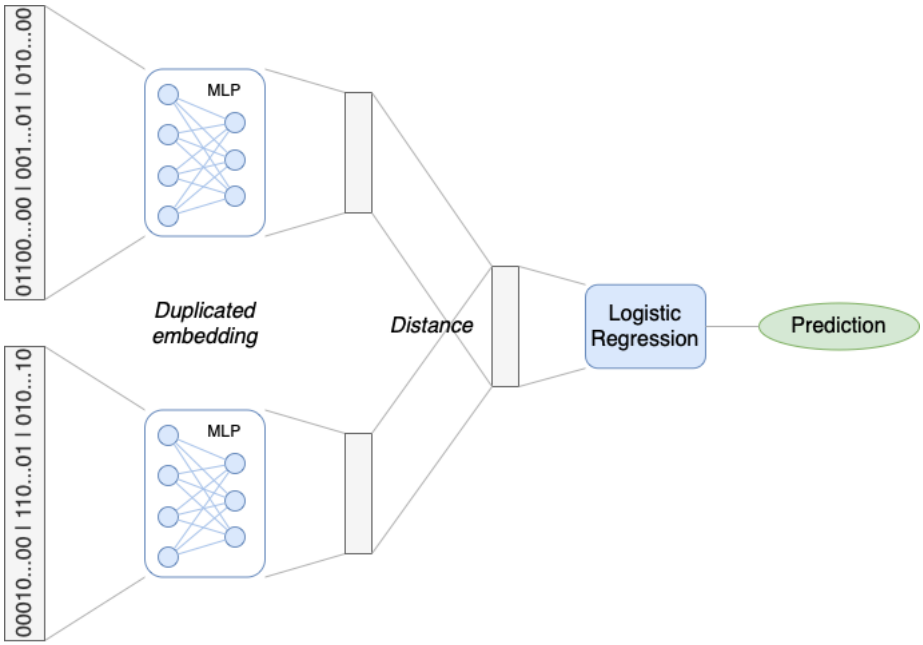


Figure 9: Siamese architecture to predict PPIs from raw FG annotations.
The input is the concatenated sparse vectors representing the FG features and the embedding step is a fully connected multi-layer perceptron.

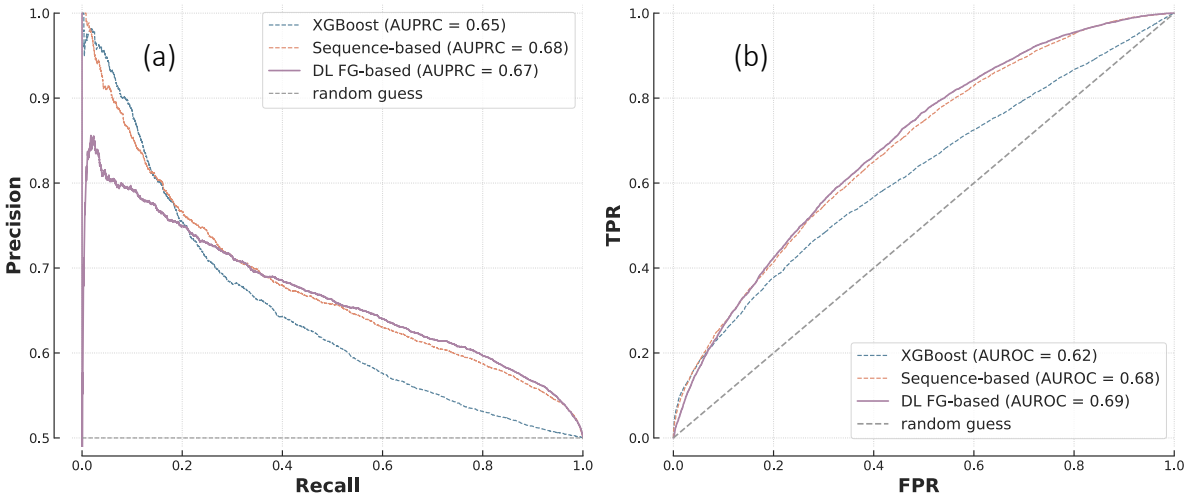
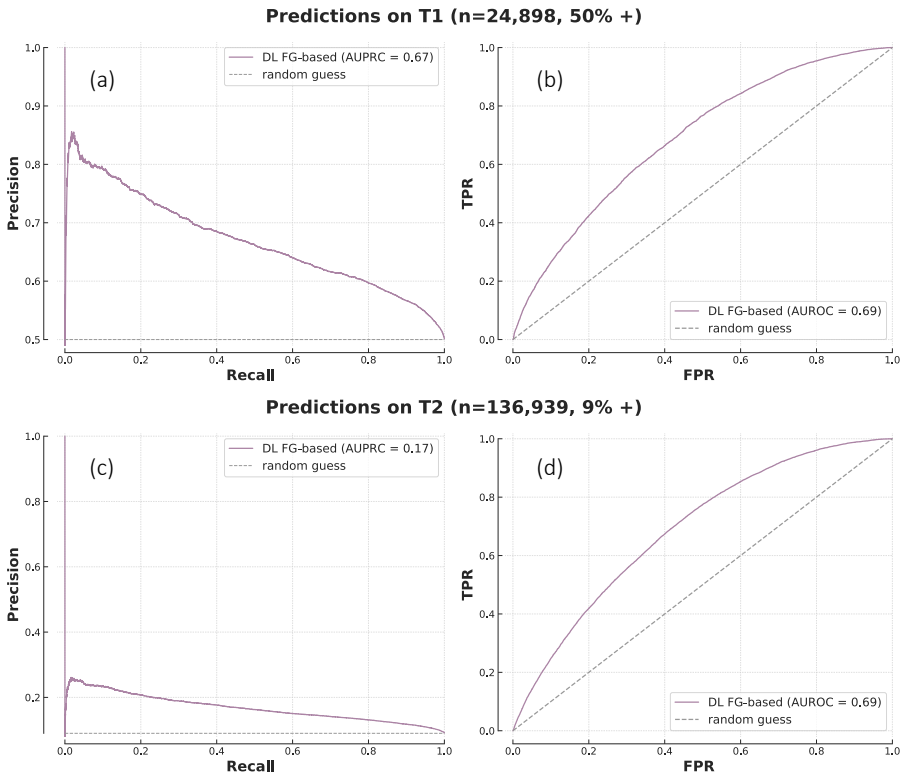


Figure 10: Deep Learning FG-based model vs previous models on *T1*.

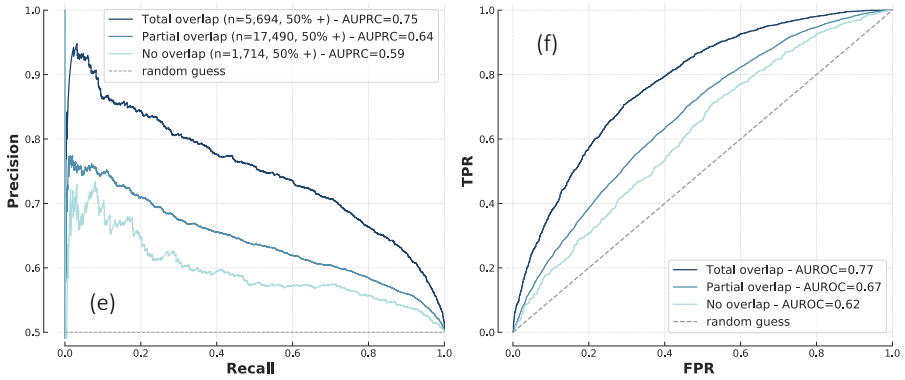
Reporting sheet B4PPI-Human: Deep Learning FG-based

PR and ROC curves on T1* and T2**



* First testing set, used to compare models on an independent set and investigate protein-level overlap.
** Second testing set, used to assess generalisation on an imbalanced dataset (10 times more negative examples than positive ones).

Impact of protein-level overlap



	Running time	Memory	Energy used	Carbon footprint (UK)
Training once	39min	15 GB	0.42 kWh	105 gCO_2e
Training incl. hyperparameters tuning	>100h	>1.5 TB	> 42 kWh	> 10.5 $kgCO_2e$
Inference	12sec	6 GB	0.002 kWh	0.54 gCO_2e

Number of (trainable) parameters: 11.9m

Figure 11: Performance sheet for the deep learning DF-based model on B4PPI-Human.

Combining FG-based and sequence-based approaches into a single hybrid model

The previous attempt at combining FG annotations with deep learning did not lead to a better model but hinted, once again, at some form of complementarity. Indeed, even when the two Siamese models had similar performance, they made different predictions for many pairs (e.g. for 25% of pairs at 30% recall, **Figure 10.a**). To leverage this complementarity, I designed a hybrid model that combines the two Siamese-based models into a single predictor (**Figure 13**). The embedding step processed sequences and FG annotations separately, using the same neural networks as before (GRU for the sequences and MLP for the FG features). The final embedding for a protein was then a concatenation of the two representations created.

Figure 12 shows how this hybrid approach performed compared to previous attempts. Compared to the sequence-based model, it had a slightly higher AUPRC (0.69 vs 0.68) and AUROC (0.70 vs 0.68), with the difference between the ROC curves being just significant (p=0.00047). The curves, however, revealed that although the hybrid approach outperformed the other tools at high recall, it was not as good as XGBoost and the sequence model at high precision. Although promising, this hybrid model did not show the improvement expected and seemed to mainly make predictions based on the amino acid sequences (at 10% recall, only 8% of predictions differed between the sequence and hybrid models).

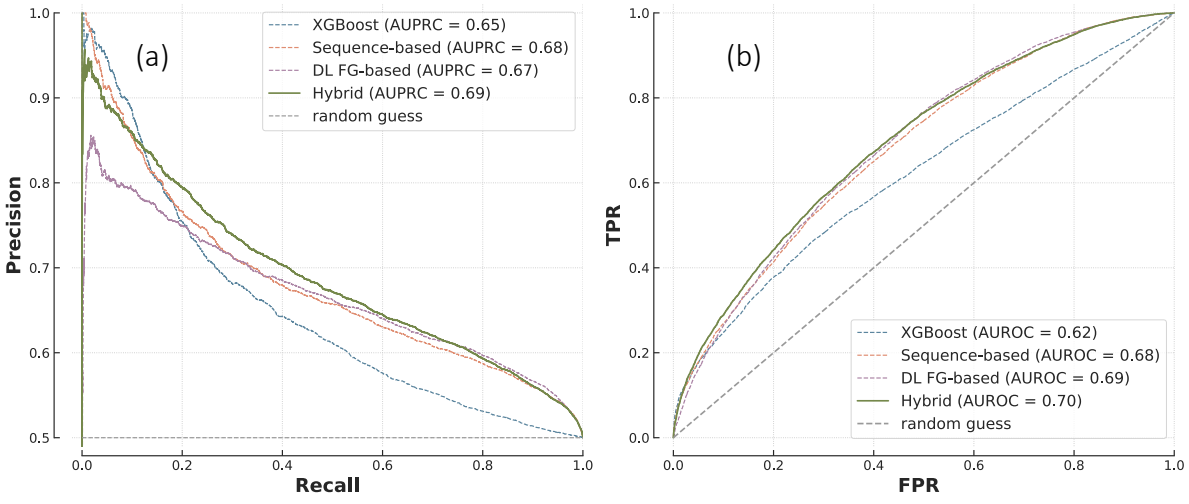


Figure 12: Hybrid model vs previous models on *T1*.

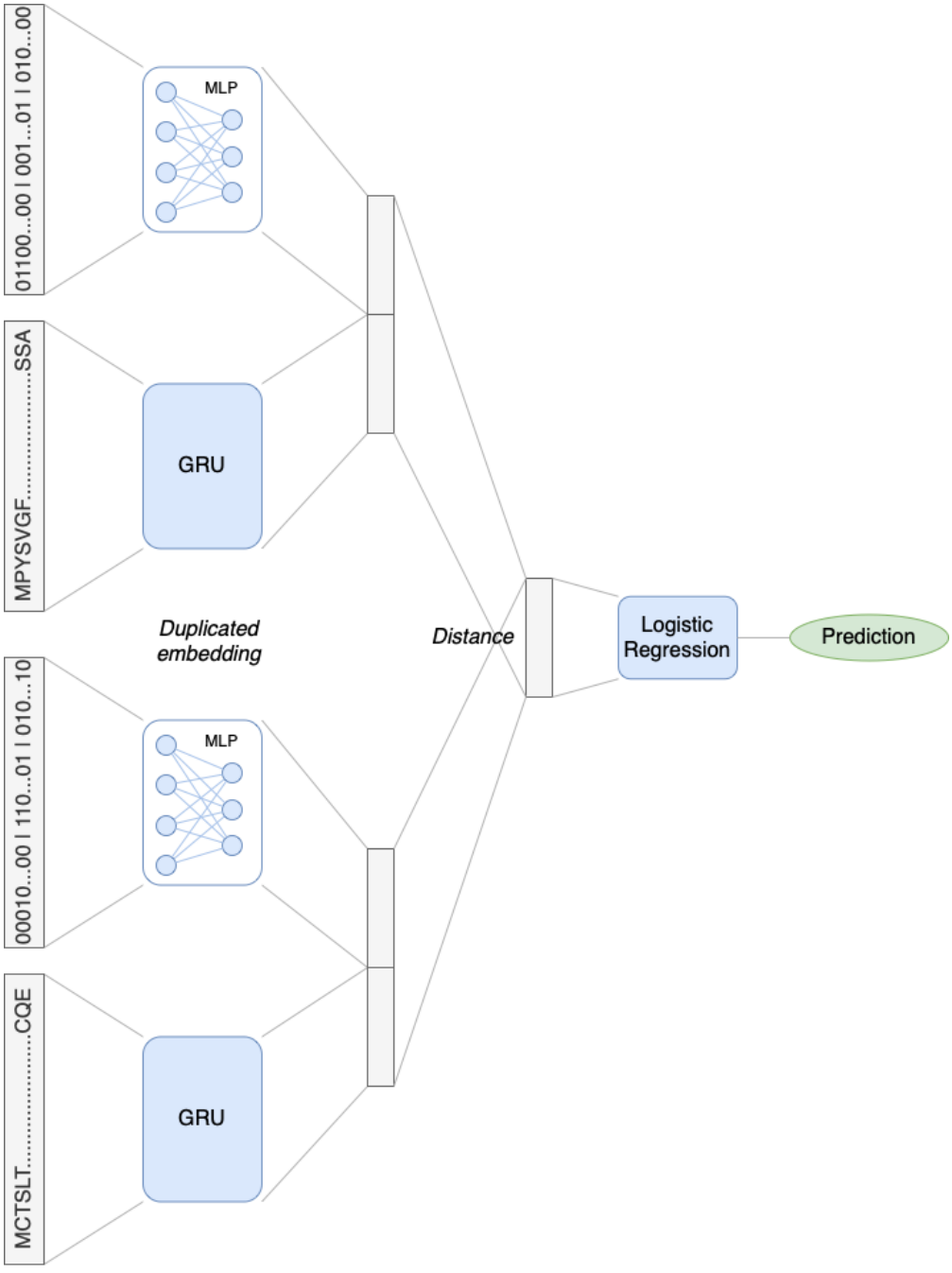


Figure 13: Hybrid architecture combining sequence and FG inputs.

Considering the lack of success with the previous approaches, I also investigated ways to include pre-computed similarity measures (**Figure 14**). I concatenated the output of a Siamese network, either sequence-based (**Figure 5**) or hybrid (**Figure 13**), with the pre-computed similarity measures. Using either a hybrid embedding or a sequence-based one yielded similar performance curves (**Figure 15**, **panels a and b**) (p=0.14), and once again, these new architectures performed similarly to the previous models tested (**Figure 15**, **panels c and d**).

Unfortunately, my different attempts at leveraging the complementarity between sequence-based and FG-based approaches were so far unsuccessful. This is in line with other failed attempts at stacking different predictors together [32]. However, it was not in vain, as it helped me better understand the inference mechanisms at play. The striking similarity between the different Siamese models, using either FG annotations or amino acid sequences, indicated that the technology influenced predictions more than the choice of features. Perhaps the error was to build only one hybrid predictor for all proteins instead of different predictors for different proteins. This is what I investigated next.

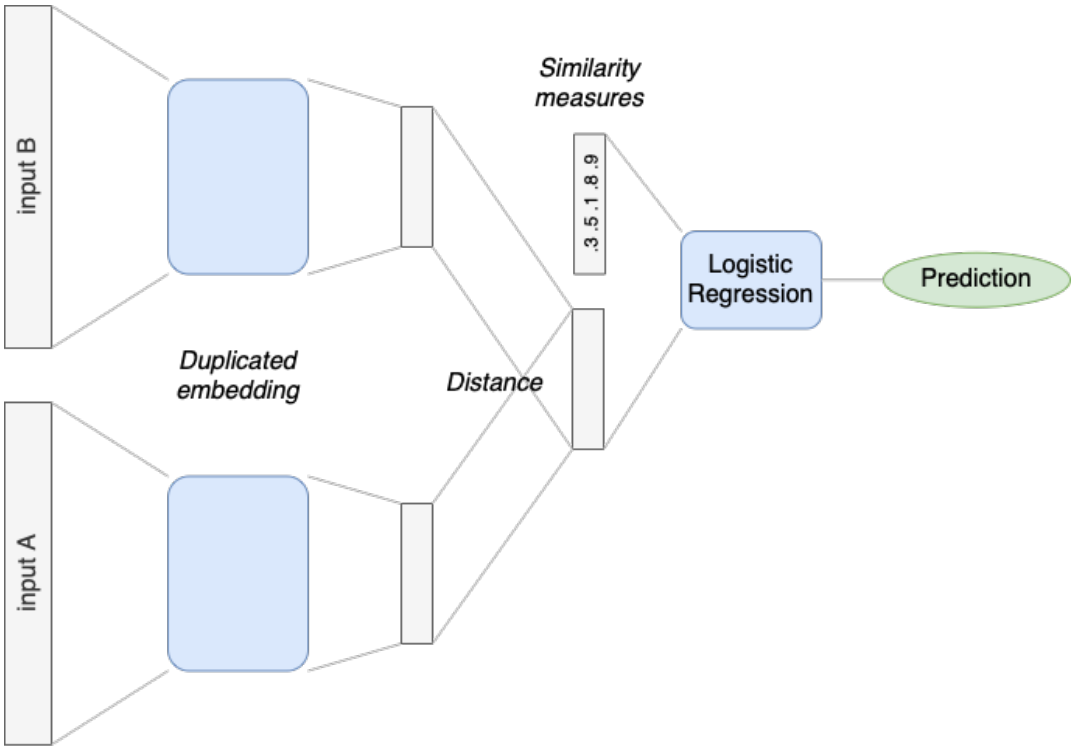


Figure 14: Siamese architecture combined with similarity measures.

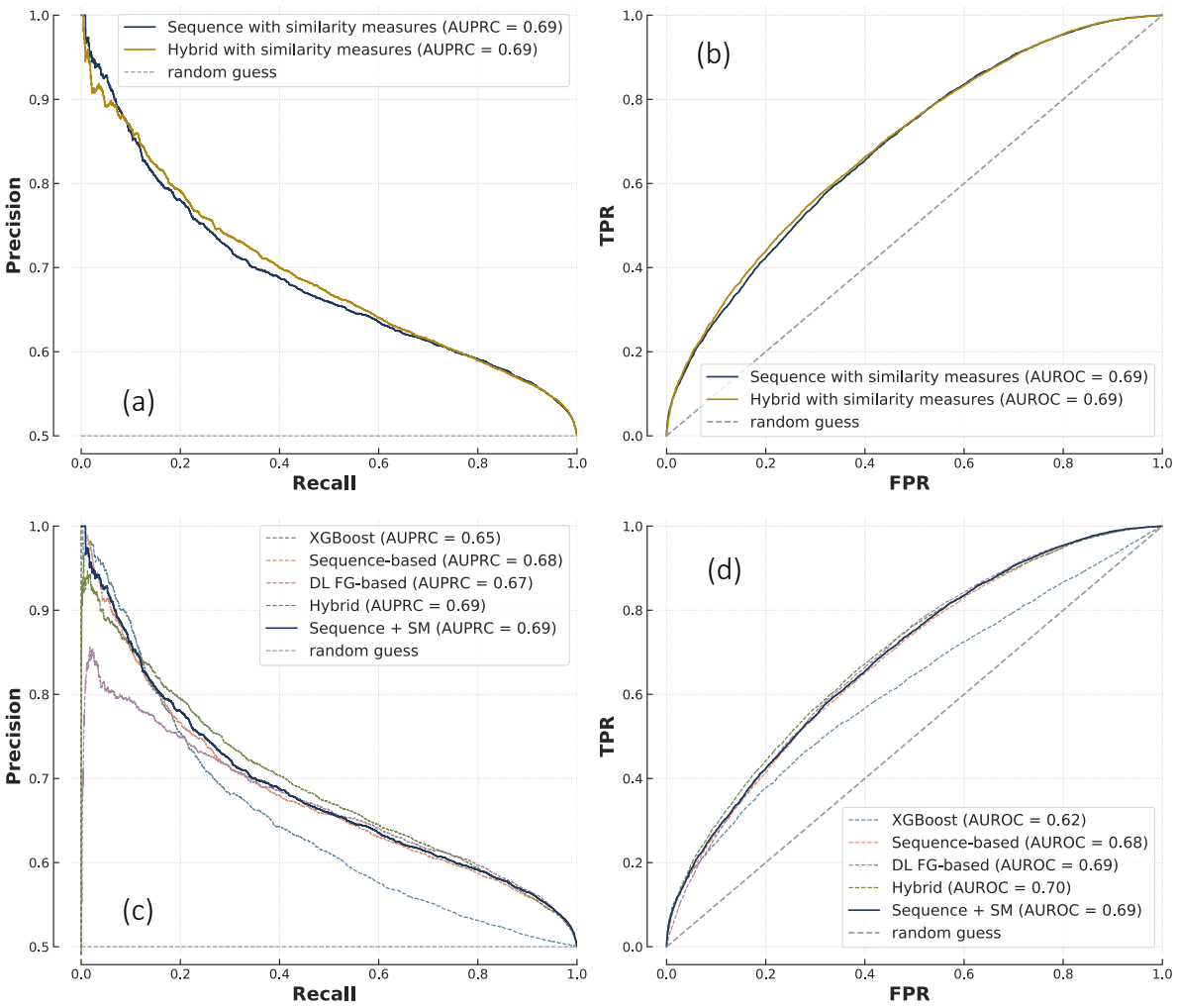


Figure 15: Performance of Siamese architectures including similarity measures. Sequence-based vs hybrid embedding when combined with similarity measures (SM) (a, b) and sequence-based + SM vs previous models (c, d).

Table 3: Optimal hyperparameters for the alternative architectures tested.

Algorithm	Missing data imputation	Scaling	Optimal hyperparameters
FG-based Siamese architecture	N/A	N/A	Batch size = 200, gradient_clip_val = 10, n_hidden_layer = 1, size_hidden_layer = 512, size_embedding = 256, dropout = 0.3, output = linear, learning rate = 0.0001 (MLP) and 0.0001 (output)
High dimension FG regression	N/A	N/A	Batch size = 200, gradient_clip_val = 10, (no regularisation) or (L1 regularisation, λ=1) or (ElasticNet, $\lambda_1 = \lambda_2 = 10$)
Hybrid architecture	N/A	N/A	Batch size = 200, gradient_clip_val = 10, RNN = bidirectional GRU, hidden size = 512 (GRU), n_layers = 1 (GRU), n_hidden_layer = 1 (MLP), size_hidden_layer = 512 (MLP), size_embedding = 256 (MLP), dropout = 0.3 (MLP), output = linear, learning rate = 0.001 (GRU) and 0.0001 (MLP and output)
Hybrid + similarity measures	Yes (mean)	Yes	Same as above, batch size = 150
Sequence + similarity measures	Yes (mean)	Yes	Same as the hybrid architecture

The role of network hubs is essential to PPI prediction

Being scale-free has significant biological consequences. Contrary to a random network (i.e. without hubs), in a scale-free network, the vast majority of proteins have few connections, which brings stability and robustness. Indeed, in case of random failures, the likelihood of a hub failing is very low, and when this happens, remaining hubs guarantee that pathways still exist. However, such networks are sensitive to non-random, targeted attacks on hubs. If several major hubs are removed, the network turns into a collection of unrelated sub-networks. This is why hubs are often linked to diseases, e.g. cancer and *P53* (a hub with 22 known interactions [25]) [33].

I hypothesised that the biological impact of PPI networks' topology translated to PPI prediction and that a one-fits-all approach for hubs and lone proteins was unlikely to be optimal. To investigate this, I studied how prediction models perform on interactions between protein hubs (*hub-hub*), between a protein hub and a lone protein (*hub-lone*) and between lone proteins (*lone-lone*). To ensure sufficient sample size in each subgroup, I built a slightly adjusted dataset, using the same training set as B4PPI-Human, aggregating PPIs from *T1* and *T2*, and using balanced sampling for the non-interacting proteins (Chapter 2). It resulted in 49,796 pairs (50% positive) (Table 4).

Table 4: Sample size in each category in the dataset used to investigate networks topology.

Type of interaction	Number of pairs	% of PPIs
Hub-hub	27,580	50.69 %
Hub-lone	19,205	49.70 %
Lone-lone	3,011	45.67 %

When testing the different models presented above on these three subgroups (hub-hub, hub-lone and lone-lone), I found a distinct pattern whereby FG models (based on similarity measures) had greater AUPRC and AUROC for interactions involving only lone proteins while sequence-based models performed better for hubs (Figure 16). The other deep learning models tested above, such as DL FG-based and hybrid, followed the same pattern

as the sequence-based model, which would indicate that this phenomenon is due to the different technologies (similarity measures associated with traditional machine learning vs deep learning) rather than FG vs sequence features.

These findings can be explained by the pre-processing of similarity measures for FG models. Because of their central role in biological pathways, hubs are highly studied and therefore annotated for many processes and localisations. For example in the training set, hubs have on average 11.6 annotations for biological processes (a significant feature in the logistic regression model discussed above), while non-hubs only have 5.8 (median 6 vs 3). The same phenomenon is observed for cellular compartments (6.2 vs 3.6 annotations on average). Because the similarity measures used by the FG models quantify overlaps in annotations, hubs annotated for many processes provide little information about the probability of interaction, which can explain why FG-based models perform best when hubs are not involved. On the other hand, sequence-based models have millions of parameters which give them the flexibility to recognise individual proteins and learn specific interaction patterns. These patterns do not translate well to unseen proteins, which explains the impact of protein-level overlap, and require a large number of examples to be learned, which is also consistent with the high performance of these models on network hubs, overrepresented in training sets and therefore well captured by the models.

These results provide insight into the strengths of each approach and, importantly, show that a PPI approach should be context-specific, particularly with respect to the network topologies of interest. Indeed, the apparent superiority of the deep learning model shown in Figure 6 is largely due to the composition of *T1*, made up of 70% of hub-hub or hub-lone interactions.

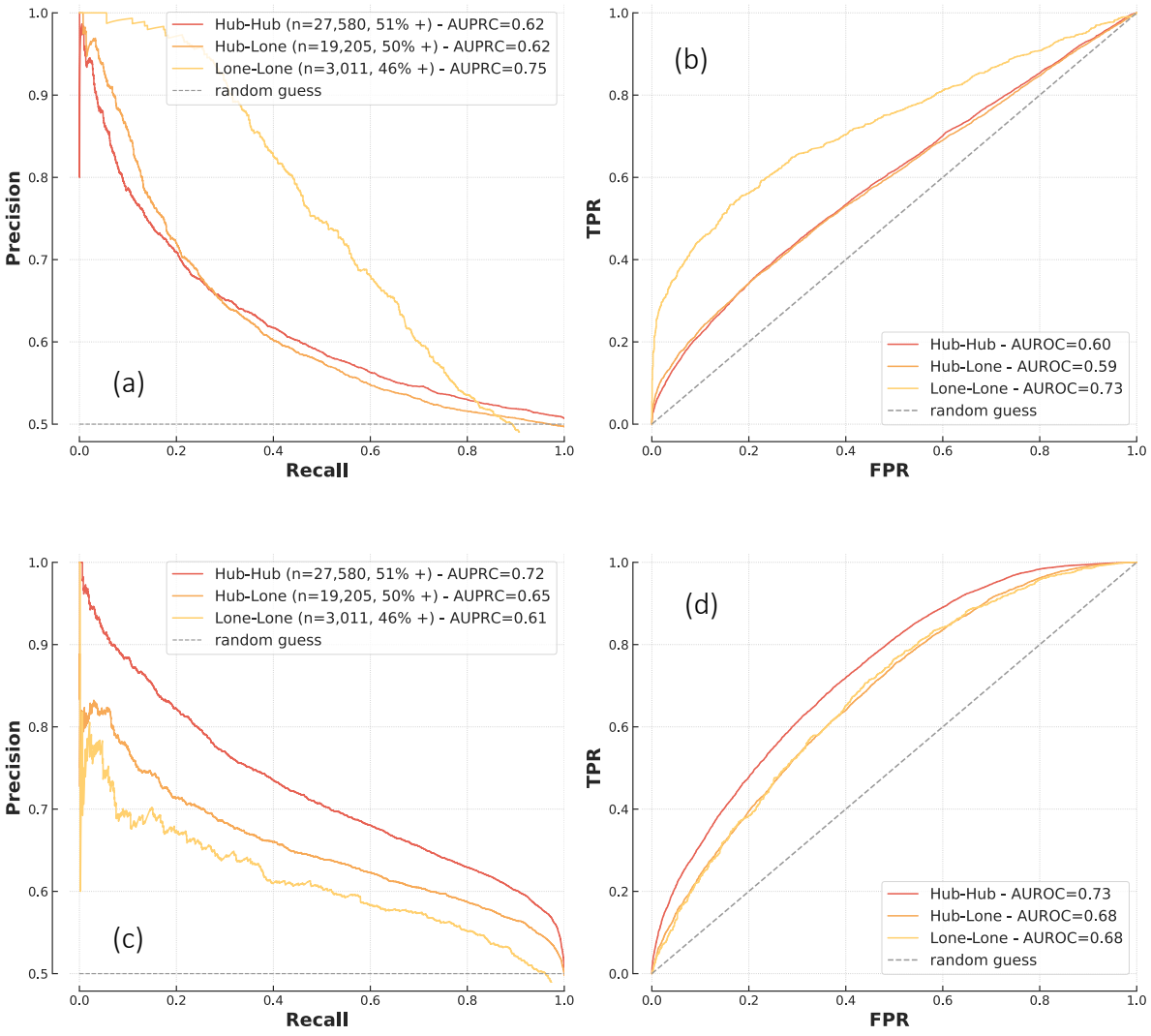


Figure 16: Impact of networks' topology on models' performance.
FG-based XGBoost (a, b) and sequence-based model (c, d) on hub-hub, hub-lone and lone-lone interactions.

Cross-species validation of PPI prediction models and relative performances

S. cerevisiae is a well-studied model organism with a known interactome and has been used extensively for *in silico* PPI predictions [6], [34], [35]. I replicated the analyses presented in the previous sections on *S. cerevisiae* proteins and found that my findings regarding the impact of network topology and models' relative performances were robust across species. The data was selected similarly to previously, extracted and curated from IntAct and UniProt but without data from the Human Protein Atlas (HPA) and Bgee as these databases do not curate yeast (Chapter 2). A range of FG-based and sequence-based models was optimised as before, and the hyperparameters of the final models are in Table 5.

Like previously, all FG-based models had similar performance with AUPRC between 0.71 and 0.73 (Figure 17); however, in this analysis, the differences between XGBoost and other models were statistically significant ($p = 2x10^{-12}$ for Naïve Bayes and $p = 9x10^{-31}$ for logistic regression). The sequence model outperformed FG-based models in most cases ($p = 2x10^{-6}$), except at high recall (Figure 17.a). Second, similar to human models, FG-based yeast models were not sensitive to protein-level overlap, while sequence-based models were, with more overlap yielding better results (Figure 18). Finally, the results related to the role of network hubs were also consistent; FG-based models were better able to predict lone-lone protein interactions, while the sequence-based model was better at predicting interactions amongst protein hubs (Figure 19).

Table 5: Optimal parameters for the models trained on the yeast dataset.

Algorithm	Missing data imputation	Scaling	Optimal hyperparameters
Logistic Regression	Yes (mean)	Yes	Penalty = none, tol = 0.0001
XGBoost	No	No	colsample_bytree = 0.7087, learning_rate = 0.00001129, max_depth = 26, min_child_weight = 3, n_estimators = 244, subsample = 0.9318
Naïve Bayes	Yes (mean)	No	N/A
Sequence-based Siamese architecture	N/A	N/A	Batch size = 200, gradient_clip_val = 10, RNN = bidirectional GRU, output = linear, hidden size = 512, n_layers = 1, learning rate = 0.001 (GRU) and 0.0001 (output)

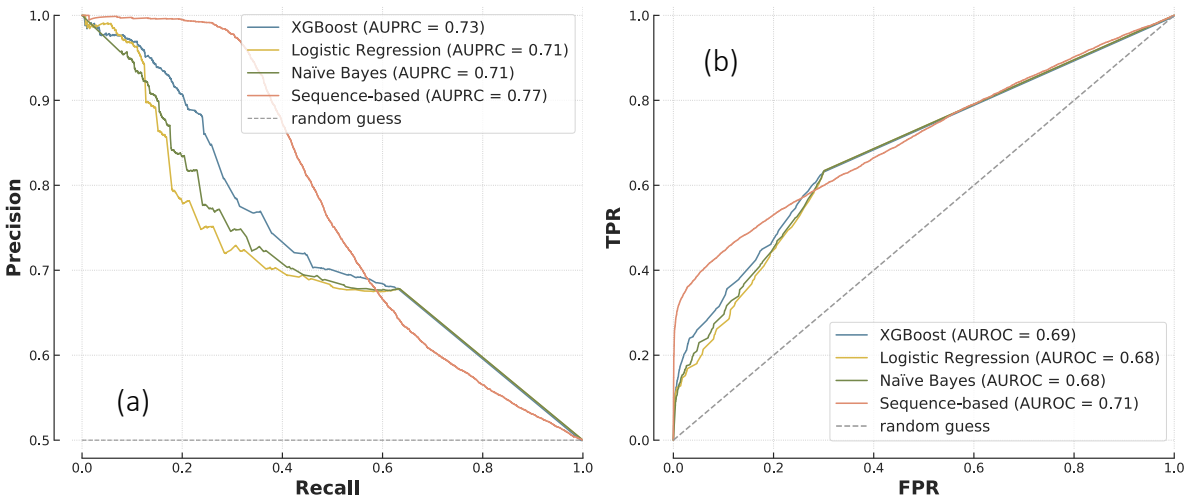


Figure 17: Comparison of a range of models on the yeast testing set.

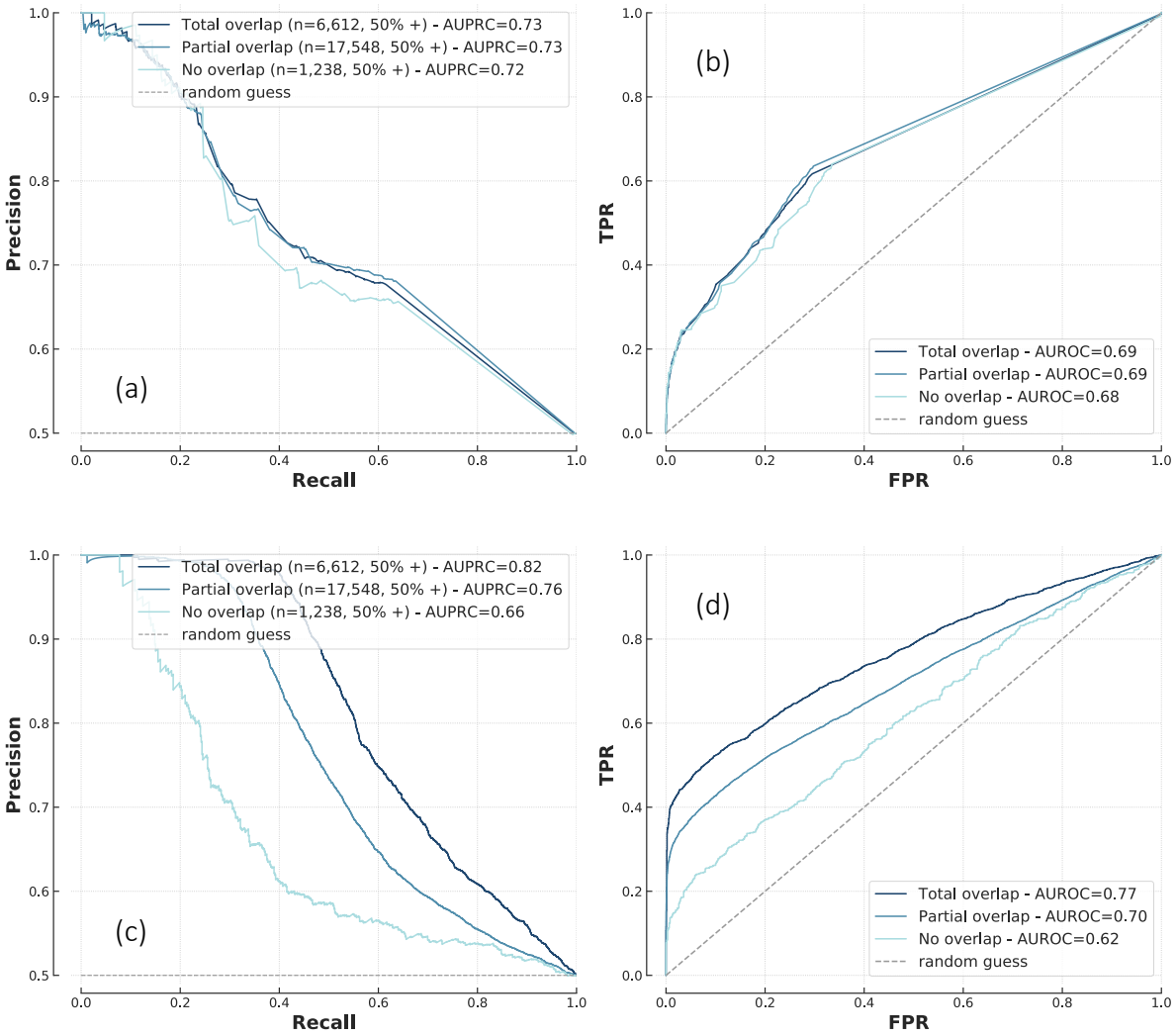


Figure 18: Impact of protein-level overlap on the yeast dataset. XGBoost (a, b) and sequence-based model (c, d).

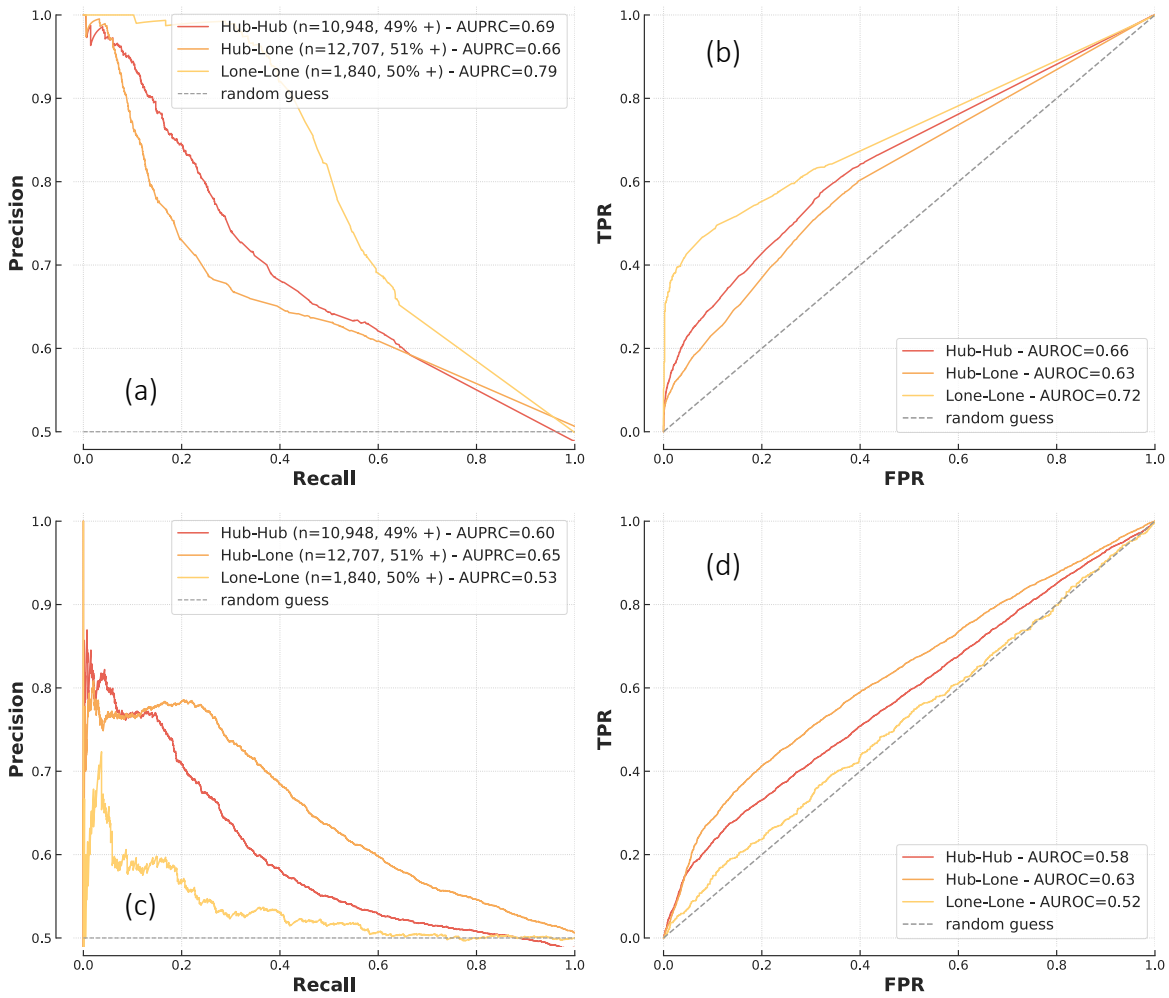


Figure 19: Impact of hubs on the yeast dataset.
XGBoost (a, b, FG-based) and sequence-based model (c, d).

While experimental data on PPIs are readily available for humans and *S. cerevisiae*, many non-model organisms lack data despite their biological relevance [36]. For these, cross-species predictions – i.e. training a model on a species to make predictions on another – are of particular interest. I found that FG-based models are generally more suitable than sequence-based ones for this task.

I first investigated whether models trained on yeast could be used to predict human PPIs, finding that the yeast-trained FG-based models (logistic regression and XGBoost) achieved similar AUPRC and AUROC as human-trained ones to predict human PPIs ($p = 0.26$ for XGBoost) (**Figure 20**). Conversely, the yeast-trained sequence model was unable to predict human PPIs (AUPRC = 0.52 vs 0.68, $p = 3 \times 10^{-272}$). I observed the same phenomenon when using human-trained models to make predictions on yeast (**Figure 21**).

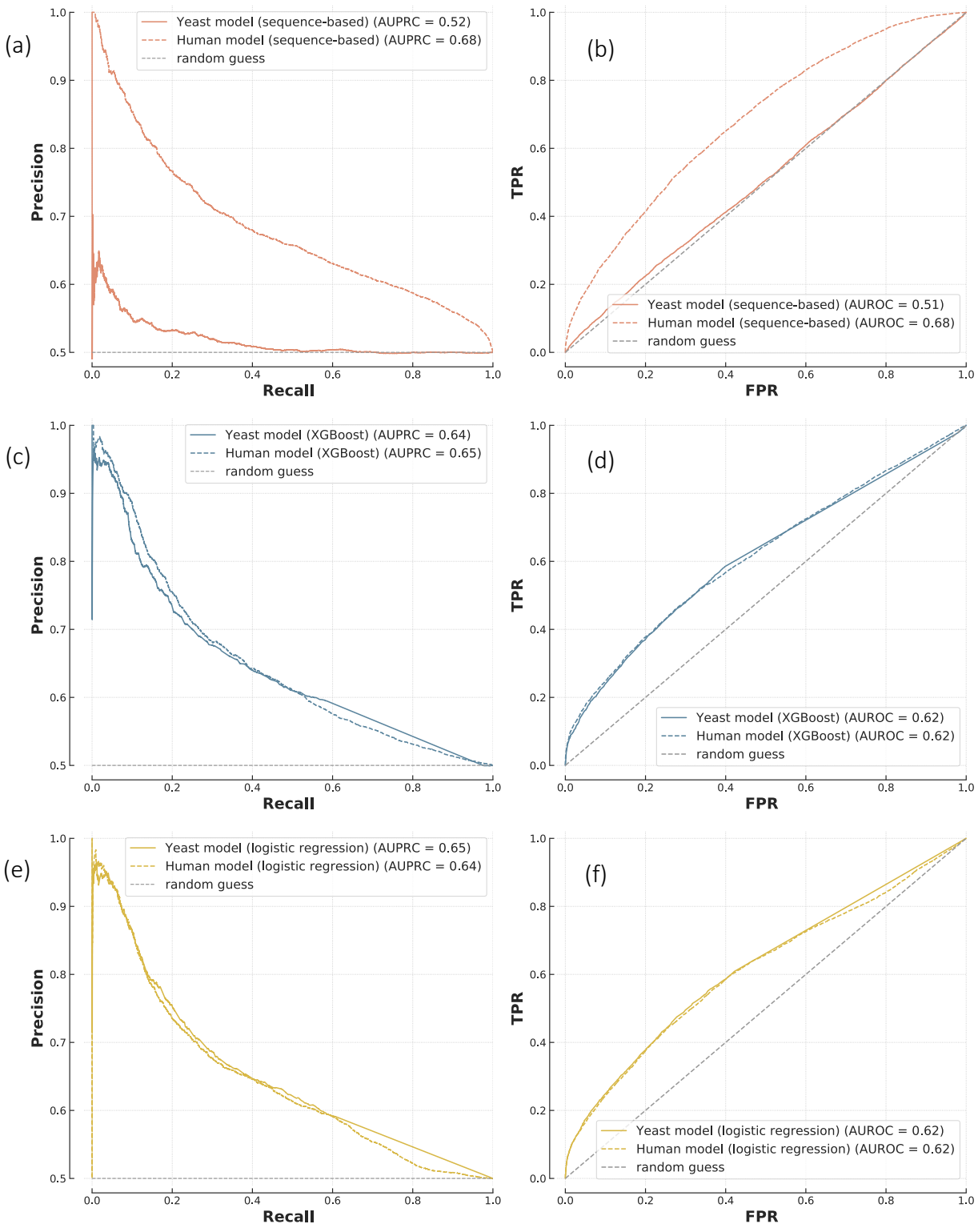


Figure 20: Cross-species predictions (on human data).
Models trained on human PPIs (dotted lines) and yeast PPIs (solid lines) were used to make predictions on the human *T1*. The top plots (a, b) are for the sequence-based model, the other ones are FG-based: XGBoost in the middle (c, d), logistic regression at the bottom (e, f).

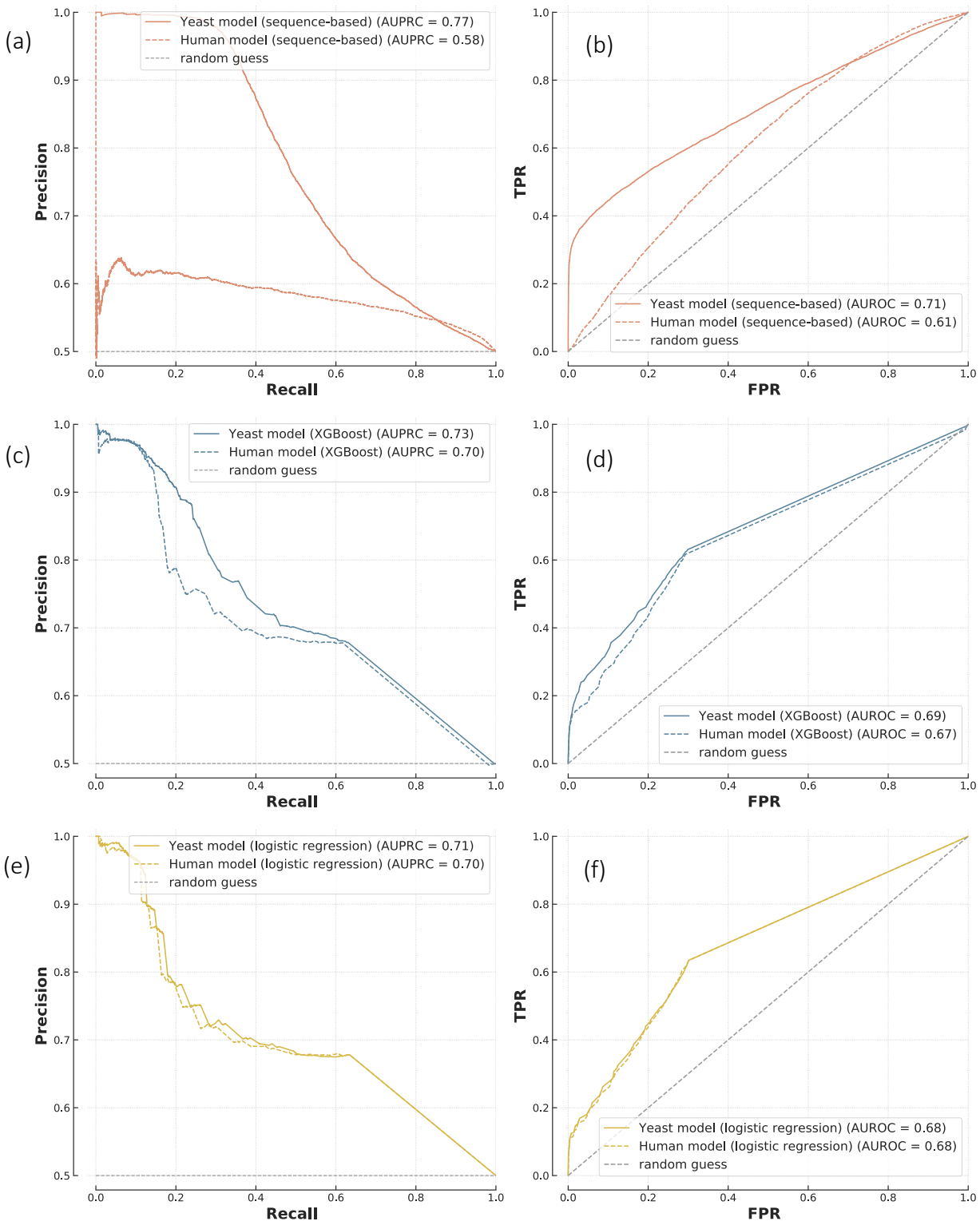


Figure 21: Cross-species predictions (on yeast data).
Models trained on human PPIs (dotted lines) and yeast PPIs (solid lines) were used to make predictions on the yeast testing set (n=25,398, 50% positive). The top plot (a, b) is the sequence-based method, the others the FG-based ones: XGBoost in the middle (c, d), logistic regression at the bottom (e, f).

The results are not sensitive to modelling choices

To test whether modelling assumptions may influence the results presented in this chapter, I tested the impact of some of them and showed that the main conclusions remain. First, I tested whether the models were impacted by the high correlation between the different FG features from the Human Protein Atlas or by the high level of missingness for domain and motif annotations (Chapter 2). I removed the different features involved, and in both cases, the coefficients of the logistic regression remained almost identical to the original ones, resulting in the same predictions.

I then investigated whether the filtering of the human gold standard to PPIs with a MIscore greater than 0.47 had a negative impact. I reran the same analyses, and found that the same conclusions regarding the role of hubs held true (Figure 22). Overall performance was also similar (Figure 23), although interestingly, FG-based models performed slightly better on the unfiltered gold standard (panels a and b), while sequence-based model permed slightly worse (panel c). This can be explained by the fact that the filtering stage mostly removed lone proteins (from 12,268 to 8,842 lone protein and from 3,240 to 3,184 hubs), which are better suited to FG-based models.

I also investigated the impact of the ratio between training and testing sets. As described in Chapter 2, 13% of proteins in the gold standard were set aside for testing, which results in a 69/31 split (close to the 70/30 targeted). Alternatively, setting aside 5% of proteins results in 82% of pairs being used for training while setting aside 20% results in only 58% of pairs being used for training. To study the sensitivity of the results to this choice, I ran the same analyses on three sets: the original one, one based on setting aside 10% of proteins and one with 15%. All results remain the same. Figure 24 shows that both logistic regression (a) and sequence-based (b) models have near-identical performance. The impact of protein-level overlap is also the same (Figure 25), no impact for logistic regression and a significant impact for the sequence-based model.

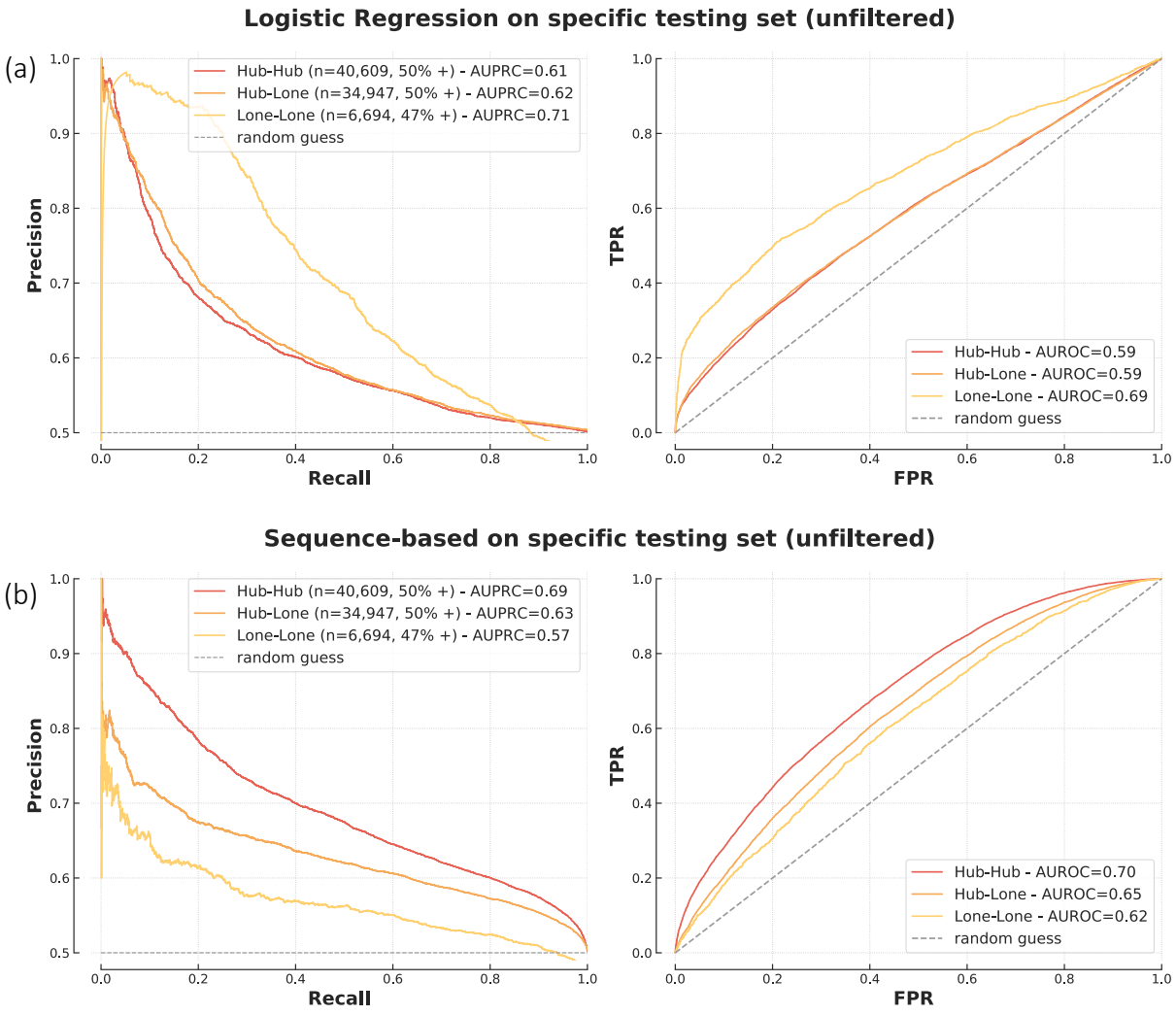


Figure 22: Impact of hubs and lone proteins on performance on the unfiltered dataset (i.e. not using a MIscore threshold of 0.47) for logistic regression (a) and sequence-based model (b).

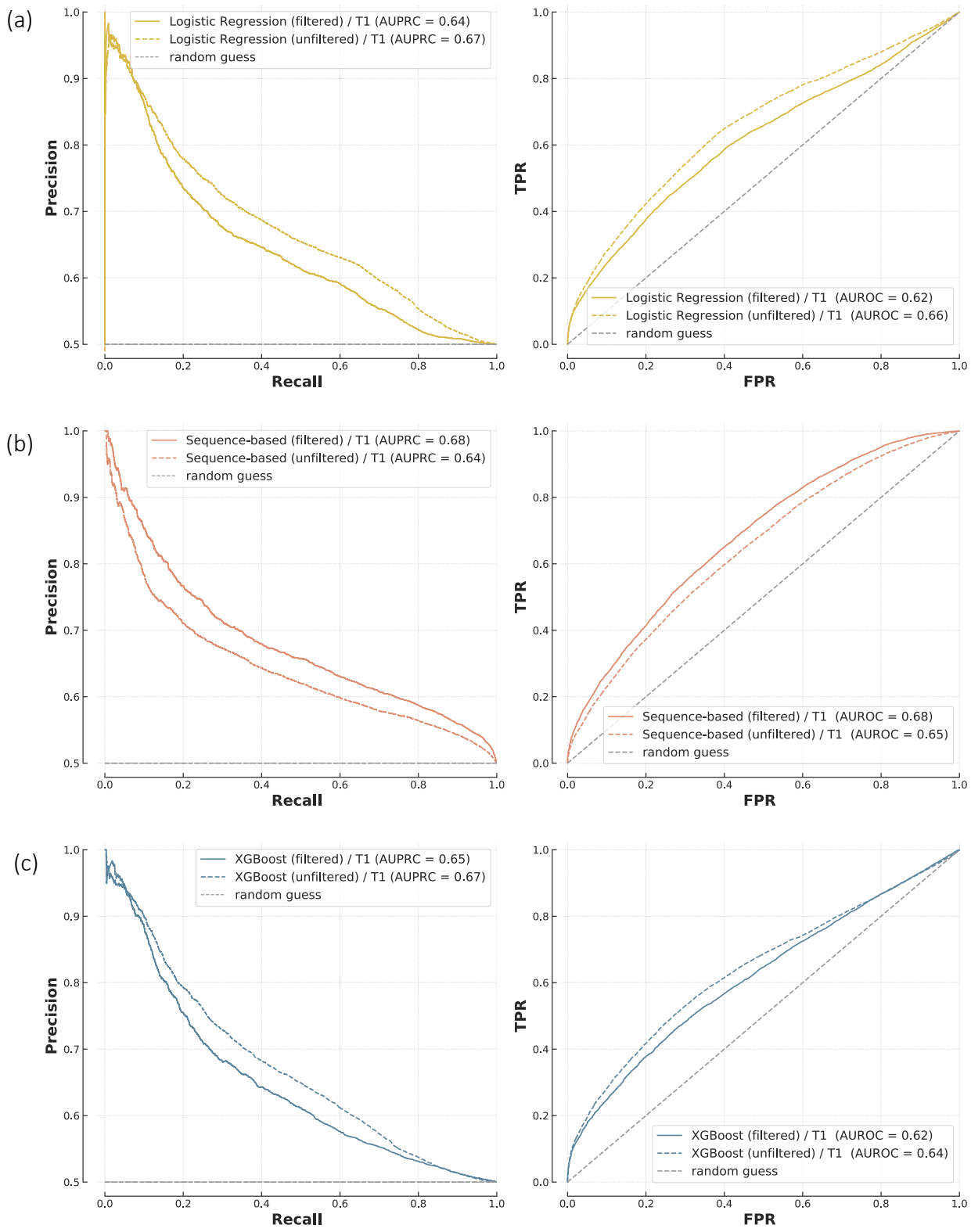


Figure 23: Performance of filtered vs unfiltered gold standard.
Logistic regression (a), XGBoost (b) and sequence-based models (c) were trained and tested on the filtered (solid lines) and unfiltered (dashed lines) gold standards (using a MIscore threshold of 0.47).

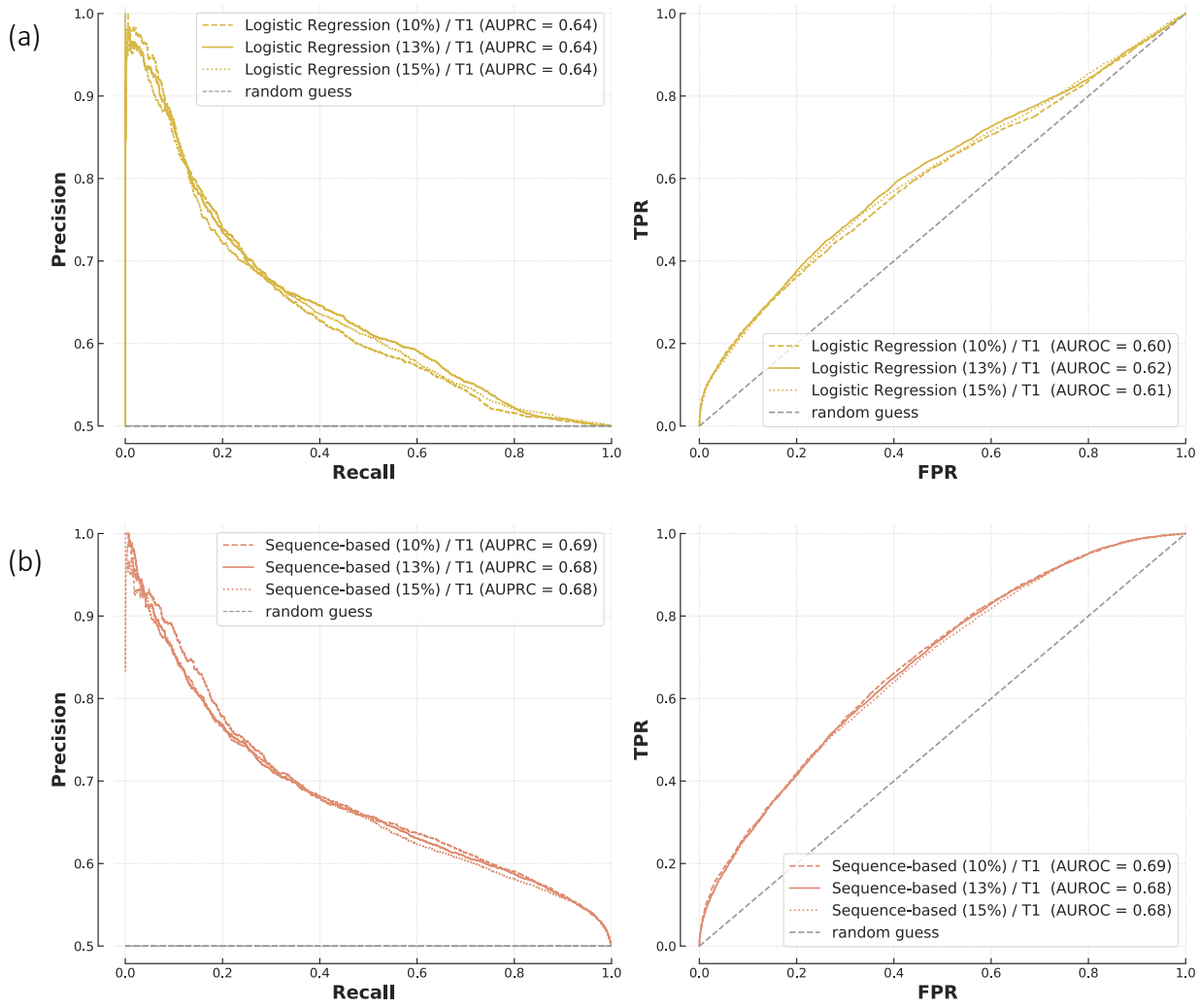
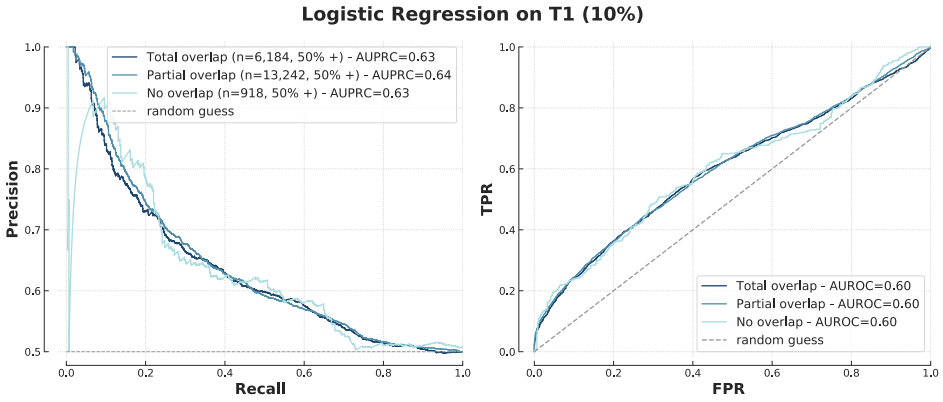
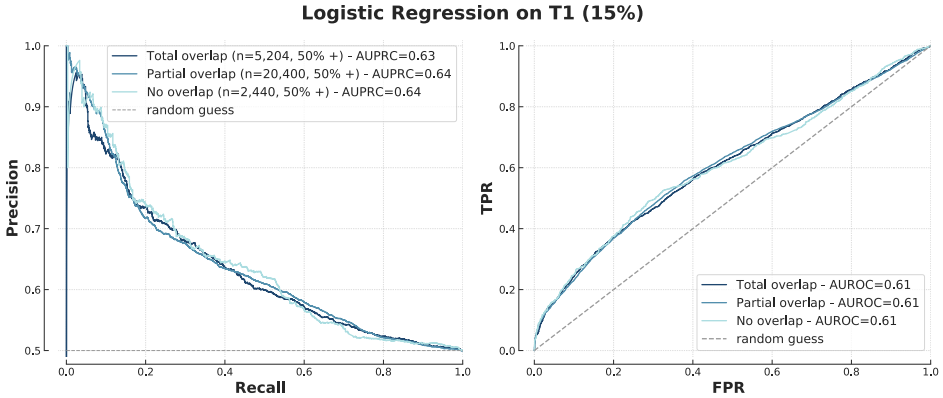


Figure 24: Impact of train/test ratio on performance.

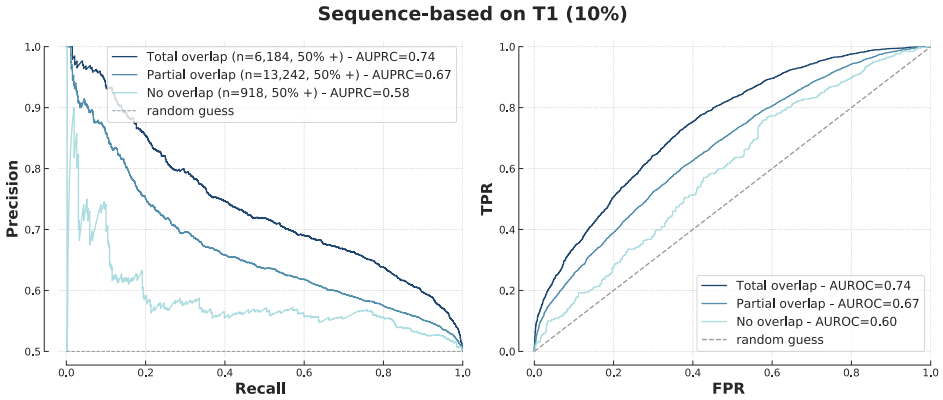
(a)



(b)



(c)



(d)

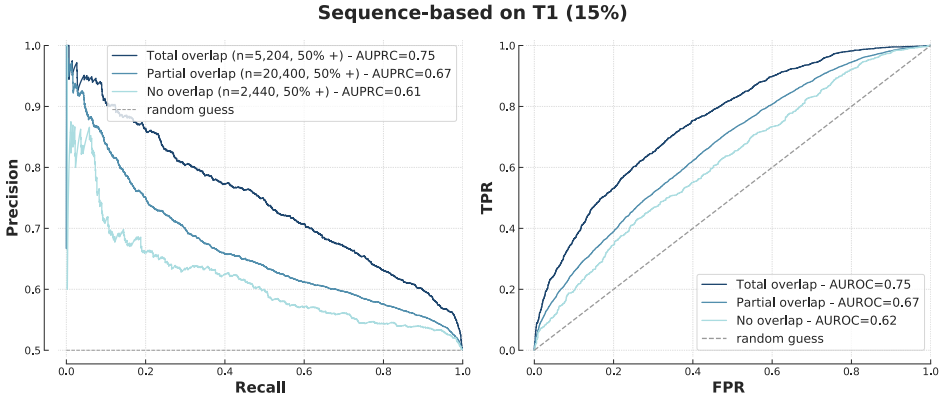


Figure 25: Impact of train/test ratio on overlap analyses for logistic regression (a, b) and sequence-based model (c, d).

Discussion

In this work, I sought to identify and explain the strengths and weaknesses of a wide selection of approaches to PPI prediction and thereby provide the community with a benchmarking resource, insights into which PPI prediction tool to select or trust in a particular scenario, and finally with a set of well-studied PPI models which have also been validated across species.

The FG-based and sequence-based models are common for PPI prediction but are rarely directly compared. In particular, it was unclear where the differences lie and if one approach should be preferred today. I found that when using FG annotations, the choice of the algorithm has little impact on the predictions and a logistic regression performs close to the state-of-the-art while providing clear insight into the decision-making process; here, colocalisation, common biological processes and shared domains are the main indicators of interaction. The success of models based on gene ontology is in line with previous results [37], and so is the importance of colocalisation [38], but interestingly, different analyses tend to highlight different FG features depending on the organism, dataset and predictor (mRNA expression [8], coexpression/colocalisation [38] or biological process [37]). The fact that a highly flexible and non-linear model such as XGBoost performs similarly to logistic regression, making identical predictions in 93% of cases, shows that performance is likely driven by the quality and the pre-processing of the FG annotations instead of the modelling; once the similarity measures have been calculated, there are limited non-linearities and a simple logistic regression achieves top performance. On the other hand, sequence-based models need specifically optimised architectures but achieve similarly high performances, if not higher in some settings, without any biological information apart from amino acid sequences. Although this complementarity between omic-based annotations and sequence or structural information had been reported before, e.g. in [6], combining FG annotations and sequences through hybrid architectures did not improve performance.

I found that the two approaches adapt to the presence of hubs and lone proteins differently and show complementary strengths. While sequence-based models are most useful when hubs are involved, FG-based models perform well for interactions between lone proteins.

It is worth noting that, since then, similar behaviours have also been observed for other approaches to PPI predictions [32]. This simple result offers important insights into the specificities of each approach and explains discrepancies in reported performance in the literature, as the topology of the testing set has a large impact on metrics. These results are not specific to human PPIs, as the same conclusions were drawn from analyses on *S. cerevisiae*. Cross-species predictions are instrumental to study non-model organisms, and I showed that FG-based decision rules translate well to new species while sequence-based ones do not.

These observations are consistent with the way each algorithm learns. FG-based models make predictions based on general but less complex rules about PPIs which translate well to new proteins and species. This is particularly useful considering that many proteins are still not represented in interaction databases. On the other hand, sequence-based models can learn specific inference patterns for each protein. Although this enables such models to make predictions without functional information, it also limits high performance to proteins present in the training set. This likely explains the poor results of sequence models on previously unseen proteins and on cross-species datasets. It also explains the better performance of deep learning models on hubs, overrepresented in training sets, as opposed to lone proteins only present a handful of times, and for which predictions are much more uncertain.

Yet, deep learning models still manage to do better than random guesses even when the proteins are new (i.e. not in the training set), which seems to contradict the previous hypothesis. It can be explained by the sequence similarities between the new proteins and the proteins in the training set; by leveraging common subsequences, the models yield some correct predictions (a form of guilt-by-association).

This study has limitations. I focused on the two most widely used approaches to PPI prediction, namely FG-based and sequence-based; however, some alternative approaches have also been proposed, using, for example, higher-level protein structures [6], phylogeny [39] and the topology of existing networks [40]–[42], but the latter depends heavily on the quality of the existing PPI networks. Most FG annotations are from gene ontologies with a hierarchical structure that I did not account for here, contrary to Armean et al. [37] for

example. I also did not quantify to what extent sequence similarity between training and testing sets influences these results, which could be the focus of future analyses. Moreover, I analysed two common interactomes, human and yeast, yet there are many more. As demonstrated with the yeast dataset, similar benchmarks and analyses can be transferred to other model organisms relatively straightforwardly.

I showed here the limits of classic sequence-based deep learning models for cross-species predictions, but it is worth noting some recent deep learning models that have been successfully used for cross-species predictions [43], [44] by including biological and chemical information about amino acids as well as structural knowledge. The results presented in this work can hopefully guide similar future work and help move this area further.

A better understanding of the inference mechanisms of PPI prediction algorithms is critical to the wider adoption of imputed PPIs for downstream analyses. For this, consistent and reliable evaluation pipelines are necessary, as well as a better understanding of what machine learning models learn. The results presented here make key progress in both areas and facilitate the development, evaluation and reliability of future PPI models.

References

[1] L. Lannelongue and M. Inouye, 'Construction of in silico protein-protein interaction networks across different topologies using machine learning', *bioRxiv*. p. 2022.02.07.479382, Feb. 09, 2022. doi: 10.1101/2022.02.07.479382.

[2] M. E. Maron, 'Automatic Indexing: An Experimental Inquiry', *J. ACM*, vol. 8, no. 3, pp. 404–417, Jul. 1961, doi: 10.1145/321075.321084.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984. [Online]. Available: https://books.google.fr/books?id=JwQx-WOmSyQC

[4] L. Breiman, 'Random Forests', *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[5] R. Jansen, 'A Bayesian Networks Approach for Predicting Protein-Protein Interactions from Genomic Data', *Science*, vol. 302, no. 5644, pp. 449–453, Oct. 2003, doi: 10.1126/science.1087361.

[6] Q. C. Zhang *et al.*, 'Structure-based prediction of protein–protein interactions on a genome-wide scale', *Nature*, vol. 490, no. 7421, pp. 556–560, Oct. 2012, doi: 10.1038/nature11503.

[7] X.-W. Chen and M. Liu, 'Prediction of protein–protein interactions using random decision forest framework', *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, Dec. 2005, doi: 10.1093/bioinformatics/bti721.

[8] L. V. Zhang, S. L. Wong, O. D. King, and F. P. Roth, 'Predicting co-complexed protein pairs using genomic and proteomic data integration', *BMC Bioinformatics*, p. 15, 2004.

[9] C. Cortes and V. Vapnik, 'Support-vector networks', *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[10] E. Fix and J. L. Hodges, 'Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties', *International Statistical Review / Revue Internationale de Statistique*, vol. 57, no. 3, p. 238, Dec. 1989, doi: 10.2307/1403797.

[11] T. Chen and C. Guestrin, 'XGBoost: A Scalable Tree Boosting System', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, San Francisco, California, USA, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[12] A. Ogunleye and Q.-G. Wang, 'XGBoost Model for Chronic Kidney Disease Diagnosis', *IEEE/ACM Trans. Comput. Biol. and Bioinf.*, vol. 17, no. 6, pp. 2131–2140, Nov. 2020, doi: 10.1109/TCBB.2019.2911071.

[13] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[14] L. Biewald, 'Experiment Tracking with Weights and Biases'. 2020. [Online]. Available: https://www.wandb.com/

[15] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson, 'Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach', *Biometrics*, vol. 44, no. 3, p. 837, Sep. 1988, doi: 10.2307/2531595.

[16] J. Neyman and E. S. Pearson, 'On the Use and Interpretation of Certain Test Criteria for Purposes of Statistical Inference: Part I', *Biometrika*, vol. 20A, no. 1/2, pp. 175–240, 1928, doi: 10.2307/2331945.

[17] A. King, 'Cambridge Service for Data-Driven Discovery (CSD3)', May 23, 2018. https://www.hpc.cam.ac.uk/high-performance-computing (accessed Mar. 22, 2022).

[18] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, 'Signature Verification using a "Siamese" Time Delay Neural Network', in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan-Kaufmann, 1994, pp. 737–744. Accessed: Oct. 21, 2019. [Online]. Available: http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network.pdf

[19] S. Chopra, R. Hadsell, and Y. LeCun, 'Learning a Similarity Metric Discriminatively, with Application to Face Verification', in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, vol. 1, pp. 539–546. doi: 10.1109/CVPR.2005.202.

[20] Y. LeCun *et al.*, 'Handwritten Digit Recognition with a Back-Propagation Network', in *Advances in Neural Information Processing Systems*, 1990, vol. 2. Accessed: Feb. 04, 2022. [Online]. Available: https://papers.nips.cc/paper/1989/hash/53c3bce66e43be4f209556518c2fcb54-Abstract.html

[21] S. Hochreiter and J. Schmidhuber, 'Long Short-Term Memory', *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[22] K. Cho *et al.*, 'Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation', *arXiv:1406.1078 [cs, stat]*, Sep. 2014, Accessed: Nov. 29, 2021. [Online]. Available: http://arxiv.org/abs/1406.1078

[23] A. Gupta, A. T. Müller, B. J. H. Huisman, J. A. Fuchs, P. Schneider, and G. Schneider, 'Generative Recurrent Networks for *De Novo* Drug Design', *Mol. Inf.*, vol. 37, no. 1–2, p. 1700111, Jan. 2018, doi: 10.1002/minf.201700111.

[24] H. Li, X.-J. Gong, H. Yu, and C. Zhou, 'Deep Neural Network Based Predictions of Protein Interactions Using Primary Sequences', *Molecules*, vol. 23, no. 8, p. 1923, Aug. 2018, doi: 10.3390/molecules23081923.

[25] S. Orchard *et al.*, 'The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases.', *Nucleic Acids Res*, vol. 42, no. Database issue, pp. D358-63, Jan. 2014, doi: 10.1093/nar/gkt1115.

[26] T. Sun, B. Zhou, L. Lai, and J. Pei, 'Sequence-based prediction of protein protein interaction using a deep-learning algorithm', *BMC Bioinformatics*, vol. 18, no. 1, p. 277, May 2017, doi: 10.1186/s12859-017-1700-2.

[27] H. Zeng, M. D. Edwards, G. Liu, and D. K. Gifford, 'Convolutional neural network architectures for predicting DNA-protein binding', *Bioinformatics*, vol. 32, no. 12, pp. i121–i127, Jun. 2016, doi: 10.1093/bioinformatics/btw255.

[28] W. Falcon and The PyTorch Lightning team, 'PyTorch Lightning'. Mar. 2019. doi: 10.5281/zenodo.3828935.

[29] A. Paszke *et al.*, 'PyTorch: An Imperative Style, High-Performance Deep Learning Library', in *Advances in Neural Information Processing Systems*, 2019, vol. 32. Accessed: Jan. 10, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[30] R. Tibshirani, 'Regression Shrinkage and Selection via the Lasso', *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[31] H. Zou and T. Hastie, 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005, doi: 10.1111/j.1467-9868.2005.00503.x.

[32] X.-W. Wang *et al.*, 'Assessment of community efforts to advance computational prediction of protein-protein interactions', Systems Biology, preprint, Sep. 2021. doi: 10.1101/2021.09.22.461292.

[33] EMBL-EBI, 'Properties of PPINs: scale-free networks | Network analysis of protein interaction data'. https://www.ebi.ac.uk/training/online/courses/network-analysis-of-protein-interaction-data-an-introduction/protein-protein-interaction-networks/properties-of-ppins-scale-free-networks/ (accessed Nov. 29, 2021).

[34] A. Ben-Hur and W. S. Noble, 'Kernel methods for predicting protein-protein interactions', *Bioinformatics*, vol. 21, no. Suppl 1, pp. i38–i46, Jun. 2005, doi: 10.1093/bioinformatics/bti1016.

[35] E. Sprinzak and H. Margalit, 'Correlated sequence-signatures as markers of protein-protein interaction1 1Edited by G. von Heijne', *Journal of Molecular Biology*, vol. 311, no. 4, pp. 681–692, Aug. 2001, doi: 10.1006/jmbi.2001.4920.

[36] J. J. Russell *et al.*, 'Non-model model organisms', *BMC Biol*, vol. 15, no. 1, pp. 55, s12915-017-0391–5, Dec. 2017, doi: 10.1186/s12915-017-0391-5.

[37] I. M. Armean, K. S. Lilley, M. W. B. Trotter, N. C. V. Pilkington, and S. B. Holden, 'Co-complex protein membership evaluation using Maximum Entropy on GO ontology and InterPro annotation', *Bioinformatics*, vol. 34, no. 11, pp. 1884–1892, Jun. 2018, doi: 10.1093/bioinformatics/btx803.

[38] B. S. Srinivasan, A. F. Novak, J. A. Flannick, S. Batzoglou, and H. H. McAdams, 'Integrated Protein Interaction Networks for 11 Microbes', in *Research in Computational Molecular Biology*, 2006, pp. 1–14.

[39] G. Marmier, M. Weigt, and A.-F. Bitbol, 'Phylogenetic correlations can suffice to infer protein partners from sequences', *PLoS Comput Biol*, vol. 15, no. 10, p. e1007179, Oct. 2019, doi: 10.1371/journal.pcbi.1007179.

[40] X. Zhong and J. C. Rajapakse, 'Graph embeddings on gene ontology annotations for protein–protein interaction prediction', *BMC Bioinformatics*, vol. 21, no. Suppl 16, p. 560, Dec. 2020, doi: 10.1186/s12859-020-03816-8.

[41] L. Becchetti, A. Fazzone, and L. Martini, 'Network and Sequence-Based Prediction of Protein-Protein Interactions', *arXiv:2107.03694 [cs, q-bio]*, Jul. 2021, Accessed: Aug. 16, 2021. [Online]. Available: http://arxiv.org/abs/2107.03694

[42] I. A. Kovács *et al.*, 'Network-based prediction of protein interactions', *Nat Commun*, vol. 10, no. 1, p. 1240, Dec. 2019, doi: 10.1038/s41467-019-09177-y.

[43] S. Mahapatra and S. S. Sahu, 'Improved prediction of protein–protein interaction using a hybrid of functional-link Siamese neural network and gradient boosting machines', *Briefings in Bioinformatics*, no. bbab255, Jul. 2021, doi: 10.1093/bib/bbab255.

[44] S. Sledzieski, R. Singh, L. Cowen, and B. Berger, 'D-SCRIPT translates genome to phenome with sequence-based, structure-aware, genome-scale predictions of protein-protein interactions', *cels*, vol. 0, no. 0, Sep. 2021, doi: 10.1016/j.cels.2021.08.010.

CHAPTER 4

THE GREEN ALGORITHMS FRAMEWORK, A TOOL TO PROMOTE MORE SUSTAINABLE COMPUTATIONAL RESEARCH

Contents

Introduction

Advances in computation, including those in hardware, software and algorithms, have enabled scientific research to progress at unprecedented rates. Weather forecasts have increased in accuracy to the point where 5-day forecasts are approximately as accurate as 1-day forecasts 40 years ago [1], physics algorithms have produced the first direct image of a black hole 55 million light-years away [2]–[4], the human genome has been mined to uncover thousands of genetic variants for disease [5], and machine learning permeates many aspects of society, including economic and social interactions [6]–[9]. Yet, the costs associated with large-scale computation are not being fully captured.

As discussed in Chapter 1, data centres have a non-negligible impact on climate change, and the contribution of scientific research to the issue cannot be ignored anymore. However, it remained unclear to what extent everyday scientific computations contributed to the issue. To grasp the scale of the field, it is interesting to look at the American Extreme Science and Engineering Discovery Environment (XSEDE). XSEDE is a way for US institutions to mutualise computing resources in order to share the workload, and they publish regular usage statistics. In 2020, 9 billion compute hours were charged, representing an hourly pace of 1 million compute hours. The scale of this figure, and the breakdown by field (Figure 1), reinforce the idea that all fields of science should be able to acknowledge and then reduce their impact.

Previous studies have made advances in estimating GHG emissions of computation but have limitations that preclude broad applicability. These limitations include the requirement that users self-monitor their power consumption [10] and are restricted with respect to hardware (e.g. GPUs and/or cloud systems [11], [12]), software (e.g. Python package integration [12]), or applications (e.g. machine learning [10]–[12]). The absence of any studies, let alone tools, investigating the carbon footprint of fields of science outside machine learning and astronomy was a surprising research gap, one that needed to be addressed. To facilitate green computing and widespread user uptake, there was a clear and arguably urgent need for a general and easy-to-use methodology for estimating carbon emissions that could be applied to any computational task.

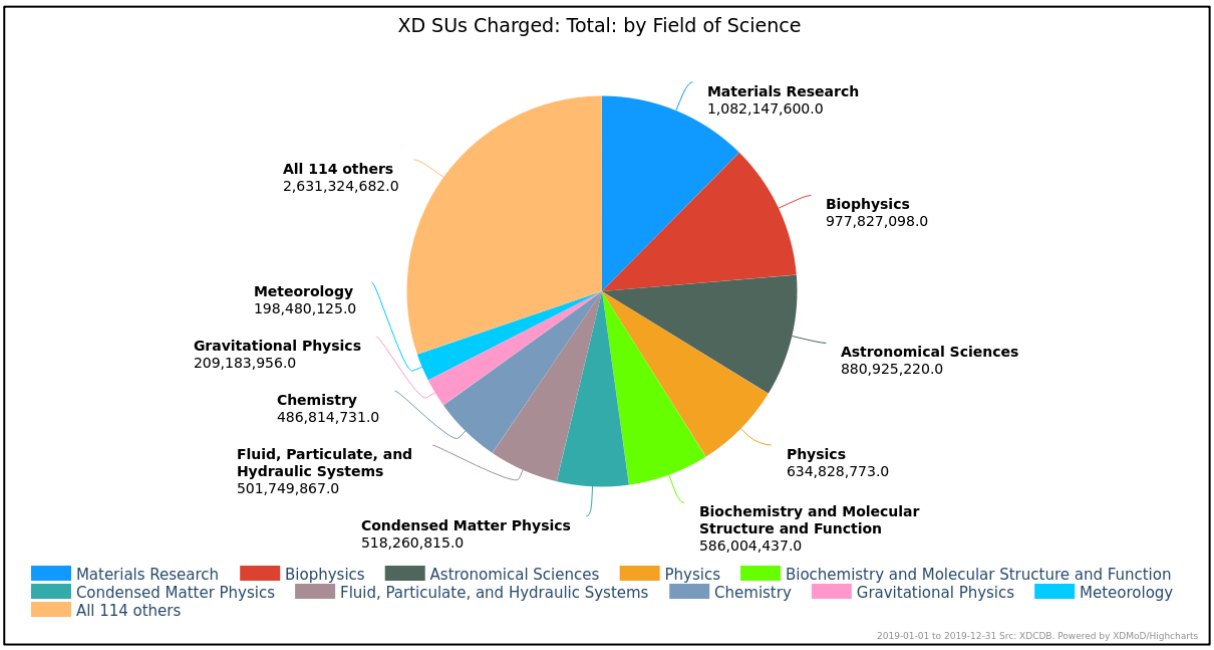


Figure 1: Number of compute hours charged on XSEDE in 2020, broken down by field of science[1].

As we wrote in an early blog on the HDR-UK website (Annexe), in today's data-driven world, a problem does not exist until data about it is collected. The absence of a trustworthy and standardised way to do so, combined with the clear potential benefits for the scientific community, motivated the Green Algorithms project.

For such a framework to be useful, it was important to balance accuracy and practicality. For example, detailed Life Cycle Assessments accounts for the extraction of raw materials, manufacturing, usage and disposal [13], and therefore require extensive investigations into the origin of each component, which is impractical at scale. With this goal of usability in mind, I rapidly thought that an online calculator was needed so that any scientist could easily estimate their impact. Before diving into the method and calculator, I will cover some basic knowledge about the main elements of a computer and a server to better understand how to calculate their energy consumption.

[1] From https://portal.xsede.org/#/gallery/total_su_last_year

Background: how computers work

A desktop or laptop computer is composed of input devices (mouse screen), a display, and a case containing most electronic components. As the focus of this work is on computation, we can focus on the latter. The motherboard is the main circuit board. It controls all the other components which are connected to it. The microprocessor is the “brain” of the computer which performs the computations we need and, as such, consumes the most energy. The most common processor type is a Central Processing Unit (CPU). Although very performant and adapted to a wide range of tasks, CPUs are limited when it comes to handling graphics. For these, Graphics Processing Units (GPUs) have been developed. They were initially for video games, but in recent years, GPUs have proved to also be extremely powerful tools for deep learning. Computers rely on two types of storage, Random-Access Memory (RAM) and hard drives. The RAM, also called memory, is used as a temporary buffer to store data being used in real-time and, as such, is extremely fast but expensive and only present in small quantities (between 8GB and 64GB in modern laptops). For long term storage, mechanical hard drives or solid-state drives (SSD) are used. These have slower read and write speeds but can store larger datasets. Other components include the power supply and cooling device (usually a fan).

Servers located in data centres (for HPC for example) work very similarly. However, since servers are usually stacked together to enable access to arbitrarily large numbers of CPUs and memory, specific processors and RAM are used to enable seamless communication between different servers.

Publications and contributions

Most of the work presented in this chapter has been published. The methodology behind Green Algorithms in *Advanced Science* [14], the Ten Simple Rules in *PLOS Computational Biology* [15] and the carbon footprint of bioinformatics in *Molecular Biology and Evolution* [16].

Lannelongue, Loïc, Jason Grealey, and Michael Inouye. 2021.
'Green Algorithms: Quantifying the Carbon Footprint of Computation'.
Advanced Science 8 (12): 2100707.
https://doi.org/10.1002/advs.202100707.

Lannelongue, Loïc, Jason Grealey, Alex Bateman, and Michael Inouye. 2021.
'Ten Simple Rules to Make Your Computing More Environmentally Sustainable'.
Edited by Russell Schwartz. *PLOS Computational Biology* 17 (9): e1009324.
https://doi.org/10.1371/journal.pcbi.1009324.

Grealey, Jason, Loïc Lannelongue, Woei-Yuh Saw, Jonathan Marten, Guillaume Méric, Sergio Ruiz-Carmona, and Michael Inouye. 2022.
'The Carbon Footprint of Bioinformatics'.
Molecular Biology and Evolution, February, msac034.
https://doi.org/10.1093/molbev/msac034.

Contributions: The Green Algorithms project is a collaboration with Jason Grealey and Prof. Michael Inouye. I led the development of the method framework to estimate carbon footprints and designed and built the online calculator (The Green Algorithms framework: estimating the carbon footprint of scientific computations); I also conducted the investigations into the carbon footprint of physics simulations, weather forecast and natural language processing (Acknowledging the carbon footprint of computational research). Jason Grealey and Michael Inouye provided feedback and suggestions throughout. Jason Grealey coordinated the survey of the carbon footprint of bioinformatics (Deep dive into the carbon footprint of bioinformatics); I contributed to it by studying a set of bioinformatic tools (as specified in the text) and providing feedback on the other analyses. Woei-Yuh Saw, Jonathan Marten, Guillaume Méric, Sergio Ruiz-Carmona provided feedback on their respective fields of expertise, alongside Michael Inouye. I coordinated the Ten Simple Rules article with key contributions from Alex Bateman, Michael Inouye and Jason Grealey.

The Green Algorithms framework: estimating the carbon footprint of scientific computations

Human activities tend to contribute to climate change in a variety of ways, which makes summarising the overall effect extremely difficult. One of the most common ones is the emission of greenhouse gases (GHGs), but here again, each gas has a different impact. The impact of a gas on climate change is measured by its *global warming potential*, which uses the impact of carbon dioxide (CO_2) as a baseline. The values are obtained from the Intergovernmental Panel on Climate Change's (IPCC) reports [17], and as shown in Table 1, they depend on the time horizon considered. In this work, I followed IPCC 's and governments' guidelines and used a 100-year horizon (GWP100) [17]–[19]. For example, over the course of 100 years, 1 kg of methane is thought to have the same impact on global warming as 28 kg of carbon dioxide. To summarise all these effects, *CO_2-equivalent* (CO_2e) represents the amount of CO_2 with the same global warming impact as a mix of GHGs.

There are several competing definitions of *carbon footprint*, and in this project, I used the extended definition from Wright et al. [20]. They defined it as the climate impact of GHGs emitted in a determined timeframe, here running a set of computations, and measured in terms of carbon dioxide equivalent (CO_2e). The GHGs considered are carbon dioxide (CO_2), methane (CH_4) and nitrous oxide (N_2O) [18]; these are the three most common GHGs of the "Kyoto basket" defined in the Kyoto Protocol [21] and represent 97.9% of global GHG emissions [22].

The carbon footprint of an algorithm depends on two factors: the energy needed to run it and the pollutants emitted when producing such energy. The former depends on the computing resources used (e.g. number of cores, runtime, data centre efficiency), while the latter, called carbon intensity, depends on the location and production methods used (e.g. nuclear, gas or coal).

Table 1: Global Warming potential for the three most common greenhouse gases of the Kyoto basket.

Gas	Global Warming Potential for a given time horizon		
	20 years	100 years	500 years
Carbon dioxide (CO_2)	1	1	1
Methane (CH_4)	84	28	7.6
Nitrous oxide (N_2O)	264	265	153

The carbon footprint C (in gCO_2e) of producing a quantity of energy E (in kWh) from sources with a carbon intensity CI (in gCO_2e/kWh) is then:

$$C = E \times CI \quad (1)$$

When estimating these parameters, accuracy and feasibility must be balanced. The study presented here focuses on a methodology that the community can easily and broadly adopt and, therefore, restricts the scope of the environmental impact considered to GHGs emitted to power computing equipment for a specific task. The framework presented requires no extra computation nor involves invasive monitoring tools.

Quantifying energy consumption

To measure the power consumption of a computer, a range of components need to be accounted for. Ideally, every single part of the server would be considered, from the CPU to the indicator lights. However, that would make it extremely impractical for scientists to do it and it was important to identify how power was distributed between components.

I modelled an algorithm's energy[2] needs as a function of the runtime, the number, type and process time of computing cores (CPU or GPU), the amount of memory mobilised and the power draw of these resources. The model further includes the efficiency of the data centre[3], i.e. how much extra power is necessary to run the facility (e.g. cooling and lighting).

Similar to previous works [10], [11], this estimate is based on the power drawn by processors and memory, and the efficiency of the data centre. However, I refined the formula and added flexibility by including a unitary power draw (per core and per GB of memory) and the processor's usage factor. The energy consumption E (in kilowatt-hours, kWh) is expressed as:

$$E = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times 0.001 \quad (2)$$

where t is the runtime (hours), n_c the number of cores and n_m the size of memory available (gigabytes). u_c is the core usage factor (between 0 and 1). P_c is the power draw of a

[2] Power (in watt, W) measures the instantaneous draw of a component. Energy (in kilowatt-hours, kWh) measures the power draw over time and is obtained by multiplying the power draw by the runtime.

[3] By data centre, I mean the facility hosting the cores and memory, which may not be a dedicated data centre.

computing core and P_m the power draw of the memory (Watt). PUE is the efficiency coefficient of the data centre. The assumptions made regarding the different components are discussed below.

The processor

Unfortunately, manufacturers do not communicate the power usage of processing cores (which shows that there is still a long road ahead for green computing). The metric commonly used to report the power draw of a processor, either CPU or GPU, is its thermal design power (TDP, in Watt) provided by the manufacturer. While TDP is not a direct measure of power consumption, rather the amount of heat a cooling system should be able to dissipate during regular use - it is commonly considered a reasonable approximation.

Modern CPUs are composed of several processing cores that can perform different operations in parallel. Some algorithms will only use one core for successive operations, while others will parallelise computations and reduce runtime by distributing the workload. To account for both cases, I use a value of TDP-per-core P_c – the TDP divided by the number of cores – in the estimations.

The energy used by the processor is the power draw multiplied by the runtime, scaled by the usage factor. Indeed, over the total runtime of an algorithm, a processor can be hanging at different times — while data is being loaded or written out, for example — which leads to a real usage time smaller than the total runtime. Although workload managers on HPC platforms can provide this information, it cannot be known *a priori* and, on some systems, tracking can be impractical at scale. Modelling the exact processing time of past projects may also necessitate re-running jobs, which would generate unnecessary GHG emissions. Therefore, when this usage factor is unknown, I make the simplifying assumption that core usage is 100% of runtime ($u_c = 1$ in **(2)**).

Memory

Memory power draw is often deemed negligible, especially when compared to power-hungry GPUs [11]. However, many fields rely on large datasets and tools with large memory requirements; in computational biology for example, it is not rare to need over 100GB of

memory for a task. Memory power draw is mainly due to background consumption with a negligible contribution from the workload and database size [23]. Moreover, the power draw is mainly affected by the total memory allocated, not by the actual size of the database used [23]. This is because the load is shared between all memory slots to prioritise access times, but it also keeps every slot in a power-hungry active state. Therefore, the primary factor influencing power draw from memory is the quantity of memory mobilised, which simply requires an estimation of the power draw per gigabyte. Measured experimentally, this has been estimated to be 0.3725 W/GB [23], [24].

For example, requesting 29 GB of memory draws 10.8 W, the same as one core of a popular Core-i5 CPU. Figure 2 further compares the power draw of memory to a range of popular CPUs.

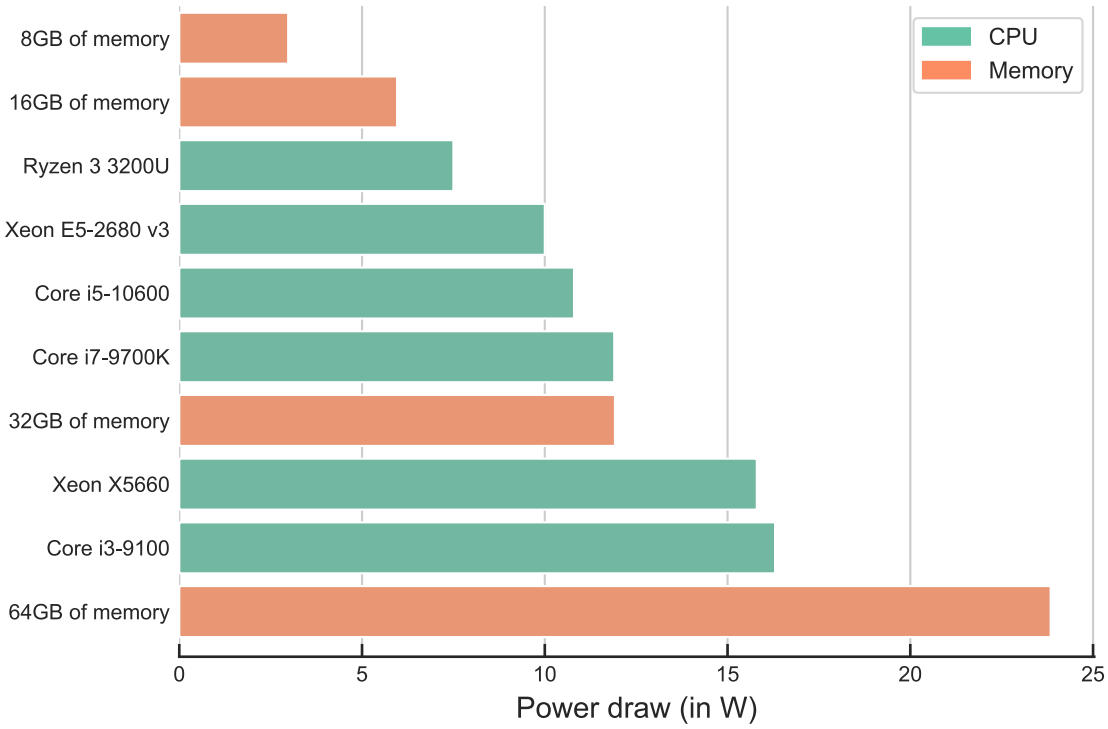


Figure 2: Power draw comparison between memory and a selection of popular processing cores.

Storage

The power draw of storage equipment (HDD or SSD) varies significantly with workload [25] and, as a result, is more difficult to estimate than for memory. To obtain some values, I used three different approaches.

First, I estimated the power needed based on a 2016 prediction that, by 2020, the average capacity of a mechanical hard drive in a data centre would be around 10TB and 5TB for an SSD [26]. The same work estimated the power draw of these disks to be between 6 W/disk (SSD) and 6.5 W/disk (HDD), which results in between 0.65 W/TB (HDD) and 1.2 W/TB (SSD). These results are in line with experimentally measured values of 5.75W/disk for idle HDD (8.75W when active) and 3W for idle SSDs (13W when active) [25]. These also show that the power draw of storage can vary significantly, including with the workload [25], but the order of magnitude remains around 1W/TB or 0.001 W/GB. At the scale of one computation, which is what I consider here, this is two orders of magnitude smaller than memory (0.3725 W/GB) and a CPU core (~10W). While the researcher overhead for approximating storage usage may not be substantial, it is unlikely to significantly affect overall power usage (and GHG emissions) estimations. Therefore, I do not consider the power consumption of storage in this work. Besides, storage is typically solicited far less than memory and is mainly used as a more permanent record of the data, independently of the task at hand.

Yet, the carbon footprint of storage over long periods should not be ignored. Using the power usage estimated above, I calculated that over a year, this represented an energy need of 8.76 kWh/TB, responsible for 4.16 $kgCO_2e$/TB/year (using the world average carbon intensity). An alternative approach was to estimate the cradle-to-grave impact from the Life Cycle Assessment done by some manufacturers, such as Seagate. They estimated that in 2019, the carbon footprint of their hard drives was around 6 $kgCO_2e$/TB/year [27]. Finally, a study from Microsoft estimated the cradle-to-grave impact of average mechanical hard drives to be higher, around 42 $kgCO_2e$/TB/year [28]. This shows the variations in estimates between studies, but the order of magnitude of ~10 $kgCO_2e$/TB/year can be remembered.

Motherboard

The motherboard distributes power to the various components and enables communication between them. In a desktop computer, its power draw can be substantial, between 25 and 40W for regular motherboards and 45 to 80W for gaming computers. However, it also handles a wide variety of tasks (multiple softwares open, graphical display, internet connections etc.), many of which are unrelated to the computations of interest. In the case of servers, motherboards don't have to handle as many tasks (no graphical displays for example) and therefore require less power, only ~5% of the server's energy usage [29]. Because of the uncertainty and the added overhead for researchers to find the model of their motherboard, I decided to not include it in the calculation.

Efficiency of the facility: cooling and other overheads

Data centre's energy consumption includes additional factors, such as server cooling systems, power delivery components and lighting. These overheads represent on average 40% of the electricity bill [30], but efficient facilities can reduce this share to 10%. The efficiency of a given data centre can be measured by the Power Usage Effectiveness (PUE) [31], [32], defined as the ratio between the total power drawn by the facility and the power used by IT equipment:

$$PUE = \frac{P_{total}}{P_{compute}} \quad (3)$$

A PUE of 1.0 represents an ideal situation where all the power supplied to the building is used by computing equipment. The global average of data centres was estimated as 1.67 in 2019 [30]. While data centres with relatively inefficient PUE may not report it as such, some data centres and companies have invested significant resources to bring their PUEs as close to 1.0 as possible using hardware optimisation and data science; for example, Google leverages machine learning to reduce its global yearly average PUE to 1.10 [33], [34].

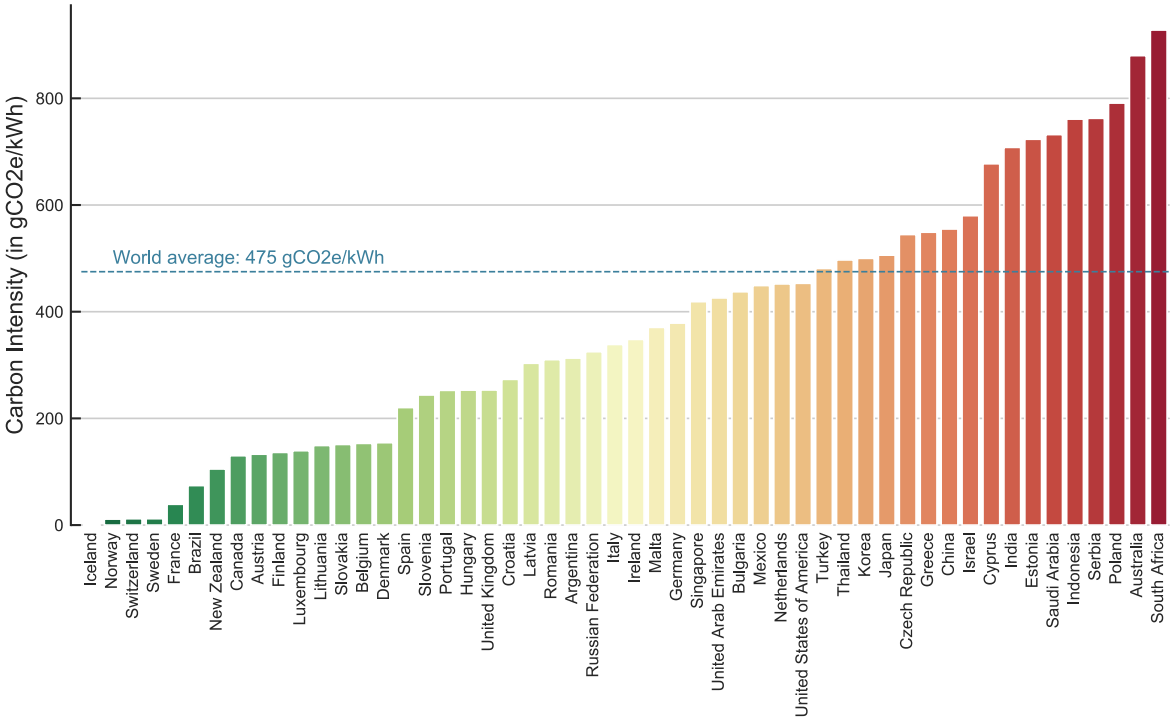


Figure 3: Carbon intensity by country.

Carbon intensity of energy production

For a given country and energy mix, the carbon footprint of producing 1 kWh of energy is called carbon intensity and is also measured in CO_2e to simplify the comparison between different electricity production methods. The carbon intensity varies significantly between locations due to the broad range of production methods (**Figure 3**), e.g. from 12 gCO_2e/kWh in Switzerland (mainly powered by hydro) to 880 gCO_2e/kWh in Australia (mainly powered by coal and gas) [35], [36]. Carbon intensity can also vary within a country, from 7.38 gCO2e/kWh in Vermont-USA to 983.3 gCO2e/kWh in Wyoming-USA. I used the 2020 carbon intensity values aggregated by *Carbon Footprint* [36]. These production factors take into account the GHG emissions at the power plants (power generation) and, when available, the footprint of distributing energy to the data centre.

Estimating the final carbon footprint

By putting together equations **(1)** and **(2)**, I obtained the long-form equation of the carbon footprint C (with the same notations as before):

$$C = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times CI \times 0.001 \qquad (4)$$

The abstract nature of "10 kg of CO2e" makes it difficult to fully comprehend the scale of the carbon footprints discussed, hence the importance of finding estimates of the impact of common activities for context. For example, it has been recently estimated that streaming Netflix emits 55 gCO_2e/hour [37]. It is also useful to compare carbon footprints to the target of 2.3 tons of CO2e per year and per person by 2030 [38]. Two other common comparisons are travel and carbon sequestration by trees.

CO_2e of traveling by car, air or train.

Previous studies have estimated the emissions of the average passenger car in Europe as 175 gCO_2e/km [18], [39] and 251 gCO_2e/km in the United States [40]. The emissions of flying on a jet aircraft in economy class have been estimated between 139 and 244 gCO_2e/km/person, depending on the length of the flight [18]. The train remains the most sustainable option, with carbon footprints ranging from 5 to 37 gCO_2e/km/person [18].

For the online calculator, in addition to car travel, I chose three reference flights: Paris to London (50,000 gCO_2e), New York to San Francisco (570,000 gCO_2e) and New York to Melbourne (2,310,000 gCO_2e) [41].

CO_2 sequestration by trees

Trees play a major role in carbon sequestration, and although not all GHGs emitted can be sequestered, CO_2 represents 74.4% of these emissions [42]. To provide a metric of reversion for CO_2e, I defined a new metric called *tree-months* measuring the number of months a mature tree needs to sequester a given quantity of CO_2. While the amount of CO_2 absorbed by a tree per unit of time depends on several factors, such as its species, size or environment, it has been estimated that a mature tree sequesters, on average, approximately 11 kg of CO_2 per year [43], giving the multiplier in tree-months a value close to 1kg of CO_2 per month (0.92g).

Pragmatic scaling factor

Many analyses are presented as a single run of a particular algorithm or software tool; however, computations are rarely performed only once. Algorithms are run multiple times, sometimes hundreds, systematically or manually, with different parameterisations.

Statistical models may include any number of combinations of covariates and fitting procedures, which should be included in the carbon footprint. To take into account the number of times a computation is performed in practice, the pragmatic scaling factor (PSF) is defined as a scaling factor by which the estimated GHG emissions are multiplied.

The value and causes of the PSF vary greatly between tasks. In machine learning, tuning the hyperparameters of a model requires hundreds, if not thousands, of runs [10], while other tools require less tuning and can sometimes be run a smaller number of times. As per published work or the user's own experience, the PSF should be estimated for any specific task; however, in Green Algorithms, I provide for, and recommend that, each user estimate their own PSF.

The Green Algorithms calculator

As discussed in the introduction, there was a real need for an easy tool that any scientists could use to estimate the carbon footprint of their work. I decided on an online calculator to remove the barrier of having to install a software – notwithstanding the compatibility issues that arise then. It was also critical for it to be open source[4] so that the results could be trusted.

Plotly Dash [44] was chosen to code the platform as it provided a good balance between flexibility (the underlying HTML and CSS code can be customised) and ease of use (most of the code is in Python). The app is hosted on Heroku [45] and can be viewed here:

www.green-algorithms.org

The app presents like most online calculators (**Figure 4** and **Figure 5**). Users can input information about the hardware they use, runtime, memory usage etc., and can see what the resulting energy usage and carbon footprint are, alongside additional metrics such as carbon sequestration by trees and travel by car or plane.

The app was initially released in March 2020 (v1.0) in a form very similar to the current version. I then worked on the v2.0 to add important functionalities, such as the option of

[4] All the code and data are available on GitHub: https://github.com/GreenAlgorithms/green-algorithms-tool

simultaneously inputting CPU and GPU usage, a way to share results via a link and a better integration of the major cloud providers. The latest version, v2.1, included minor UX (user experience) optimisations performed by an external freelance developer[5,6].

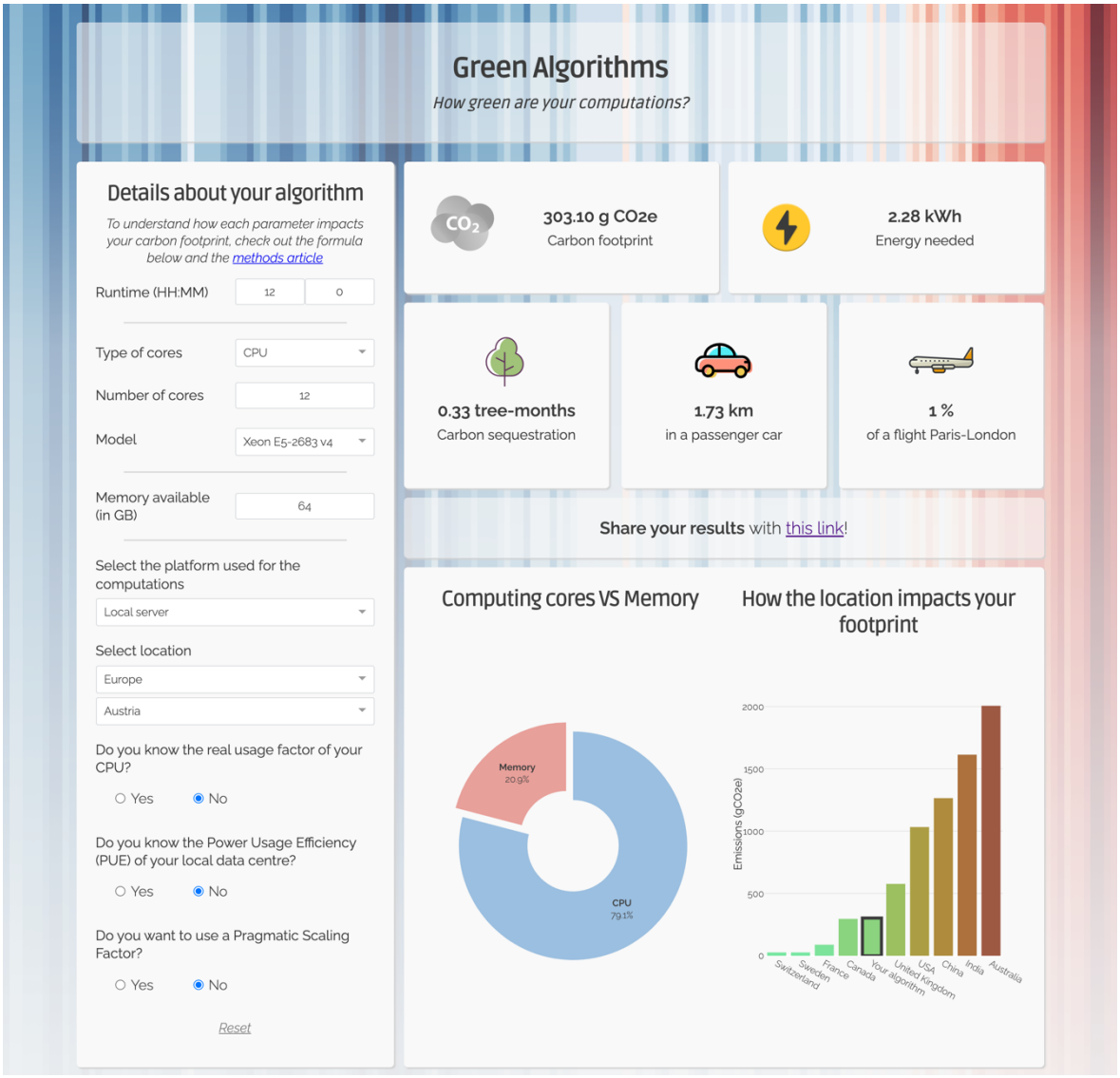


Figure 4: Green Algorithms app (first screen).
The main part of the calculator with input form and results.

[5] https://github.com/GreenAlgorithms/green-algorithms-tool/releases

[6] https://www.fiverr.com/ongcp97

Figure 5: Green Algorithms app (second screen).
It presents details about the methodology.

Uptake

As of March 2022, there have been almost 7,500 users of the app, and recent weeks saw an average traffic of ~180 users each week (**Figure 6**). These figures, and the steady increase in traffic, demonstrates the need for such a tool and the momentum gathered by sustainable research. It is also interesting to note that users come from all over the world.

Figure 6: Audience of the Green Algorithms app, as of March 2022.

Accuracy of the estimations

To assess the accuracy of this method, I would need to physically measure energy usage on a range of hardware, something difficult to do in practice. Fortunately, as part of their defence of the sustainability of large language models, Patterson et al. from Google benchmarked the Green Algorithms calculator [46] for several deep learning language models.

They found that Green Algorithms is actually very accurate for models trained on commercially available GPUs, on average within 1.5% of the measured value, with an

average ratio estimate/measurement of 1.015 (standard deviation = 0.09). However, for models trained on Google's proprietary TPUs[7], Green Algorithms tends to overestimate energy needs by as much as 142% (average ratio 1.8, standard deviation 0.56). This is partly explained by the fact that, contrary to commercial GPUs, the TDP of Google's TPU is unknown, and I had to use estimated and leaked values for the calculator [47]. This shows that with transparency regarding hardware specification, the Green Algorithms is an accurate and reliable tool to estimate carbon emissions.

Removing frictions further: Green Algorithms for High performance Computing

Although the online calculator made it relatively easy for any scientists to estimate the carbon footprint of their computations, it became clear that it was not yet easy enough. Especially for projects involving hundreds of computations with different runtimes and memory requirements, it was not practical to use the calculator accurately. Intending to remove frictions further, I started developing a program that would run directly on the HPC cluster and called it GA4HPC (Green Algorithms for High-Performance Computing).

HPC clusters have a finite set of computing resources (processing cores and memory) and a list of users' requests (usually more requests than resources). The allocation of resources to each user is handled by a workload manager, such as SLURM [48], that considers the resources needed, the job's priority, the expected runtime etc. This workload manager also logs all the resources used for archiving and billing purposes. Users have access to their own usage statistics, and GA4HPC takes advantage of that. In a nutshell, it queries usage information from the workload manager, aggregates and cleans up the information, and then calculates carbon footprints locally using the Green Algorithms method.

It is still at a beta stage, but the first version is up and running on CSD3, the Cambridge HPC cluster. I also made the code available on GitHub[8] so that users in other departments or at other institutions could import it.

[7] A type of GPU designed specifically for deep learning and only available in Google's data centres.

[8] https://github.com/Llannelongue/GreenAlgorithms4HPC

It is extremely straightforward to use; by simply running the command below, users would get the 2021 carbon footprint of all their projects.

```
myCarbonFootprint.sh --startDay 2021-01-01 --endDay 2021-12-31
```

Figure 7 shows the options available. Users can customise the output by restricting it to only certain job IDs or jobs started from certain directories. Figure 8 shows the output. The usage information logged by the workload manager is much more detailed than what can be achieved with the calculator. Besides carbon footprint and context (tree-months, driving, flying), GA4HPC also gives information about memory overallocation and failed jobs. It also breakdowns energy and GHG emissions between CPUs, GPUs and memory.

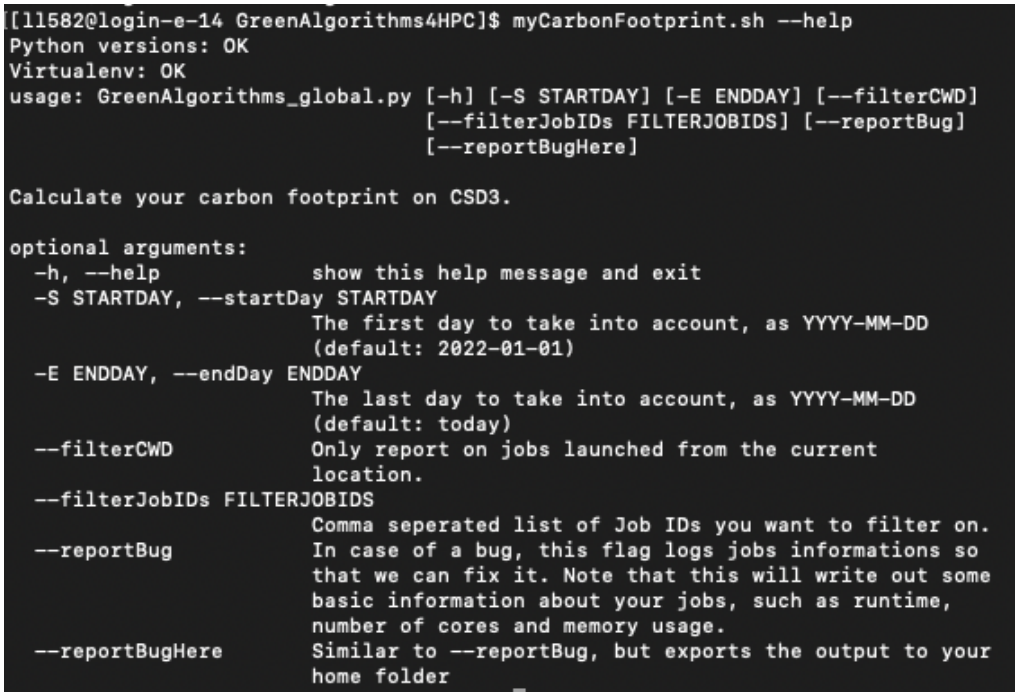


Figure 7: Option available in the GA4HPC tool.

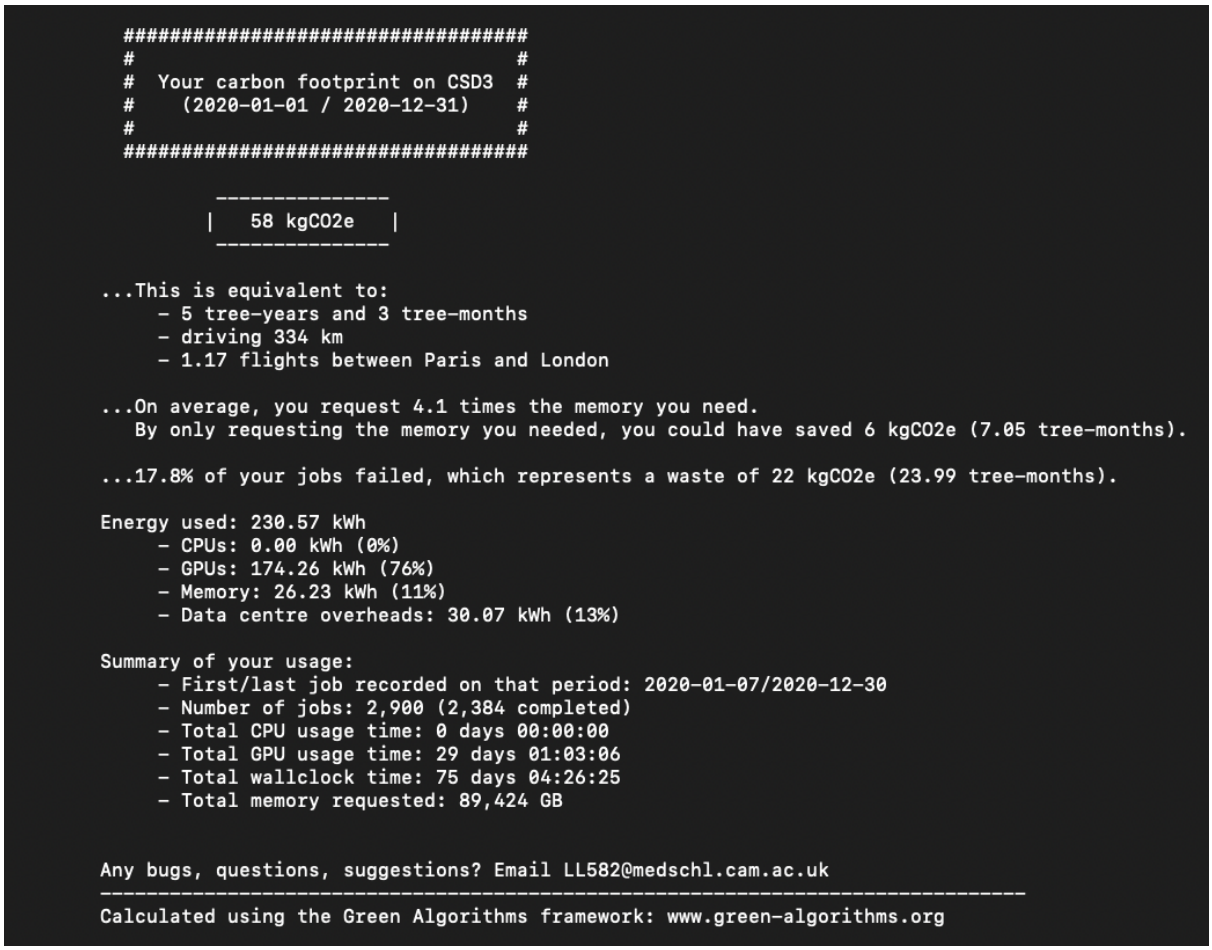


Figure 8: Output of GA4HPC.

Acknowledging the carbon footprint of computational research

In the first place, I applied the Green Algorithms framework to a range of algorithms selected from a variety of scientific fields: physics (particle simulations and DNA irradiation), atmospheric sciences (weather forecasting), and machine learning (natural language processing) (**Figure 9**). Then, in collaboration with Jason Grealey, we focused on computational biology with a more extensive survey of bioinformatic tools. These analyses also provided an opportunity to investigate the carbon impact of common computing practices, such as parallelisation or memory overallocation.

For each task, the published literature was curated to identify peer-reviewed studies that computationally benchmarked popular tools. To be selected, publications had to report at least the runtime and preferably memory usage and hardware used for the experiments, particularly the model and number of processing cores. For parameters independent of the algorithm itself, I used average worldwide values, such as the global average PUE of 1.67 [30] and carbon intensity of 475 gCO_2e/kWh [49]. In-house computations (for cohort-scale eQTL mapping and RNAseq quality control pipelines, code in **Supplementary Note 1**) were run on the Baker Heart and Diabetes Institute's computing cluster (Intel Xeon E5-2683 v4 CPUs and a Tesla T4 GPU) and the University of Cambridge's CSD3 computing cluster (Tesla P100 PCIe GPUs and Xeon Gold 6142 CPUs).

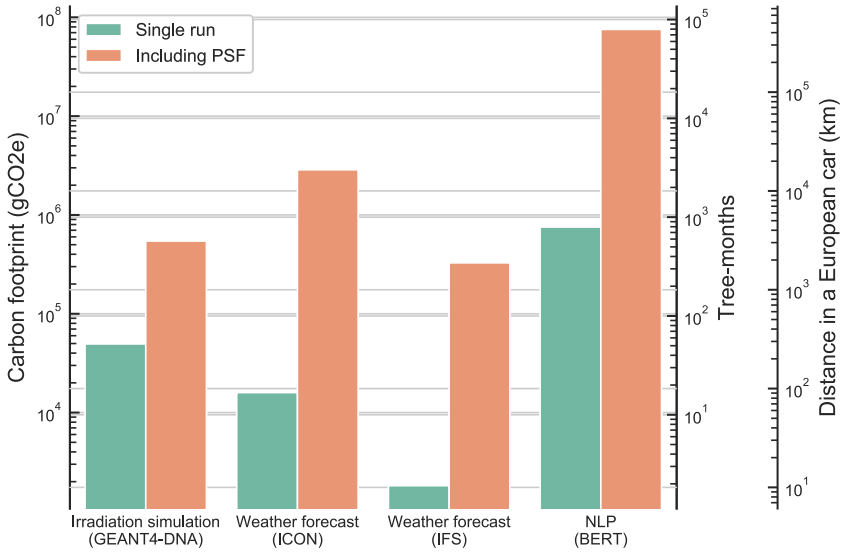


Figure 9: Carbon footprint for a selection of algorithms, with and without their Pragmatic Scaling Factor.

Particle physics simulations

In particle physics, complex simulations are used to model the passage of particles through matter. Geant4 [50] is a popular toolkit based on Monte-Carlo methods with wide-ranging applications, such as the simulation of detectors in the Large Hadron Collider and analysis of radiation burden on patients in clinical practice or external beam therapy [51]–[53]. Meylan et al. investigated the biological effects of ionising radiations on DNA on an entire human genome (6.4×10^9 nucleotide pairs) using GEANT4-DNA, an extension of GEANT4 [54].

To quantify the DNA damage of radiation, they ran experiments with photons of different energy, from 0.5 MeV to 20 MeV. Each experiment ran for three weeks to simulate 5,000 particles (protons) using 24 processing threads and up to 10GB of memory. Using the Green Algorithms tool, and assuming an average CPU power draw (such as the Xeon E5-2680, capable of running 24 threads on 12 cores) and worldwide average values for PUE and carbon intensity, I estimated that a single experiment emits 49,465 gCO_2e. When taking into account a PSF of 11, corresponding to the 11 different energy levels tested, the carbon footprint of this study was 544,115 gCO_2e. Using estimates of car and air travel, 544,115 gCO_2e is approximately equivalent to driving 3,109 km (in a European car) or flying economy from New York to San Francisco. In terms of carbon sequestration, it would take a mature tree 49 years to remove the CO_2 equivalent to the GHG emissions of this study from the atmosphere (593 tree-months) (Figure 9).

A common way to reduce the runtime of algorithms is to distribute the computations over multiple processing cores. If the benefit in terms of time is well documented for each task, as in Schweitzer et al. [55], the environmental impact is usually not considered. GEANT4 is a versatile toolbox; it contains an electromagnetic package simulating particle transport in matter and high energy physics detector response [56]. Schweitzer et al. use a standardised example, TestEm12 [57], to compare the performances of different hardware configurations, from 1 to 60 cores (i.e. a full Xeon Phi CPU). With the Green Algorithms tool, I estimated the carbon footprint of each configuration (Figure 10), which showed that increasing the number of cores up to 15 improved both runtime and GHG emissions. However, when multiplying the number of cores further by 4 (from 15 to 60), the runtime

was only halved, resulting in a two-fold increase in emissions, from 238 to 481 gCO2e. Generally, if the reduction in runtime is lower than the relative increase in the number of cores, distributing the computations will worsen the carbon footprint. In particular, scientists should be mindful of marginal improvements in runtimes which have disproportionally large effects on GHG emissions, as demonstrated by the gap between 30 and 60 cores in Figure 10. For any parallelised computation, there is likely to be a specific optimal number of cores for minimal GHG emissions.

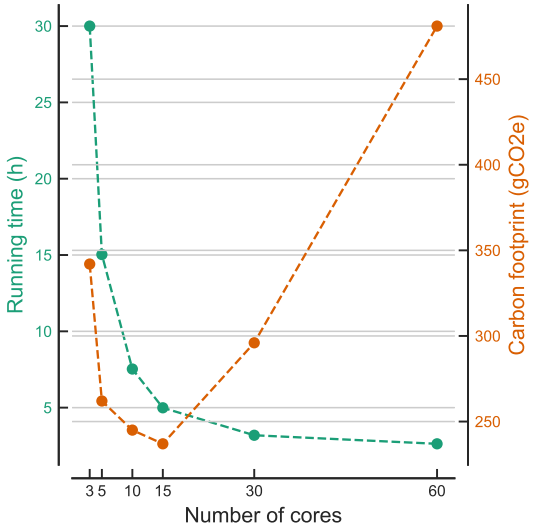


Figure 10: Effect of parallelisation on runtime and carbon footprint, using TestEm12 GEANT4 simulation.

Weather forecasting

Weather forecasts are based on sophisticated models simulating the dynamics between different earth components (such as the atmosphere and oceans). Operational models face stringent time requirements to provide live predictions to the public, with a goal of running about 200-300 forecast days (FDs) in one (wall clock) day [58]. Neumann et al. [58] present the performances of two models in use for current weather forecasts: (i) the Integrated Forecast System (IFS) [59] used by the European Centre for Medium-Range Weather Forecasts (ECMWF) for 10-day forecasts, and (ii) the ICOsahedral Non-hydrostatic (ICON) [60] designed by the German Weather Service (Deutscher Wetterdienst, DWD) and whose predictions are used by more than 30 national weather services [61].

The configurations in daily use by the ECMWF include a supercomputer based in Reading, UK, which has a PUE of 1.45 [62], while ICON is run on the German Meteorological

Computation Centre (DMRZ) [63] based in Germany (PUE unknown). Neumann et al. ran their experiments on hardware similar to that equipped by both facilities, "Broadwell" CPU nodes (Intel E5-2695v4, 36 cores) and minimum 64GB of memory per node. I used these parameters for the CO_2e emission estimates. It is important to note that ICON and IFS solve slightly different problems and are not directly comparable.

The DWD uses ICON with a horizontal resolution of 13km[9] [64] and generates a forecast day in 8 minutes. Based on the experiments run by Neumann et al. [58], this requires 575 Broadwell nodes (20,700 CPU cores). I estimated that generating one forecast day emits 12,848 gCO_2e (14 tree-months). With a runtime of 8min/FD, ICON can generate 180 forecast days in 24 hours. When taking into account this pragmatic scaling factor of 180, I estimated that each day, the ICON weather forecasting algorithm releases approximately 2,312,653 gCO_2e, equivalent to driving 13,215 km or flying from New York to San Francisco four times. In terms of carbon sequestration, the emissions of each day of ICON weather forecast are equivalent to 2,523 tree-months.

At ECMWF, IFS makes 10-day operational weather forecasts with a resolution of 9km. To achieve a similar threshold of 180 FDs/day, 128 Broadwell nodes are necessary (4,608 cores) [58], [65]. Using the PUE of the UK ECMWF facility (1.45), I estimated the impact of producing one forecast day with IFS to be 1,660 gCO_2e. Using a PSF of 180 for one day's forecasts, I estimated emissions of 298,915 gCO_2e, equivalent to driving 1,708 km or three return flights between Paris and London. These emissions are equivalent to 326 tree-months.

Furthermore, I modelled the planned scenario of the ECMWF transferring its supercomputing mid-2022 to Bologna, Italy [66], [67]. Compared to the data centre in Reading, the new data centre in Bologna is estimated to have a more efficient PUE of 1.27 [68]. *Prima facie,* this move appears to save substantial GHG emissions; however, it is notable that the carbon intensity of Italy is 40% higher than the UK in 2022 [36]. Unless the

[9] The horizontal resolution represents the level of geographical detail achieved when modelling the different weather phenomenon.

sources of electricity for the data centre in Bologna are different from the rest of Italy and in the absence of further optimisations, I estimated that the move would result in an 23% increase in GHG emissions from the ECMWF.

Natural language processing

In natural language processing (NLP), the complexity and financial costs of model training are major issues [69]. This has motivated the development of language representations that can be trained once to model the complexity of natural language and could be used as input for more specialised algorithms. The BERT (Bidirectional Encoder Representations from Transformers) [70] algorithm is a field leader that yields both high performance and flexibility: state-of-the-art algorithms for more specific tasks are obtained by fine-tuning a pre-trained BERT model, for example for scientific text analysis [71] or biomedical text mining [72]. Yet, while the BERT model is intended to avoid retraining, many data scientists, perhaps understandably, continue to recreate or attempt to improve upon BERT, leading to redundant and ultimately inefficient computation and excess GHG emissions. Even with optimised hardware (such as NVIDIA Volta GPUs), a BERT training run may take three days or more [73].

Using these optimised parameters, Strubell et al. [10] showed that a run time of 79 hours on 64 Tesla V100 GPUs was necessary to train BERT, with a usage factor of the GPUs of 62.7%. With the Greens Algorithms calculator, I estimated that a BERT training run would emit 754,407 gCO_2e (driving 4,311 km in a European car; 1.3 flights from New York to San Francisco; or 823 tree-months). When considering a conservative PSF of 100 for hyperparameters search, I obtain a carbon footprint of 75,440,740 gCO_2e (6,858 tree-years).

While BERT is a particularly widely utilised NLP tool, Google has also developed a chatbot algorithm, Meena, trained for 30 days on a TPU-v3 Pod containing 2,048 Tensor Processing Unit (TPU) cores [74]. There is limited information on the power draw of TPU cores and memory; however, the power supply of this pod has been estimated to be 288 kW [47]. Using a run time of 30 days, assuming full usage of the TPUs and ignoring memory power draw, the Greens Algorithms calculator estimated that Meena training emitted 164,488,320 gCO_2e, which corresponds to 179,442 tree-months or 71 flights between New

York and Melbourne. It is worth noting that since I did these estimations, Google released measurements of the energy usage to train Meena, 232 MWh [46], resulting in 110,200,000 gCO_2e. The difference between the two is likely due to the lack of transparency regarding the TDP of TPUs and the unknown usage factor (assumed to be 1 in my previous calculations).

Deep dive into the carbon footprint of bioinformatics

Biological and biomedical research now requires the analysis of large and complex datasets, which wouldn't be possible without large-scale computational resources. Whilst bioinformatic research has enabled major advances in the understanding of a myriad of diseases such as cancers [75]–[77] and COVID-19 [78], similarly to all the other fields of science, the environmental costs of the associated computing requirements, and their detrimental impact on human health, have been underappreciated.

The growth of large biological databases, such as UK Biobank [79], All of Us Initiative [80], and Our Future Health [81], has substantially increased the need for computational resources to analyse these data and will continue to do so. With climate change an urgent global emergency, it is important to assess the carbon footprint of these analyses and their requisite computational tools so that environmental impacts can be minimised.

We considered a wide range of bioinformatic analyses: genome assembly, metagenomics, phylogenetics, RNA sequencing, genome-wide association analysis, molecular simulations, and virtual screening (Table 2). For each software, we used benchmarks of runtime and computational resources; in the rare cases where published benchmarks were unavailable, we used in-house analyses to estimate resource usage. We also show that hardware choices substantially affect the carbon footprint of a given analysis, in particular, cloud vs local computing platforms, memory usage, processor options, and parallel computing. The same applies to software choices, including software versions. These results present orders of magnitude, and we note how the estimations are likely to scale with different parameters (e.g. sample size or number of features), but for precise estimations of specific analysis, scientists should estimate their own footprint.

This work is a collaboration with Jason Grealey who coordinated this survey. I first present the tools I investigated and then, in the section ***More results****, I present analyses I didn't lead but to which I contributed.*

Table 2: Carbon footprint of a range of bioinformatic tasks.

Task	Tool	Version	Details about the experiments	Carbon footprint		Tree-months	km in a car (EU)	Runtime and memory	Approximate scaling (if known)
				Increase (%)	$kgCO_2e$				
Genome scaffolding	SSPACE	2.0	Scaffolding 2.4 million long reads from human chromosome 14 [82].	–	**0.0010**	0.0011	0.01	3min 21s 30GB	Linearly with number of reads.
	SOAPdenovo2	r223		**+45%**	**0.0015**	0.0016	0.01	4min 52s 30GB	
	SGA	0.9.43		**+2,752%**	**0.029**	0.032	0.17	1h 35min 30GB	
Genome scaffolding	SSPACE	2.0	Scaffolding 23 million short reads from human chromosome 14 [82].	–	**0.0027**	0.0029	0.02	8min 40s 30GB	
	SOAPdenovo2	r223		**+34%**	**0.0036**	0.0039	0.02	1min 38s 30GB	
	SGA	0.9.43		**+4,801%**	**0.13**	0.14	0.74	7h 05min 30GB	
Genome assembly	Abyss	2.0	De novo assembly of a human genome from Illumina sequencing reads [83].	–	**11**	12	61	20h 34GB	
	MEGAHIT	1.0.6		**+42%**	**15**	16	86	26h 197GB	
Metagenome assembly	MetaVelvet k101	1.2.01	Metagenome assembly from 100 soil samples [84].	–	**14**	16	82	1h 06min 130GB	
	MEGAHIT	1.0.3		**+438%**	**77**	84	439	15h 36min 12GB	
	metaSPAdes	3.8.0		**+1,206%**	**186**	203	1,065	29h 24min 60GB	
Metagenome classification (short read)	Kraken2	2.0.7	Metagenomic classification of 5Gb of randomly sampled reads from Zymo mock community (batch ZRC190633), containing yeast, gram-negative and positive bacteria [85]	–	**0.0052**	0.0057	0.03	20min 21GB	Linearly with number of reads.
	Centrifuge	1.0.4		**+141%**	**0.013**	0.014	0.07	58min 12GB	
	Kraken/Bracken	0.10.5/1.0.0		**+1,650%**	**0.092**	0.10	0.52	1h 40min 154GB	
Metagenome classification (long read)	MetaMaps	-		–	**18.25**	19.91	104.27	209h 53min 262GB	

Phylogenetics	BEAST/BEAGLE	1.8.4/2.1.2	Codon substitution modelling of extant carnivores and a pangolin group. Nucleotide substitution and phylogeographic modelling of Ebola virus genomes. See Supplementary Table 2 for detailed results [86].	–	**0.012 – 0.30**	0.013 – 0.33	0.069 – 1.72	3min 30s to 7h 45min 2 to 8GB	Power law with number of loci.
Phylogenetics	RAxml/ExaML, PhyML, IQ-TREE, FastTree	8.2.0/3.0.17, 20160530 1.4.2, 2.1.9	Over 670,000 tree inferences on about 45,000 single-gene alignments and supermatrices from 19 empirical phylogenomic datasets with thousands of genes and around 200 taxa. [87]	–	**3565**	3889	20,371	300,000h 8GB	
Phylogenetics	ExaML	-	A 322-million-bp MULTIZ alignment of putatively orthologous genome regions across all species, comprising ~30% of an average assembled avian genome. This corresponded to the maximal orthologous sequence obtainable across all orders of *Neoaves*.[88]	–	**4372**	4769	24,983	367,920h 8GB	
RNA read alignment	HISAT2	2.0.0beta	Alignment of 10 million 100-base read pairs to Homo Sapiens hg19 genome [89].	-	**0.0054**	0.0059	0.031	1min 48s 5GB	Linearly with number of reads.
	STAR	2.5.0a		**+78%**	**0.0097**	0.011	0.055	6min 01s 35GB	
	TopHat2	2.1.0		**+5,756%**	**0.32**	0.35	1.81	2h 14min 16GB	
	Novoalign	3.02.13		**+17,926 %**	**0.98**	1.07	5.58	32h 12min 64GB	
RNA read alignment	HISAT2	2.0.0beta	Alignment of 10 million 100-base read pairs to Plasmodium falciparum genome [89].	–	**0.0052**	0.0057	0.030	1min 44s 1GB	
	TopHat2	2.1.0		**+4,519%**	**0.24**	0.26	1.37	1h 25min 13GB	
	STAR	2.5.0a		**+7,025%**	**0.37**	0.40	2.11	2h 27min 8GB	
	Novoalign	3.02.13		**+12,847 %**	**0.67**	0.73	3.83	38h 04min 21GB	

RNA-seq QC pipeline	FastQC, TrimGalore, bbmap/clumpify and STAR	-/v0.6.0/-/v2.7.0e	Quality control analysis of raw reads quality of 392 samples from the Childhood Asthma Study (in-house).	–	**54.97**	59.97	314.11	485h 12min 8GB	
Transcript isoform abundance estimation	Sailfish 1 core	0.6.3	Transcript isoform quantification of 100 million in silico reads generated from Flux Simulator with hg19 genome and GENCODE v19 annotation set [90]	–	**0.0081**	0.0088	0.046	42min 7GB	Linearly with the number of reads.
	Sailfish 16 cores			**+344%**	**0.036**	0.039	0.21	14min 7GB	
	Cufflinks 1 core	2.1.1		**+451%**	**0.045**	0.049	0.26	3h 30min 11GB	
	Cufflinks 16 cores			**+3,262%**	**0.27**	0.30	1.56	1h 45min 12GB	
	RSEM 1 core	1.2.18		**+6,982%**	**0.57**	0.63	3.28	47h 10min 9GB	
	RSEM 16 cores			**+17,162 %**	**1.40**	1.53	8.00	8h 50min 21GB	
GWAS	Bolt-LMM	2.3	Analyses of a single trait in UK Biobank (N=500,000) [91]	–	**4.70**	5.13	26.87	60h 58min 100GB	Linearly with number of variants.
	Bolt-LMM	1.0		**+268%**	**17.29**	18.86	98.81	224h 10min 100GB	
Cohort scale eQTL analysis	TensorQTL	1.0.2	Cis-eQTL mapping of 10.7M SNPs against 18,373 genetic features in a cohort of 2,745 individuals (in-house).	–	**2.04**	2.22	11.7	1h 14min 192GB	Non-linearly with the number of traits or the sample size.
	LIMIX	2.0.3		**+9,256%**	**190.73**	208.07	1,089.9	9705h 41GB – 221GB	
Single cis-eQTL gene mapping	TensorQTL	-	Cis-eQTL mapping one gene from skeletal muscle in GTEx (v6p) [92].	–	**0.00001**	0.00001	0.00004	0.11s 52GB	
	FastQTL	-		**+2,681%**	**0.0002**	0.0002	0.001	30s 52GB	
Molecular dynamics simulation	AMBER	18	Simulation of a Satellite Tobacco Mosaic Virus with 1,066,628 atoms for 100ns * [93], [94].	–	**18**	19	102	75h (^)	
	NAMD	2.13		**+433%**	**95**	104	544	400h (^)	
Molecular Docking	Glide	57111	Molecular docking of four DUD systems, scaled to 1m ligands [95]	–	**13**	14	74	1,027h 47min 0.05GB	
	rDock	-		**+1,092%**	**154**	168	878	12,250h 0.05GB	
	AutoDock Vina	-		**+3,886%**	**514**	561	2,938	40,972h 0.05GB	

***** Note different simulation parameters between the two: AMBER18 (4fs timestep, 9A cut-off) NAMD (2fs timestep with rigid bonds, 12A cut-off with PME every two steps).
^No memory included due to a lack of information.

Metagenomics classification

Metagenomics is the sequencing and analysis of all genetic material in a sample. Dilthey et al. [85] benchmarked several metagenomic classifiers, MetaMaps [85], Kraken2 [96], Kraken/Bracken [97], [98], and Centrifuge [99]. They compared these tools on ~5Gb of randomly sampled reads from an Oxford Nanopore GridION sequencing run from Zymo mock communities, comprising five Gram-positive bacteria, three Gram-negative bacteria and two types of yeast. Carbon footprints differed by several orders of magnitude, 18.25 $kgCO_2e$ for the long-read classifier MetaMaps but less than 0.1 $kgCO_2e$ for the short-read classifiers. (Table 2). The carbon footprints per Gb of classified reads ranged from 0.001 to 0.018 $kgCO_2e$ (0.001 to 0.02 tree-months) using the short-read classifiers (Kraken2, Centrifuge, Kraken/Bracken) and 3.65 $kgCO_2e$ (4 tree-months) when using MetaMaps. Kraken2 had the highest performance over all taxonomic ranks when all reads were assembled, followed by Kraken/Bracken, Centrifuge and MetaMaps. However, when considering long reads (>1000bp), MetaMaps had the highest precision and recall for all available taxonomic levels, followed by Kraken2, Kraken/Bracken, and Centrifuge.

Genome-wide association analysis

Genome-wide association analysis aims to identify genetic variants across the genome associated with a phenotype. We assessed both genome-wide association studies (GWAS) and expression qualitative trait locus (eQTL) mapping. We estimated the carbon footprint of GWAS with two different versions of Bolt-LMM [91] on the UK Biobank [79] (500k individuals, 93M imputed SNPs). We found that a single trait GWAS would emit 17.29 $kgCO_2e$ with Bolt-LMM v1 and 4.70 $kgCO_2e$ with Bolt-LMM v2.3 (Table 2), a reduction of 73%. GWAS typically assess multiple phenotypes, e.g. metabolomics GWAS consider from several hundred to several thousand metabolites; since the association models in GWAS are typically fit on a per-trait basis, the carbon footprint is proportional to the number of traits analysed. Bolt-LMM's carbon footprint also scales linearly with the number of genetic variants [100], meaning that a single biobank-scale GWAS using UK Biobank (500k individuals) has a carbon footprint of 0.05 $kgCO_2e$ per million variants (0.06 tree-months) with Bolt-LMM v2.3 and 0.2 $kgCO_2e$ per million variants (0.2 tree-months) with Bolt-LMM

v1. However, Bolt-LMM doesn't scale linearly with the number of samples (*time ~ $O(N^{1.5})$* [100]), which must be taken into account when scaling the values to different sample sizes.

For cis-eQTL mapping, we compared the carbon footprint using either CPUs or GPUs on two datasets, first on a small sample size using skeletal muscle data from GTEx [101] (1 gene, 700 individuals) with a benchmark of FastQTL (CPU) [102] and TensorQTL (GPU) [92], [103] from Taylor-Weiner et al. [92]. Both tools were shown to yield similar mappings. Secondly, we used an in-house assessment to estimate the carbon footprint of a CPU-based analysis with LIMIX [104] and with the GPU-based TensorQTL, using a larger cohort of 2,745 individuals with 18k genetic features and 10.7m SNPs (Table 2). In both cases, footprints were lower (28x and 94x) when using GPUs instead of CPUs. The scaling of eQTLs is complex, and the carbon footprint doesn't scale linearly with the number of traits or sample size [92], [104].

Molecular dynamic simulations

Molecular simulations and virtual screening use computational simulations to model and understand molecular behaviour and *in silico* scanning of small molecules for drug discovery. We estimated the carbon footprint of simulating molecular dynamics of the Satellite Tobacco Mosaic Virus (1,066,628 atoms) for 100ns (nanoseconds) using AMBER and NAMD [93], [94] [105], [106], and obtained between 18 and 95 $kgCO_2e$, which corresponds to 0.2 to 1 $kgCO_2e$ per ns (Table 2). It should be noted that there are small discrepancies between the simulation parameters used by the tools so they can't be compared directly, and due to a lack of information, neither of these estimations include the power usage from memory.

More results

Genome and metagenome assembly

Genome assembly is the process of combining sequencing reads (short or long reads, or a combination) into a single or a set of consensus sequences for an organism. Hunt et al. [82] compared SSPACE [107], SGA [108] and SOAPdenovo2 [109] for genome scaffolding using contigs produced with the Velvet assembler [110] and the human chromosome 14 GAGE dataset [111]; two read sets were compared, one using 22.7 million short reads (fragment

length of 3 kb) and the other 2.4 million long reads (35 kb). Scaffolding the short or long reads resulted in similarly low carbon footprints (0.0010 to 0.13 $kgCO_2e$) (Table 2). However, SGA had a carbon footprint up to 49 times higher than the other tools, but it may result from the increased time needed to build the FM index (full-text minute-space index) [108]. As the runtime of many genome assembly tools scales linearly with the number of reads [112], these results equate to between 0.00012 to 0.0057 $kgCO_2e$ (0.00013 to 0.0063 tree-months) per million short reads assembled and 0.00043 to 0.012 $kgCO_2e$ (0.00047 to 0.013 tree-months) per million long reads assembled. On average, long read assembly had a carbon footprint per million reads 3.2x larger than short-read assembly for the tools we measured. All three methods had similar performance on these read sets, with SOAPdenovo2 slightly outperforming SGA and SSPACE.

For whole-genome assembly of humans, ABySS [83] and MEGAHIT [113] were benchmarked by Jackman et al. [83] using Illumina short-read sequencing (815M reads, 379M uniquely mapped reads, 6kbp mean insert size). We estimated the carbon footprint of these tasks to be between 11 and 15 $kgCO_2e$ (12 to 16 tree-months) (Table 2), or per million reads, between 0.013 and 0.019 $kgCO_2e$ (0.014 to 0.020 tree-months). It is difficult to succinctly quantify the accuracy of these tools as it has been shown to vary greatly between use cases and datasets [114]. Instead, relevant published benchmarks, such as [83], [114], [115], can indicate the assembler that excels in the area of interest, e.g. number of error-free bases, coverage or continuity.

Metagenomic assembly

Based on a benchmark by Vollmers et al. [84], we estimated the carbon footprint of metagenome assembly with three commonly used assemblers, metaSPAdes [116], MEGAHIT [113] and MetaVelvet (k-mer length 101bp) [117] on 100 samples from forest soil (33M reads, median length 360 bp). It ranged between 14 and 186 $kgCO_2e$ (Table 2), corresponding to 0.14 to 1.9 $kgCO_2e$ per sample (0.2 to 2 tree-months). MetaSPAdes had the greatest carbon footprint but also the best performance, followed by MetaVelvet and MEGAHIT, respectively.

Phylogenetics

Phylogenetics uses genetic information to analyse the evolutionary history and relationships amongst individuals or groups. Baele et al. [86] benchmarked nucleotide substitution models with and without spatial location information to study the evolution of the Ebola virus during the 2013-2016 West African epidemics (1,610 genomes, 18,992 nucleotides [118]). These nucleotide substitution models are based on a four-partition model (one for each codon position and one for the intergenic region) as well as generalised linear models [118] when including spatial information in the phylogeographic analysis. Additionally, Baele et al. benchmarked more complex Goldman and Yang's [119] codon substitution models on a set of mitochondrial genomes from extant carnivores and a pangolin outgroup. For all these tasks, they used the Bayesian inference framework implemented in BEAST [120] combined with BEAGLE [121] for computational speedup.

We estimated the carbon footprint of nucleotide-based modelling of the Ebola virus dataset was between 0.012 and 0.076 $kgCO_2e$ depending on hardware choices and up to 25 times higher (up to 0.30 $kgCO_2e$) when including spatial information. More complex codon modelling of extant carnivores and pangolins resulted in a greater footprint, from 0.017 to 0.10 $kgCO_2e$ (Figure 12, Table 2, Supplementary Table 2). The impact of hardware choices illustrates a trade-off between runtime and carbon footprints, discussed in more detail below. It should be noted that the runtime of BEAST, and therefore its carbon footprint, scales as a power law, i.e. not linearly, with the number of loci [122].

We also estimated the carbon footprint of two large scale empirical phylogenetic studies that each used over 300,000 CPU hours (Table 2) [87], [88]. As both studies lacked hardware information, we assumed a CPU power draw of 12W per core (the average from our database). Four different maximum likelihood-based phylogenetic programs were evaluated - RAxML [123] with ExaML [124], PhyML [125], [126], IQ-TREE [127], and FastTree [128] - by conducting more than 670,000 tree inferences on 19 empirical phylogenomic data sets with thousands of genes and around 200 taxa. We estimated this would have a carbon footprint of 3,565 $kgCO_2e$ (3,889 tree-months or 324 tree-years). Additionally, using the maximum likelihood program ExaML, Jarvis et al. performed a 322-million-bp MULTIZ

alignment of putatively orthologous genome regions across 48 species of *Neoaves* and had a similarly large carbon footprint of 4,372 $kgCO_2e$ (4,769 tree-months).

RNA sequencing

RNA sequencing (RNAseq) is the sequencing and analysis of all RNA in a sample. We first assessed the read alignment step in RNAseq using an extensive benchmark where Baruzzo et al. looked at different datasets of 10 million 100-base paired-end strand-specific simulated reads of two different genomes, *Homo Sapiens* (hg19) and *Plasmodium falciparum* [89], which have substantially differing levels of complexity (*P. falciparum* has higher rates of polymorphisms and errors). We estimated the carbon footprint of aligning two sets of reads, one to each genome (T1 human and T3 Malaria). The three most-cited software tested, STAR [129], HISAT2 [130, p. 2] and TopHat2 [131], all had low recall when aligning malaria reads to the *P. falciparum* genome, so we also assessed Novoalign [132] as it performed significantly better for this task (Table 2). The carbon footprints ranged from 0.0052 to 0.67 $kgCO_2e$ for *P. falciparum,* with Novoalign having both the best performances and the largest carbon footprint. For human read alignment, despite all four methods obtaining high recall, their footprints varied by over two orders of magnitude (0.0054 to 0.98 $kgCO_2e$). As alignment tools are often reported with alignment speed (number of reads aligned in a given time) [129], [130, p. 2], the carbon footprints of the analyses above scaled accordingly and ranged from 0.001 to 0.1 $kgCO_2e$ per million human or *P. falciparum* reads (0.001 to 0.1 tree-months).

To quantify the carbon footprint of a full quality control pipeline with FastQC, we utilised 392 RNAseq read sets obtained from PBMC samples [133], [134], with a median depth of 45 million paired-end reads and average length 146bp. Adapters were trimmed with TrimGalore [135], followed by the removal of optical duplicates using bbmap/clumpify [136]. Reads were then aligned to the human genome reference (Ensemble GRCh 38.98) using STAR [129]. We estimated the carbon footprint of this pipeline to be 54.97 $kgCO_2e$ for the full dataset, or 1.22 $kgCO_2e$ per million reads (Table 2), which scales linearly with the number of reads (Supplementary Table 3).

For transcript isoform abundance estimation, we assessed Sailfish [137], RSEM [138], and Cufflinks [139] using the benchmark from Kanitz et al. [90] on simulated human RNAseq

data (hg19). The Flux Simulator software [140] and GENCODE [141] were used to generate 100 million single-end 50bp reads. The carbon footprints of this task were between 0.0081 and 1.40 $kgCO_2e$ (Table 2), and the authors showed that the time complexity, and therefore the carbon footprint, is proportional to the number of reads. Additionally, these tools offer the option of parallelisation, which can reduce runtime but, in this case, not carbon footprint; indeed, the decrease in runtime when using 16 cores instead of one was not sufficient to offset the increase in power consumption, which resulted in a 2- to 6-fold increase in carbon footprint when utilising 16 cores (Table 2). There were significant differences between tools despite RSEM and Sailfish having similar accuracy performances in this benchmark. Since Sailfish did not perform a read alignment step, it was on average 53 times faster than RSEM, with a carbon footprint 71 times smaller when using 1 core and 39 times smaller with 16 cores. Lastly, whilst Cufflinks is largely used for abundance estimation, its main purpose is transcript isoform assembly, resulting in a significantly lower accuracy (at a higher carbon cost).

Molecular docking

Using a benchmark from Ruiz-Carmona et al. [95], we estimated the carbon footprint of three molecular docking methods, AutoDock Vina, Glide and rDock [95], [142], [143]. The data originate from four systems (ADA, COMT, PARP, and Trypsin) from the Directory of Useful Decoys benchmark set [144]. To estimate their carbon footprints, we used the average computational runtimes for a 1 million ligand campaign and found values ranging from 13 to 514 $kgCO_2e$ (Table 2). Glide was the fastest tool and had the smallest footprint, although it is not freely available. Of the two freely available tools (AutoDock Vina and rDock), rDock had the smallest carbon footprint with a performance comparable to Glide [95].

Local vs cloud data centres and the role of geography

Cloud computing facilities and large data centres are optimised to significantly reduce overhead power consumption such as cooling and lighting and, as such, are often more energy-efficient than smaller facilities. A report from 2016 estimated that energy usage by data centres in the US could be reduced by 25% if 80% of the smaller data centres were aggregated into larger and more efficient data centres (hyperscale facilities) [145].

Compared to the global average PUE of 1.67, Google Cloud's average PUE of 1.11 [146] reduces the carbon footprint of a task by 34%. Other cloud providers also achieve low PUEs, Microsoft Azure reduces the carbon footprint by 33% (PUE=1.125 [147]) and Amazon Web Service by 28% (PUE=1.2 [148]).

The use of cloud facilities may also enable further carbon footprint reductions by allowing users to choose a geographic location with relatively low carbon intensity. As an example, we found that a typical GWAS of UK Biobank considering 100 traits using the aforementioned GWAS framework (see **Genome-wide association analysis**) together with BoltLMM v2.3 on a Google Cloud server in the UK would lower the carbon footprint by 81% when compared to the average local data centre in Australia (**Figure 11**), potentially saving 705 $kgCO_2e$ (769 tree-months, or 64 tree-years). To find the optimal strategy for specific analysis and facilities, it is best to directly use the Green Algorithm calculator.

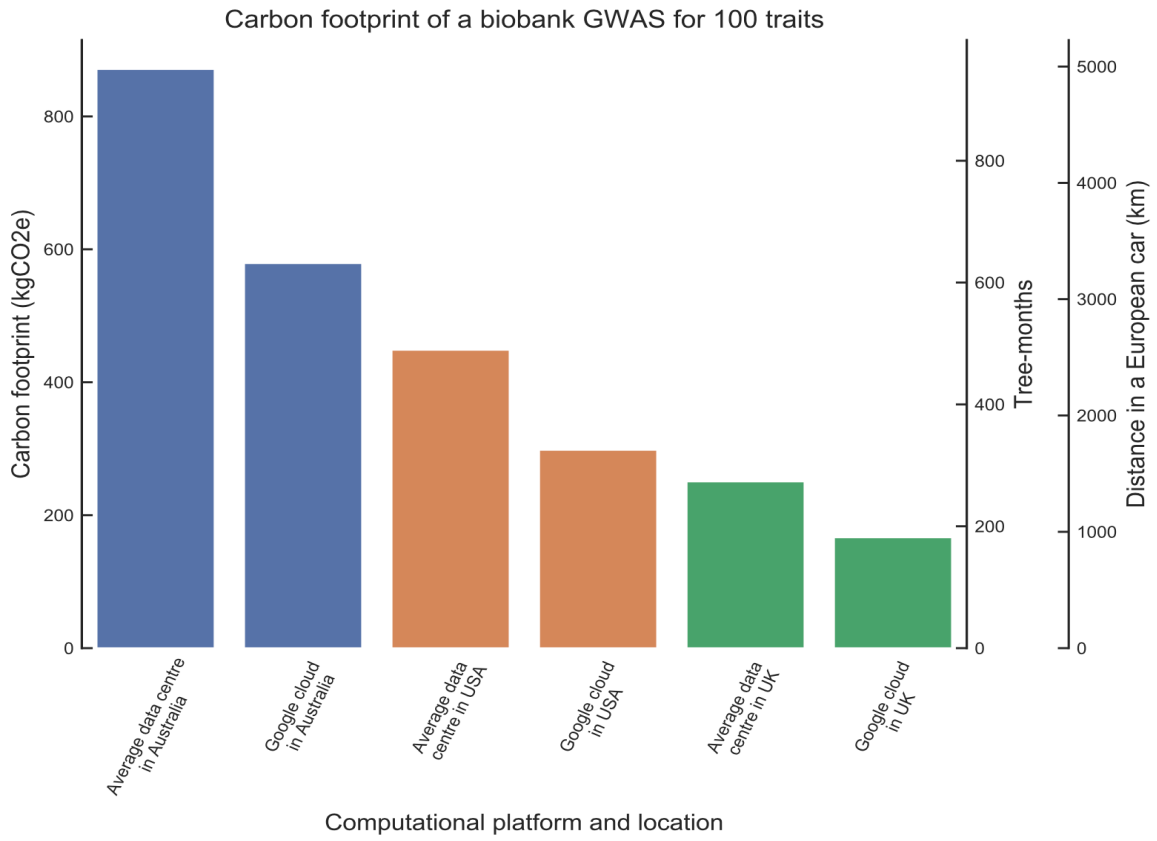


Figure 11: Impact of location and computational platform on carbon footprint of a GWAS. Carbon footprint (in kgCO2e, tree-months, and European car km) of a biobank scale 100 trait GWAS in various locations and platforms. Average data centres have a PUE of 1.67 [30], Google cloud has a PUE of 1.11 [146], Australia has a carbon intensity of 0.88 kgCO2e/kWh, the USA 0.453 kgCO2e/kWh, and the UK 0.253 kgCO2e/kWh [149].

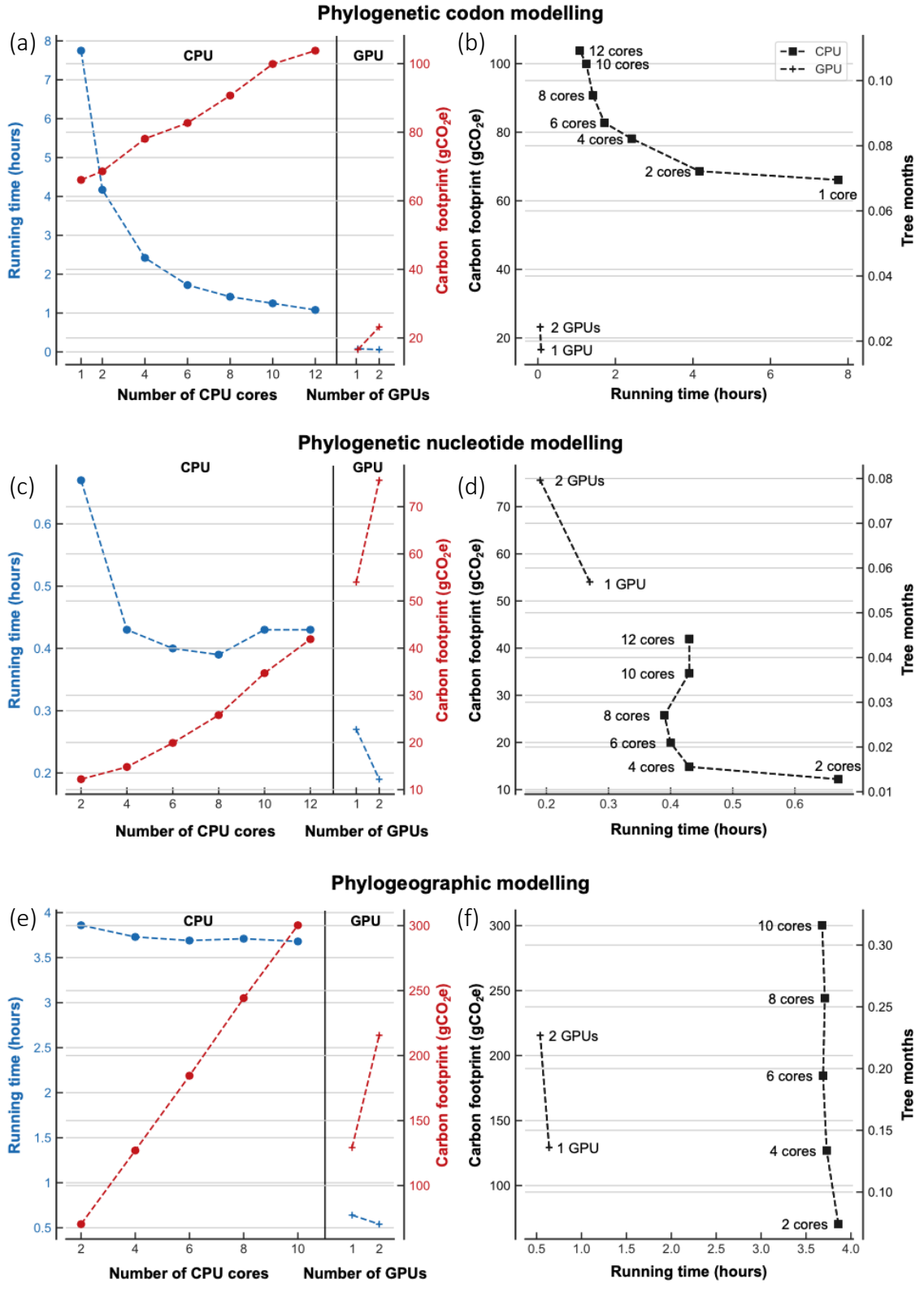


Figure 12: The effect of processor choices and parallelisation on carbon footprint.
The carbon footprint of BEAST/Beagle implemented on multi-core CPUs or GPUs for three different tasks. The plots on the left (a, c and e) detail both the runtime and carbon footprint against the number of cores utilised. The plots on the right (b, d an f) detail the runtime solely against carbon footprint (contextualised with tree-months) for both CPUs and GPUs. The numerical data is available in Supplementary Table 2.

The impact of hardware

It is common to use parallelisation to share the workload between several computing cores and reduce the total runtime. However, we showed above that this could increase carbon footprint, and we found that parallelisation frequently results in trade-offs between runtime and carbon footprint (**Figure 10**). A general optimal solution to this trade-off is difficult to find as the relationship between carbon footprint and number of cores used may not be linear depending on the power management strategy of the servers. For modelling purposes, we assume here that cores are allocated independently to different users and that each core is used at 100%. In some cases, the reduction in runtime is substantial. For example, executing the phylogenetic codon model (**Phylogenetics**) on a single core would take 7.8 hours and emit 0.066 $kgCO_2e$, but with two cores, the carbon footprint increased by only 4% while runtime was decreased by 46% (1.9x speedup) (**Figure 12 a and b**, **Supplementary Table 2**). With 12 cores, runtime decreased 86% (7.2x speedup) but the carbon footprint increased by 57%. In other cases, speedup was marginal, making the added GHG emissions unnecessary. E.g. the phylogeographic model had a runtime of 3.86 hours with a carbon footprint of 0.070 $kgCO_2e$ when using two cores; increasing to 10 cores reduced runtime by only 5% but increased carbon footprint by 4-fold (**Figure 12 e and f, Supplementary Table 2**).

We estimated the carbon footprint of algorithms executed on both GPUs and CPUs. For cis-eQTL mapping (**Genome-wide association analysis**), we estimated that, compared to CPU-based FastQTL and LIMIX, using a GPU-based software like TensorQTL can reduce the carbon footprint by 96% and 99% and the runtime by 99.63% and 99.99%, respectively (**Table 2**). For the codon modelling benchmark (**Phylogenetics**), utilising GPUs had a speedup factor of 93x and 13x when compared to 1 and 12 CPU cores, resulting in a decrease in carbon footprint of 75% and 84% respectively. These estimations demonstrate that GPUs can be well suited to both reducing runtime and carbon footprint for algorithms.

However, there are situations where the use of GPUs can increase carbon footprints. Using a GPU instead of 8 CPU cores for phylogenetic nucleotide modelling decreased runtime by 31% but also doubled the carbon footprint. We estimated that a single GPU would need to run the model in under four minutes to match the CPU's carbon footprint, as opposed to

the 16 minutes it currently takes. Similarly, using a GPU for the phylogeographic modelling of the Ebola virus dataset reduced the runtime by 83% (6x speedup) when compared to the method with the lowest footprint (2 CPU cores) but increased carbon footprint by 84%. The exact equations are shown below, but a simple approximation can be used by scaling the runtime of the GPU by the ratio of the power draws of the CPU and GPU. For example, we compared the popular Xeon E5-2683 CPU (using all 16 cores, power draw of 120W) to the Tesla V100 GPU (300W) and found that to have the same carbon footprint with both configurations, an algorithm needs to run ~2.5 times (300/120) faster on GPU than CPU.

More precisely, from rearranging equation **(4)**, it can be shown that the runtime $t_{GPU,eq}$ at which GPU has a lower carbon footprint is (with the same notations):

$$t_{GPU,eq} = t_{CPU\times}\left(\frac{n_{CPU} \times P_{CPU} \times u_{CPU} \ + \ n_{m,CPU} \times P_m}{n_{GPU} \times P_{GPU} \times u_{GPU} \ + \ n_{m,GPU} \times P_m}\right) \quad (5)$$

Or put more simply, when ignoring memory and utilising 1 CPU and 1 GPU with identical core usage factors, this simplifies to:

$$t_{GPU,eq} = t_{CPU\times}\left(\frac{P_{CPU}}{P_{GPU}}\right) \quad (6)$$

Where, t_{CPU} is scaled by the ratio of the power required to utilise the CPU to the GPU.

Towards more sustainable computational research

Many guidelines on how to move towards more sustainable scientific practice can be inferred from the discussions above. However, it would be clearer, and probably more effective, to gather the most impactful actions scientists can take in a single article directed at all computer scientists. This is the aim of this section, extracted from our Ten Simple Rules article [15].

We are in the midst of a man-made climate emergency, and yet, there is a widespread underappreciation in our community as to the effects of our computations on carbon emissions and global warming. Global temperatures are rising, largely caused by greenhouse gas (GHG) emissions, which is leading to the melting of the ice caps and permafrost. Acidification of the oceans is threatening the entire ocean ecosystem. Global

biodiversity is rapidly declining across the globe. Although science can contribute to our understanding of the environment and has great potential to develop strategies and technology to slow down global warming, we do not have a free pass when it comes to the environmental impact of our work. There is little doubt that data science in particular will be a key tool to tackle climate change, but we often forget to consider how our own work also contributes to the problem. Since the start of this project, I have been wondering why.

One of the reasons could be that it has always been assumed that the potential gains would outweigh the costs, especially in biology, where treatments for diseases is a major goal. Although this is most likely true in many cases, it can only be verified by acknowledging the environmental costs in the first place. The distance between users and sources of GHG emissions can also explain this. When the computer is a loud tower in the corner of the room, the countless cables and the noisy fan are a reminder of the sort of energy needed to power it. On the other hand, the physical distance between users and data centres, which makes running demanding algorithms easier than ever, also make us forget about the resources used behind the scenes. Moreover, energy efficiency and carbon footprint are rarely a topic of discussion for computing hardware. Fridges, dishwashers, TVs, cars etc., all these devices promote their low energy needs and sustainability: being (or claiming to be) environmentally friendly is now a key element of an advertising campaign (as well as a legal requirement in many cases). This is not yet the case for computers, for which mostly speed and performance are discussed, and when energy efficiency is brought up to non-IT-specialists, it is usually to promote battery life and fast charging, not sustainability. Notwithstanding that computing time is virtually free for a lot of scientists who have access to institutional servers. Many scientists are also under the false impression that there is little they can do to make their research more sustainable; the ten rules below address this. They are meant as a succinct, accessible summary of the best ways a scientist can move towards greener computational science.

Ten Simple Rules for greener computing

Rule 1: Calculate the carbon footprint of your work

We live in a world ruled by data, where a problem doesn't exist until it has been measured. There is still very limited information available about the carbon footprint of computational research, so the only way to appreciate the scale of the issue is for all of us to routinely calculate and report the carbon footprint of our work. This may seem intimidating at first, but as I have shown above, estimating it can be quite straightforward, for example with the Green Algorithms calculator.

When doing so, we should not forget the carbon footprint of long-term storage. Although servers represent the largest share, over 50%, of a data centre's energy usage, hard-drive storage accounts for about 10% of the electricity bill [145]. As discussed in the Storage section (p.123), the order of magnitude of the carbon footprint of storage is around 10 $kgCO_2e/TB/year$.

Finally, the end-to-end environmental impact of computers and data centres is substantial but difficult to quantify. In a life cycle assessment, all aspects are taken into account, from the extraction of raw materials to transportation, manufacturing and disposal [13]. One thing is certain, a clear way to limit your impact is to reduce technological waste (Rule 3).

Rule 2: Include the carbon footprint in your cost-benefit analysis

When deciding whether to start a project, the estimated carbon footprint should be considered. This would both motivate researchers to reduce the environmental cost of their computation (using rules 1 to 7) and shatter the illusion of "free compute time". The concepts discussed in these ten simple rules highlight important questions that need to be answered beforehand, such as: "Is there a real need for speed if it comes at the expense of significant GHG emissions?", "Can investing in new hardware be a sustainable solution?", "Can we move our computation to a greener facility?" and "Are funds available to offset our carbon footprint?". In the same way that no wet-lab experiment would be launched without an estimation of the financial costs, the environmental cost should be an integral part of modern science.

Rule 3: Keep, repair and reuse devices to minimise electronic waste

The devices that we use, such as laptops, notepads, phones etc., have significant environmental impacts. As of 2015, the global carbon footprint of manufacturing and using computers (laptop and desktop) was 150 Mt CO_2e, and 72 Mt CO_2e for smartphones [150]. These devices have been heavily optimised to be energy efficient, and in general, the energy consumption of our desktop set-up is relatively low: between 70 and 80% of the cradle-to-grave carbon footprint of devices comes from the production lifecycle [150]. For reference, the total carbon footprint of an iPhone 12 used for 3 years is 70 kgCO_2e [151], and most recent smartphones are in the range 55-80 kgCO_2e [150]. Unsurprisingly, laptops have a higher carbon footprint, around 280 kg CO_2e; 185 kgCO_2e for a 2020 13in MacBook Pro, but over twice as much (394 kgCO_2e) for a 16in model (used 4 years) [151].

There are several ways to reduce the environmental impact of these devices. Firstly, do you need so many of them? Having fewer devices will reduce your total environmental impact. Secondly, try to use your gear for as long as is reasonable. As mentioned above, the carbon footprint of using a device is relatively small compared to the cost of producing it, and this usage footprint improves only marginally in new models (15 kgCO2e in an iPhone 4 from 2010 vs 13 kg in an iPhone X from 2017 [151]). This shows that limiting production footprint by keeping, maintaining and passing on the same device for longer can drastically reduce our environmental impact. To achieve that, try to have your devices fixed rather than replaced. This requires manufacturers to make their products repairable (with detachable batteries for example), and although some companies like Fairphone work in this direction, easily fixable phones remain too rare. Finally, it's important to ensure that devices are disposed of through the correct route. Informal recycling of electronic waste, which is generally labour-intensive, low-paid and unregulated [152], causes high levels of air, soil and water pollution in recycling areas (often in low- and middle-income countries), which poses serious risks to human health and the environment [153]. Remarkably, some devices like the Fairphone can even have a negative end-of-life footprint (i.e. a beneficial environmental impact) if recycled correctly by recovering precious metal [154].

Rule 4: Choose your computing facility

Due to the differences in energy production methods between countries, geographic location is perhaps the actionable factor with the greatest impact on carbon footprint (Figure 3 p125). For example, the footprint of producing 1 kWh of electricity in Switzerland is 12 gCO_2e, thanks to low-carbon energy production methods such as hydroelectricity and nuclear power. Australia, on the other hand, relies largely on coal and natural gas, which explains why producing the same kilo-Watthour has a carbon footprint of 880 gCO_2e, 73 times larger [155].

Furthermore, regardless of the location, all computing facilities are not equal. Infrastructure overheads (mainly cooling) are responsible for 40% of the total electricity bill of an average data centre [30], but large optimised ones can reduce these overheads by 84% so that they only represent 10% of the total electricity usage [33]. With cloud computing, you can both use optimised data centres and choose their location, reducing energy needs and carbon footprint significantly.

Finally, whether you use institutional data centres or cloud facilities, transparency regarding energy usage is crucial. Computational facilities need to be open about where the electricity they use comes from and their own energy efficiency.

Rule 5: Choose your hardware carefully

Processors, memory and runtime are the main parameters that control the energy usage, and the carbon footprint, of a task. By adjusting the type of processor used and the memory available, something that users can do on most HPC platforms, the carbon footprint of a task can be greatly reduced. Among these, memory is perhaps the easiest one to act upon since, and this may come as a surprise, the energy needs depend almost entirely on the memory available, not on the memory actually used [23]. This means that allocating extra gigabytes to a task "just to be safe" or to avoid having to optimise the code (we have all done it) increases the carbon footprint substantially. The number and type of processors are often optimised to reduce runtime through parallelisation or by replacing CPUs with GPUs. However, the trade-off regarding carbon footprint can be complicated: both these choices can sometimes increase the energy usage of a task despite reducing the total

runtime. Whether these strategies can reduce both carbon footprint and runtime need to be assessed for each situation.

Investing in more energy-efficient hardware can be a way to reduce the carbon footprint of a task without compromising on performance. However, as we have just discussed in Rule 3, it is important to take into account the environmental impact of producing the new equipment and disposing of the old one.

Rule 6: Increase efficiency of the code

There are many avenues leading to more energy-efficient code, even if you don't write the code yourself: one of the simplest ways to reduce the carbon footprint of your analysis is simply to ensure that you are using the most efficient software for the task. For example, we estimated that a biobank scale GWAS with the v1 of BOLT-LMM emits a substantial 17 $kgCO_2e$; however, simply updating to the latest v2.3 reduces the carbon footprint by 73% (to 5 $kgCO_2e$) (Genome-wide association analysis).

When writing code, keep in mind the key principles mentioned here: limit peak memory requirements, be wary of hardware choices resulting in only small runtime improvements (which often come with large carbon footprints), and use recent optimised libraries when possible. In particular, code profilers such as Rstudio's built-in tool (and the *profvis* visualisation library) or the equivalent *cProfile* package in Python can be used to find sections that would benefit most from optimisation.

An algorithm's carbon footprint is proportional to how many times it is used, so increasing the efficiency of a more popular tool will have a larger overall reduction in total GHG emissions. This is why, as a community, we propose that heavily utilised software should be prioritised for optimisation. Software developers should be aware of this proportionality as it is an avenue for them to limit the environmental impact of their work.

Rule 7: Be a frugal analyst

As part of a project, an analysis is rarely performed just once. Often, the same computational pipeline is run several times to debug, optimise and replicate, which multiplies the carbon footprint of the project. Keeping these duplications to a minimum is

a painless way to reduce your carbon footprint. There are many ways to do that; we describe some below.

First, test and debug the entire pipeline on small example datasets. This reduces both the runtime and the memory requirements, which saves you time, money and will significantly reduce your carbon footprint. Then, if despite careful debugging, some steps are known to have a high risk of failure, checkpoint your code (i.e. save intermediary states to resume from in case of error) to limit unnecessary computations.

Finding the optimal settings for a software requires extensive testing, which causes disproportionate GHG emissions. Start by looking into the software's documentation and public benchmarks to find optimal hardware choices and minimum memory requirements, which will be clearly identified if developers followed the previous rule. In some cases, it is possible to do the remaining tests on a smaller, representative dataset, and when it is not, strategies exist to reduce the number of trials and errors. For example, instead of testing all possible combinations of parameters, performing a random search has been shown to be more efficient and yields better results [156]. Some approaches have also been proposed to speed up inference and evaluation to respect a computational budget [157], [158].

Rule 8: Releasing a new software? Make its hardware requirements and carbon footprint clear

As discussed in Rule 5, selecting the type of hardware to use for an analysis - CPU or GPU, number of cores and memory size - has a significant impact on the carbon footprint. To ensure that new softwares are used as sustainably as possible, their developers should make it clear what the best hardware choices are for each situation.

In particular, it is crucial to detail how the memory requirements scale with the dimension of the input data. Otherwise, users tend to over-allocate memory out of caution or under-allocate memory and have to restart the analysis several times, both resulting in unnecessary GHG emissions. It is also useful to make it clear when increasing parallelisation, or using GPUs instead of CPUs, only results in marginal reductions in runtime at the expense of increased carbon footprint. Making this information available

also enables future users to compare different tools, estimate the carbon footprint of a project ahead of the analysis, and include it in their cost-benefit analysis to make better-informed choices; we talk more about this in Rule 2.

Rule 9: Be aware of unanticipated consequences of improved software efficiency

We have witnessed incredible gains in speed of softwares over the last two decades, e.g. for DNA assembly and search. However, surprisingly, increasing the efficiency of a software doesn't always lead to a reduction in computing. When given a faster implementation of an algorithm, a researcher will often imagine what larger questions might be addressed or run more analysis in the same timeframe. This rebound effect describes the situation where an increased efficiency in a process leads to greater increases in usage, in this case increasing the total carbon footprint. So you may need to make a conscious decision to keep efficiency wins without expanding the scope of the compute.

Rule 10: Offset your carbon footprint

Carbon offsetting is complex, with debated benefits and should never replace decreasing GHG emissions in the first place; however, it can mitigate the environmental impact of a project. Offsetting is flexible and can be implemented by a single user or by an institution.

There is a variety of offsetting initiatives investing in tree plantations, fuel-efficient stoves, solar panels, or hydropower generators, for example. Such initiatives, which tend to address both GHG emissions and other issues like food security, can be found on platforms such as Carbon Footprint [159]. Importantly, one should ensure that the projects are legitimate and certified by issuers that satisfy international standards set, for example, by the British Standard Institution. Examples of certified issuers are Gold Standard [160], Verra [161] and the American Carbon Registry [162].

In summary, if one wishes to invest in carbon offsetting, they should first ensure that they have minimised their carbon footprint as much as is feasible and then rely on certified offsetting projects.

Discussion

The Green Algorithms framework presented here provides users with a practical way to estimate the carbon footprint of their computations. The method focuses on producing sensible estimates with small overheads for scientists wishing to measure the footprint of their work. Consequently, the online calculator (www.green-algorithms.org) is simple to use, generalisable to nearly any computational task and may be used as a foundation for other aspects of green computing. I applied the Green Algorithms calculator to a variety of tasks, including particle physics simulations, weather forecasting and natural language processing, to estimate their relative and ongoing carbon emissions. Real-world changes to computational infrastructures, such as moving data centres, was also quantifiable in terms of carbon footprint and was shown to be of substantive importance, e.g. moving data centres may attain a more efficient PUE, but a difference in carbon intensity may negate any efficiency gains, potentially making such a move detrimental to the environment. We also estimated the carbon footprint of various bioinformatic algorithms and investigated how memory over-allocation, processor choice and parallelisation affect carbon footprints, and showed the impact of transferring computations to cloud facilities. Finally, we laid out ten simple rules that scientists can follow to make their computation more environmentally sustainable. I discussed how to estimate and reduce one's computational carbon footprint, the environmental impact of electronic hardware, and carbon offsetting. I have also detailed steps computational researchers and software developers can follow and prioritise to reduce carbon footprints.

This work substantially enhances and extends prior frameworks for estimating the carbon footprint of computation. In particular, I have integrated and formalised previously unclear factors such as usage factor and unitary power draw (per core or per GB of memory). As a result, the carbon footprint of an algorithm can be broken down to a small number of key, easily quantifiable elements, such as number of cores, memory size and usage factor. This reduces the burden on the user, who is not required to either measure the power draw of hardware manually or use a limited range of cloud providers for their computations. This makes the method highly flexible in comparison to previous work. Besides drawing attention to the growing issue of GHG emissions of data centres, one of the benefits of

presenting a detailed open methodology and tool is to provide users with the information they need to reduce their carbon footprint. Finally, one of the most important challenges in green computing is to make the estimation and reporting of GHG emissions a standard practice; a transparent, accessible and open-source methodology like the one presented here is crucial to reach this goal. As a research team, we have started to include such acknowledgments in publications [163], [164], and we hope more teams will follow.

The Green Algorithms estimation method has a number of limitations. First, the carbon footprint estimated is restricted to GHGs emitted to power computers during a particular task. I do not perform a Life Cycle Assessment (LCA) and, therefore, do not consider the full environmental and social impact of manufacturing, maintaining and disposing of the hardware used, or the maintenance of the power plants. Including these is impractical at scale and would greatly reduce who can use the method. Besides, the conversion of the impact of various GHG into CO_2e is commonly based on a 100-year timescale; however, this is now debated as it can misrepresent the impact of short-lived climate pollutants like methane [165] and new standards may be needed in the future. Second, the TDP may substantially underestimate power draw in some situations. For example, when hyperthreading is used, the real power consumption can be double the indicated TDP [166]. Depending on how cores on a same processor are allocated to users, the linear relationship between number of cores and power usage (represented by the TDP-per-core) may sometimes not stand. The TDP value remains a sensible estimate of the base consumption of the processor in most situations, but users using hyperthreading should be aware of the impact on power consumption. Third, while the power consumption from storage is usually minimal at the scale of one computation, if central storage is constantly queried by the algorithm (e.g. to avoid overloading memory), this can be an important factor in power draw; however, there are resources which can be utilised if the algorithm is designed to be heavily storage reliant [25]. Moreover, at the scale of the data centre, storage represents a significant part of electricity usage [25], and research projects relying on large databases should separately acknowledge the long-term carbon footprint of storage. Regarding the memory, with some recent and/or future designs, it is possible that the power usage will start to depend on the workload, which will require revisiting how memory is included in this model. Fourth, while some averaging is necessary, the energy

mix of a country varies by the hour. For example, the carbon intensity of South Australia, which relies on wind and gas to produce electricity [167], can vary between 112 and 592 gCO_2e/kWh within one day, depending on the quantity of coal-produced electricity imported from the neighbouring state of Victoria [35]. Although most regions are relatively stable, these outliers may require a finer estimation. The online calculator uses averaged values sourced from government reports [36]. Fifth, the PUE has some limitations as a measure of a data centre's energy usage due to inconsistencies in ways to calculate it [168], [169]. For example, reporting of PUE is highly variable from yearly averages to best-case scenarios, e.g. in winter when minimal cooling is required (as demonstrated by Google's quarterly results [33]). Whether to include infrastructure components such as security or on-site power generation is also source of discrepancies between data centres [32]. Although some companies present well-justified results, many PUEs have no or insufficient justification. Furthermore, PUE is not defined when computations are run on a laptop or desktop computer. As the device is used for multiple tasks simultaneously, it is impossible to estimate the power overhead due to the algorithm. In the calculator, we use a PUE of 1 because of the lack of information, but we caution this should not be interpreted as a sign of efficiency. Even though discrepancies will remain, the widespread adoption of an accurate, transparent and certified estimation of PUE, such as the ISO/IEC standard [170], would be a substantial step for the computing community.

The various carbon footprints estimated also come with some caveats. They are based on manual curation of the literature, and it is generally the case that at least some parameters needed to estimate the carbon footprint are missing from published articles, e.g. runtime, hardware information, or software versions. In this case, we made some assumptions (e.g. 100% core usage and using average TDP) that can explain the differences between our estimates and the real emissions. Besides, many of the bioinformatic carbon footprints estimated are for a single run of any given tool; however, most algorithms have parameters that must be fine-tuned through trial and error, frequently extensively so. For example, in GWAS, various adjustments are made to the initial association analysis to reduce non-biological variation, such as different phenotype normalisations, batch-effect correction, and ancestry-effect adjustments. Each of these adjustments multiplies the analysis' total carbon footprint, and the real GHG emissions are likely to be orders of magnitude greater

than reported here. There are other areas of computational biology, such as imaging or artificial intelligence analyses, that are not estimated here but are likely to have substantial carbon footprints. Similarly, there are a number of other popular bioinformatics algorithms that have not been estimated within this study, examples include BLAST [171], GROMACS [172] and GATK [173]. If we are to fully understand the carbon footprint of the field of bioinformatics, or any computational research, it is crucial that this information is reported systematically (processor runtime, memory usage, hardware and software information) and that authors estimate their own carbon footprint using reliable tools. They can then acknowledge it in publications, as in [163] and [164], notwithstanding that it can then become an argument to support a newly released tool, as in [174].

It is quite surprising that, given the data richness of computing science, it is still difficult to come up with accurate estimates of the environmental impact of computing. This is in part due to the diversity of computing platforms available but also to the lack of reporting standards for hardware and software. I therefore hope that this tool and metrics raise awareness of these issues as well as facilitate pragmatic solutions which may help to mitigate the environmental consequences of modern computation. Coupling the substantial carbon footprints presented in this chapter with the growing global demand for computation warrants that we must both individually and collectively do our utmost to make our computations more environmentally friendly. That will require that our community invests time into creating new tools to understand the environmental impact of our work as well as promotes software (if not hardware) design practices that prioritise GHG minimisation. Overall, with the right tools and practices, I believe HPC and cloud computing can be immensely positive forces for both improving the human condition and saving the environment.

References

[1] R. B. Alley, K. A. Emanuel, and F. Zhang, 'Advances in weather prediction', *Science*, vol. 363, no. 6425, pp. 342–344, Jan. 2019, doi: 10.1126/science.aav7274.

[2] T. E. H. T. Collaboration *et al.*, 'First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole', *ApJL*, vol. 875, no. 1, p. L1, Apr. 2019, doi: 10.3847/2041-8213/ab0ec7.

[3] T. E. H. T. Collaboration *et al.*, 'First M87 Event Horizon Telescope Results. III. Data Processing and Calibration', *ApJL*, vol. 875, no. 1, p. L3, Apr. 2019, doi: 10.3847/2041-8213/ab0c57.

[4] T. E. H. T. Collaboration *et al.*, 'First M87 Event Horizon Telescope Results. IV. Imaging the Central Supermassive Black Hole', *ApJL*, vol. 875, no. 1, p. L4, Apr. 2019, doi: 10.3847/2041-8213/ab0e85.

[5] A. Buniello *et al.*, 'The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019', *Nucleic Acids Res*, vol. 47, no. D1, pp. D1005–D1012, Jan. 2019, doi: 10.1093/nar/gky1120.

[6] D. Ben-Israel *et al.*, 'The impact of machine learning on patient care: A systematic review', *Artificial Intelligence in Medicine*, vol. 103, p. 101785, Mar. 2020, doi: 10.1016/j.artmed.2019.101785.

[7] D. M. Dimiduk, E. A. Holm, and S. R. Niezgoda, 'Perspectives on the Impact of Machine Learning, Deep Learning, and Artificial Intelligence on Materials, Processes, and Structures Engineering', *Integr Mater Manuf Innov*, vol. 7, no. 3, pp. 157–172, Sep. 2018, doi: 10.1007/s40192-018-0117-8.

[8] G. Choy *et al.*, 'Current Applications and Future Impact of Machine Learning in Radiology', *Radiology*, vol. 288, no. 2, pp. 318–328, Jun. 2018, doi: 10.1148/radiol.2018171820.

[9] S. Athey, 'The Impact of Machine Learning on Economics', *The Economics of Artificial Intelligence: An Agenda*, pp. 507–547, Jan. 2018.

[10] E. Strubell, A. Ganesh, and A. McCallum, 'Energy and Policy Considerations for Deep Learning in NLP', *arXiv:1906.02243 [cs]*, Jun. 2019, Accessed: Jan. 01, 2020. [Online]. Available: http://arxiv.org/abs/1906.02243

[11] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, 'Quantifying the Carbon Emissions of Machine Learning', *arXiv:1910.09700 [cs]*, Nov. 2019, Accessed: Jan. 07, 2020. [Online]. Available: http://arxiv.org/abs/1910.09700

[12] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, 'Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning', *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020.

[13] B. Whitehead, D. Andrews, A. Shah, and G. Maidment, 'Assessing the environmental impact of data centres part 1: Background, energy use and metrics', *Building and Environment*, vol. 82, Dec. 2014, doi: 10.1016/j.buildenv.2014.08.021.

[14] L. Lannelongue, J. Grealey, and M. Inouye, 'Green Algorithms: Quantifying the Carbon Footprint of Computation', *Advanced Science*, vol. 8, no. 12, p. 2100707, 2021, doi: 10.1002/advs.202100707.

[15] L. Lannelongue, J. Grealey, A. Bateman, and M. Inouye, 'Ten simple rules to make your computing more environmentally sustainable', *PLoS Comput Biol*, vol. 17, no. 9, p. e1009324, Sep. 2021, doi: 10.1371/journal.pcbi.1009324.

[16] J. Grealey *et al.*, 'The Carbon Footprint of Bioinformatics', *Molecular Biology and Evolution*, p. msac034, Feb. 2022, doi: 10.1093/molbev/msac034.

[17] R. K. Pachauri *et al.*, *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Geneva, Switzerland: IPCC, 2014, p. 151. Accessed: May 26, 2020. [Online]. Available: https://epic.awi.de/id/eprint/37530/

[18] N. Hill *et al.*, '2020 Government greenhouse gas conversion factors for company reporting: Methodology paper', *Department for Business, Energy and Industrial Strategy*, p. 128, Jul. 2020.

[19] O. US EPA, 'Understanding Global Warming Potentials', Jan. 12, 2016. https://www.epa.gov/ghgemissions/understanding-global-warming-potentials (accessed Mar. 02, 2022).

[20] L. A. Wright, S. Kemp, and I. Williams, '"Carbon footprinting": towards a universally accepted definition', *Carbon Management*, vol. 2, no. 1, pp. 61–72, Feb. 2011, doi: 10.4155/cmt.10.39.

[21] 'Kyoto Protocol To The United Nations Framework Convention On Climate Change', Dec. 1997.

[22] H. Ritchie and M. Roser, 'CO_2 and Greenhouse Gas Emissions', *Our World in Data*, 2020, Accessed: Jun. 13, 2020. [Online]. Available: https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions

[23] A. Karyakin and K. Salem, 'An analysis of memory power consumption in database systems', in *Proceedings of the 13th International Workshop on Data Management on New Hardware - DAMON '17*, Chicago, Illinois, 2017, pp. 1–9. doi: 10.1145/3076113.3076117.

[24] C. Angelini, I. W. August 29, and 2014, 'Intel Core i7-5960X, -5930K And -5820K CPU Review: Haswell-E Rises', *Tom's Hardware*. https://www.tomshardware.com/uk/reviews/intel-core-i7-5960x-haswell-e-cpu,3918-13.html (accessed Jan. 31, 2020).

[25] E. Tomes and N. Altiparmak, 'A Comparative Study of HDD and SSD RAIDs' Impact on Server Energy Consumption', in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2017, pp. 625–626. doi: 10.1109/CLUSTER.2017.103.

[26] A. Shehabi *et al.*, 'United States Data Center Energy Usage Report', 2016.

[27] 'Seagate Product Sustainability | Seagate UK'. https://www.seagate.com/gb/en/global-citizenship/product-sustainability/ (accessed Mar. 03, 2022).

[28] B. Nguyen *et al.*, 'Architecting Datacenters for Sustainability: Greener Data Storage using Synthetic DNA', presented at the Electronics Goes Green 2020, Sep. 2020. Accessed: Jul. 13, 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/architecting-datacenters-for-sustainability-greener-data-storage-using-synthetic-dna/

[29] H. Geng, *Data Center Handbook*. John Wiley & Sons, 2014.

[30] Andy Lawrence, 'Is PUE actually going UP?', *Uptime Institute Blog*, May 15, 2019. https://journal.uptimeinstitute.com/is-pue-actually-going-up/ (accessed Apr. 14, 2020).

[31] C. L. Belady and C. G. Malone, 'Metrics and an Infrastructure Model to Evaluate Data Center Efficiency', Jan. 2010, pp. 751–755. doi: 10.1115/IPACK2007-33338.

[32] V. Avelar, D. Azevedo, A. French, and E. N. Power, 'PUE: a comprehensive examination of the metric', *White paper*, vol. 49, 2012, [Online]. Available: https://datacenters.lbl.gov/sites/all/files/WP49-PUE%20A%20Comprehensive%20Examination%20of%20the%20Metric_v6.pdf

[33] 'Efficiency – Data Centers – Google', *Google Data Centers*. https://www.google.com/about/datacenters/efficiency/ (accessed Jan. 22, 2020).

[34] J. Gao, *Machine Learning Applications for Data Center Optimization*. 2014.

[35] B. Tranberg, O. Corradi, B. Lajoie, T. Gibon, I. Staffell, and G. B. Andresen, 'Real-time carbon accounting method for the European electricity markets', *Energy Strategy Reviews*, vol. 26, p. 100367, Nov. 2019, doi: 10.1016/j.esr.2019.100367.

[36] 'carbonfootprint.com - International Electricity Factors'. https://www.carbonfootprint.com/international_electricity_factors.html (accessed Feb. 04, 2020).

[37] C. Fletcher *et al.*, 'Carbon impact of video streaming', Carbon Trust, Jun. 2021. [Online]. Available: https://www.carbontrust.com/resources/carbon-impact-of-video-streaming

[38] 'Carbon emissions of richest 1% set to be 30 times the 1.5°C limit in 2030', *Oxfam International*, Nov. 05, 2021. https://www.oxfam.org/en/press-releases/carbon-emissions-richest-1-set-be-30-times-15degc-limit-2030 (accessed Mar. 03, 2022).

[39] E. Helmers, J. Leitão, U. Tietge, and T. Butler, 'CO2-equivalent emissions from European passenger vehicles in the years 1995–2015 based on real-world use: Assessing the climate benefit of the European "diesel boom"', *Atmospheric Environment*, vol. 198, pp. 122–132, Feb. 2019, doi: 10.1016/j.atmosenv.2018.10.039.

[40] O. US EPA, 'Greenhouse Gas Emissions from a Typical Passenger Vehicle', *US EPA*, Jan. 12, 2016. https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle (accessed Jan. 31, 2020).

[41] 'Carbon Footprint Calculator'. https://calculator.carbonfootprint.com/calculator.aspx?tab=3 (accessed Jun. 18, 2020).

[42] H. Ritchie and M. Roser, 'Greenhouse gas emissions', *Our World in Data*. https://ourworldindata.org/greenhouse-gas-emissions (accessed Nov. 30, 2020).

[43] H. Akbari, 'Shade trees reduce building energy use and CO2 emissions from power plants', *Environmental Pollution*, vol. 116, pp. S119–S126, Mar. 2002, doi: 10.1016/S0269-7491(01)00264-0.

[44] 'Dash Documentation & User Guide | Plotly'. https://dash.plotly.com/ (accessed Mar. 03, 2022).

[45] 'Documentation | Heroku Dev Center'. https://devcenter.heroku.com/categories/reference (accessed Mar. 03, 2022).

[46] D. Patterson *et al.*, 'Carbon Emissions and Large Neural Network Training', *arXiv:2104.10350 [cs]*, Apr. 2021, Accessed: Mar. 04, 2022. [Online]. Available: http://arxiv.org/abs/2104.10350

[47] P. Teich, 'Tearing Apart Google's TPU 3.0 AI Coprocessor', *The Next Platform*, May 10, 2018. https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/ (accessed Apr. 26, 2020).

[48] A. B. Yoo, M. A. Jette, and M. Grondona, 'SLURM: Simple Linux Utility for Resource Management', in *Job Scheduling Strategies for Parallel Processing*, vol. 2862, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60. doi: 10.1007/10968987_3.

[49] 'Emissions – Global Energy & CO2 Status Report 2019 – Analysis', *IEA*. https://www.iea.org/reports/global-energy-co2-status-report-2019/emissions (accessed Feb. 10, 2020).

[50] S. Agostinelli *et al.*, 'Geant4—a simulation toolkit', *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, Jul. 2003, doi: 10.1016/S0168-9002(03)01368-8.

[51] S. Jan *et al.*, 'GATE - Geant4 Application for Tomographic Emission: a simulation toolkit for PET and SPECT', *Phys Med Biol*, vol. 49, no. 19, pp. 4543–4561, Oct. 2004.

[52] S. Jan *et al.*, 'GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy', *Phys Med Biol*, vol. 56, no. 4, pp. 881–901, Feb. 2011, doi: 10.1088/0031-9155/56/4/001.

[53] D. Sarrut *et al.*, 'A review of the use and potential of the GATE Monte Carlo simulation code for radiation therapy and dosimetry applications', *Med Phys*, vol. 41, no. 6, p. 064301, Jun. 2014, doi: 10.1118/1.4871617.

[54] S. Meylan *et al.*, 'Simulation of early DNA damage after the irradiation of a fibroblast cell nucleus using Geant4-DNA', *Sci Rep*, vol. 7, Sep. 2017, doi: 10.1038/s41598-017-11851-4.

[55] P. Schweitzer *et al.*, 'Performance Evaluation of Multithreaded Geant4 Simulations Using an Intel Xeon Phi Cluster', *Scientific Programming*, 2015. https://www.hindawi.com/journals/sp/2015/980752/ (accessed Apr. 23, 2020).

[56] J. Apostolakis *et al.*, 'Validation and verification of Geant4 standard electromagnetic physics', *J. Phys.: Conf. Ser.*, vol. 219, no. 3, p. 032044, Apr. 2010, doi: 10.1088/1742-6596/219/3/032044.

[57] ‘Geant4 - an Object-Oriented Toolkit for Simulation in HEP’, *GitLab*. https://gitlab.cern.ch/geant4/geant4/tree/edb408b5618b3b1cd3f40c5759aa5da4aa56bb7b/examples/extended/electromagnetic/TestEm5 (accessed Jun. 25, 2020).

[58] P. Neumann *et al.*, ‘Assessing the scales in numerical weather and climate predictions: will exascale be the rescue?’, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 377, no. 2142, p. 20180148, Apr. 2019, doi: 10.1098/rsta.2018.0148.

[59] S. Keeley, ‘Modelling and Prediction: Integrated Forecast System’, *ECMWF*, Nov. 29, 2013. https://www.ecmwf.int/en/research/modelling-and-prediction (accessed Apr. 23, 2020).

[60] G. Zängl, D. Reinert, P. Rípodas, and M. Baldauf, ‘The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core’, *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 687, pp. 563–579, 2015, doi: 10.1002/qj.2378.

[61] ‘Wetter und Klima - Deutscher Wetterdienst - Numerical weather prediction models - ICON (Icosahedral Nonhydrostatic) Model’. https://www.dwd.de/EN/research/weatherforecasting/num_modelling/01_num_weather_prediction_modells/icon_description.html (accessed Jun. 18, 2020).

[62] S. Baylis, ‘Green computing’, 2011, doi: 10.21957/CEMUB9F8.

[63] ‘Wetter und Klima - Deutscher Wetterdienst - Information technology - Data processing, DMRZ’. https://www.dwd.de/EN/aboutus/it/functions/Teasergroup/dataprocessing.html?nn=24864 (accessed Jun. 19, 2020).

[64] ‘Footnote horizontal resolution’, *The horizontal resolution represents the level of geographical detail achieved when modelling the different weather phenomenon.*.

[65] J. A. P. León, ‘Progress in using single precision in the IFS’, *ECMWF*, Oct. 16, 2018. https://www.ecmwf.int/en/newsletter/157/meteorology/progress-using-single-precision-ifs (accessed Jun. 19, 2020).

[66] G. Lentze, ‘ECMWF signs contract with Atos for new supercomputer’, *ECMWF*, Jan. 10, 2020. https://www.ecmwf.int/en/about/media-centre/news/2020/ecmwf-signs-contract-atos-new-supercomputer (accessed Apr. 23, 2020).

[67] J. Jeppesen, ‘ECMWF becomes a multi-site organisation’, *ECMWF*, Sep. 11, 2021. https://www.ecmwf.int/en/about/media-centre/news/2021/ecmwf-becomes-multi-site-organisation (accessed Mar. 04, 2022).

[68] ECMWF, ‘ECMWF Q&A’. [Online]. Available: https://www.ecmwf.int/sites/default/files/medialibrary/2019-02/Industry_day_-_QA.pdf

[69] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, ‘Green AI’, *arXiv:1907.10597 [cs, stat]*, Aug. 2019, Accessed: Jan. 07, 2020. [Online]. Available: http://arxiv.org/abs/1907.10597

[70] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *arXiv:1810.04805 [cs]*, May 2019, Accessed: Apr. 24, 2020. [Online]. Available: http://arxiv.org/abs/1810.04805

[71] I. Beltagy, K. Lo, and A. Cohan, ‘SciBERT: A Pretrained Language Model for Scientific Text’, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 3615–3620. doi: 10.18653/v1/D19-1371.

[72] J. Lee *et al.*, ‘BioBERT: a pre-trained biomedical language representation model for biomedical text mining’, *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020, doi: 10.1093/bioinformatics/btz682.

[73] NVIDIA AI, ‘BERT Meets GPUs – Future Vision – Medium’, May 21, 2019. https://web.archive.org/web/20190521104957/https://medium.com/future-vision/bert-meets-gpus-403d3fbed848 (accessed Apr. 24, 2020).

[74] D. Adiwardana *et al.*, 'Towards a Human-like Open-Domain Chatbot', *arXiv:2001.09977 [cs, stat]*, Feb. 2020, Accessed: Apr. 26, 2020. [Online]. Available: http://arxiv.org/abs/2001.09977

[75] L. Kachuri *et al.*, 'Pan-cancer analysis demonstrates that integrating polygenic risk scores with modifiable risk factors improves risk prediction', Genetics, preprint, Jan. 2020. doi: 10.1101/2020.01.28.922088.

[76] The ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Consortium, 'Pan-cancer analysis of whole genomes', *Nature*, vol. 578, no. 7793, pp. 82–93, Feb. 2020, doi: 10.1038/s41586-020-1969-6.

[77] PCAWG Structural Variation Working Group *et al.*, 'Patterns of somatic structural variation in human cancer genomes', *Nature*, vol. 578, no. 7793, pp. 112–121, Feb. 2020, doi: 10.1038/s41586-019-1913-9.

[78] The Severe Covid-19 GWAS Group, 'Genomewide Association Study of Severe Covid-19 with Respiratory Failure', *New England Journal of Medicine*, vol. 383, no. 16, pp. 1522–1534, Oct. 2020, doi: 10.1056/NEJMoa2020283.

[79] C. Bycroft *et al.*, 'The UK Biobank resource with deep phenotyping and genomic data', *Nature*, vol. 562, no. 7726, Art. no. 7726, Oct. 2018, doi: 10.1038/s41586-018-0579-z.

[80] 'National Institutes of Health (NIH) — All of Us'. https://allofus.nih.gov/ (accessed Oct. 27, 2020).

[81] 'Accelerating Detection of Disease - UK Research and Innovation'. https://www.ukri.org/innovation/industrial-strategy-challenge-fund/accelerating-detection-of-disease/ (accessed Oct. 27, 2020).

[82] M. Hunt, C. Newbold, M. Berriman, and T. D. Otto, 'A comprehensive evaluation of assembly scaffolding tools', *Genome Biology*, vol. 15, no. 3, p. R42, Mar. 2014, doi: 10.1186/gb-2014-15-3-r42.

[83] S. D. Jackman *et al.*, 'ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter', *Genome Res*, vol. 27, no. 5, pp. 768–777, May 2017, doi: 10.1101/gr.214346.116.

[84] J. Vollmers, S. Wiegand, and A.-K. Kaster, 'Comparing and Evaluating Metagenome Assembly Tools from a Microbiologist's Perspective - Not Only Size Matters!', *PLOS ONE*, vol. 12, no. 1, p. e0169662, Jan. 2017, doi: 10.1371/journal.pone.0169662.

[85] A. T. Dilthey, C. Jain, S. Koren, and A. M. Phillippy, 'Strain-level metagenomic assignment and compositional estimation for long reads with MetaMaps', *Nature Communications*, vol. 10, no. 1, Art. no. 1, Jul. 2019, doi: 10.1038/s41467-019-10934-2.

[86] G. Baele, D. L. Ayres, A. Rambaut, M. A. Suchard, and P. Lemey, 'High-Performance Computing in Bayesian Phylogenetics and Phylodynamics Using BEAGLE', in *Evolutionary Genomics: Statistical and Computational Methods*, M. Anisimova, Ed. New York, NY: Springer, 2019, pp. 691–722. doi: 10.1007/978-1-4939-9074-0_23.

[87] X. Zhou, X.-X. Shen, C. T. Hittinger, and A. Rokas, 'Evaluating Fast Maximum Likelihood-Based Phylogenetic Programs Using Empirical Phylogenomic Data Sets', *Molecular Biology and Evolution*, vol. 35, no. 2, pp. 486–503, Feb. 2018, doi: 10.1093/molbev/msx302.

[88] E. D. Jarvis *et al.*, 'Whole-genome analyses resolve early branches in the tree of life of modern birds', *Science*, vol. 346, no. 6215, pp. 1320–1331, Dec. 2014, doi: 10.1126/science.1253451.

[89] G. Baruzzo, K. E. Hayer, E. J. Kim, B. Di Camillo, G. A. FitzGerald, and G. R. Grant, 'Simulation-based comprehensive benchmarking of RNA-seq aligners', *Nature Methods*, vol. 14, no. 2, Art. no. 2, Feb. 2017, doi: 10.1038/nmeth.4106.

[90] A. Kanitz, F. Gypas, A. J. Gruber, A. R. Gruber, G. Martin, and M. Zavolan, 'Comparative assessment of methods for the computational inference of transcript isoform abundance from RNA-seq data', *Genome Biol*, vol. 16, no. 1, 2015, doi: 10.1186/s13059-015-0702-5.

[91] P.-R. Loh, G. Kichaev, S. Gazal, A. P. Schoech, and A. L. Price, 'Mixed-model association for biobank-scale datasets', *Nat Genet*, vol. 50, no. 7, pp. 906–908, Jul. 2018, doi: 10.1038/s41588-018-0144-6.

[92] A. Taylor-Weiner *et al.*, 'Scaling computational genomics to millions of individuals with GPUs', *Genome Biology*, vol. 20, no. 1, p. 228, Nov. 2019, doi: 10.1186/s13059-019-1836-7.

[93] 'NAMD Performance'. https://www.ks.uiuc.edu/Research/namd/benchmarks/ (accessed Jul. 25, 2020).

[94] 'The pmemd.cuda GPU Implementation'. https://ambermd.org/GPUPerformance.php (accessed Jul. 23, 2020).

[95] S. Ruiz-Carmona *et al.*, 'rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids', *PLoS Computational Biology*, vol. 10, no. 4, p. e1003571, Apr. 2014, doi: 10.1371/journal.pcbi.1003571.

[96] D. E. Wood, J. Lu, and B. Langmead, 'Improved metagenomic analysis with Kraken 2', *Genome Biology*, vol. 20, no. 1, p. 257, Nov. 2019, doi: 10.1186/s13059-019-1891-0.

[97] D. E. Wood and S. L. Salzberg, 'Kraken: ultrafast metagenomic sequence classification using exact alignments', *Genome Biology*, vol. 15, no. 3, p. R46, Mar. 2014, doi: 10.1186/gb-2014-15-3-r46.

[98] J. Lu, F. P. Breitwieser, P. Thielen, and S. L. Salzberg, 'Bracken: estimating species abundance in metagenomics data', *PeerJ Comput. Sci.*, vol. 3, p. e104, Jan. 2017, doi: 10.7717/peerj-cs.104.

[99] D. Kim, L. Song, F. P. Breitwieser, and S. L. Salzberg, 'Centrifuge: rapid and sensitive classification of metagenomic sequences', *Genome Res.*, vol. 26, no. 12, pp. 1721–1729, Jan. 2016, doi: 10.1101/gr.210641.116.

[100] 'BOLT-LMM v2.3.4 User Manual'. https://data.broadinstitute.org/alkesgroup/BOLT-LMM/#x1-150003.2 (accessed Jul. 23, 2020).

[101] 'Genetic effects on gene expression across human tissues', *Nature*, vol. 550, no. 7675, pp. 204–213, Oct. 2017, doi: 10.1038/nature24277.

[102] H. Ongen, A. Buil, A. A. Brown, E. T. Dermitzakis, and O. Delaneau, 'Fast and efficient QTL mapper for thousands of molecular phenotypes', *Bioinformatics*, vol. 32, no. 10, pp. 1479–1485, May 2016, doi: 10.1093/bioinformatics/btv722.

[103] *broadinstitute/tensorqtl*. Broad Institute, 2020. Accessed: Nov. 16, 2020. [Online]. Available: https://github.com/broadinstitute/tensorqtl

[104] C. Lippert, F. P. Casale, B. Rakitsch, and O. Stegle, 'LIMIX: genetic analysis of multiple traits', Genetics, preprint, May 2014. doi: 10.1101/003905.

[105] D. A. Case *et al.*, 'The Amber biomolecular simulation programs', *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1668–1688, 2005, doi: 10.1002/jcc.20290.

[106] J. C. Phillips *et al.*, 'Scalable Molecular Dynamics with NAMD', *J Comput Chem*, vol. 26, no. 16, pp. 1781–1802, Dec. 2005, doi: 10.1002/jcc.20289.

[107] M. Boetzer, C. V. Henkel, H. J. Jansen, D. Butler, and W. Pirovano, 'Scaffolding pre-assembled contigs using SSPACE', *Bioinformatics*, vol. 27, no. 4, pp. 578–579, Feb. 2011, doi: 10.1093/bioinformatics/btq683.

[108] J. T. Simpson and R. Durbin, 'Efficient de novo assembly of large genomes using compressed data structures', *Genome Res.*, vol. 22, no. 3, pp. 549–556, Mar. 2012, doi: 10.1101/gr.126953.111.

[109] R. Luo *et al.*, 'SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler', *GigaScience*, vol. 1, no. 1, Dec. 2012, doi: 10.1186/2047-217X-1-18.

[110] D. R. Zerbino and E. Birney, 'Velvet: algorithms for de novo short read assembly using de Bruijn graphs', *Genome Res*, vol. 18, no. 5, pp. 821–829, May 2008, doi: 10.1101/gr.074492.107.

[111] S. L. Salzberg *et al.*, 'GAGE: A critical evaluation of genome assemblies and assembly algorithms', *Genome Res.*, vol. 22, no. 3, pp. 557–567, Jan. 2012, doi: 10.1101/gr.131383.111.

[112] T. D. S. Sutton, A. G. Clooney, F. J. Ryan, R. P. Ross, and C. Hill, 'Choice of assembly software has a critical impact on virome characterisation', *Microbiome*, vol. 7, no. 1, Dec. 2019, doi: 10.1186/s40168-019-0626-5.

[113] D. Li *et al.*, 'MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices', *Methods*, vol. 102, pp. 3–11, 01 2016, doi: 10.1016/j.ymeth.2016.02.020.

[114] K. R. Bradnam *et al.*, 'Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species', *GigaSci*, vol. 2, no. 1, p. 10, Dec. 2013, doi: 10.1186/2047-217X-2-10.

[115] H. E. L. Lischer and K. K. Shimizu, 'Reference-guided de novo assembly approach improves genome reconstruction for related species', *BMC Bioinformatics*, vol. 18, no. 1, p. 474, Dec. 2017, doi: 10.1186/s12859-017-1911-6.

[116] S. Nurk, D. Meleshko, A. Korobeynikov, and P. Pevzner, 'metaSPAdes: a new versatile de novo metagenomics assembler', *arXiv:1604.03071 [q-bio]*, Aug. 2016, Accessed: Oct. 28, 2020. [Online]. Available: http://arxiv.org/abs/1604.03071

[117] T. Namiki, T. Hachiya, H. Tanaka, and Y. Sakakibara, 'MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads', *Nucleic Acids Res*, vol. 40, no. 20, p. e155, Nov. 2012, doi: 10.1093/nar/gks678.

[118] G. Dudas *et al.*, 'Virus genomes reveal factors that spread and sustained the Ebola epidemic', *Nature*, vol. 544, no. 7650, pp. 309–315, 20 2017, doi: 10.1038/nature22040.

[119] N. Goldman and Z. Yang, 'A codon-based model of nucleotide substitution for protein-coding DNA sequences.', *Molecular Biology and Evolution*, vol. 11, no. 5, pp. 725–736, Sep. 1994, doi: 10.1093/oxfordjournals.molbev.a040153.

[120] A. J. Drummond, M. A. Suchard, D. Xie, and A. Rambaut, 'Bayesian phylogenetics with BEAUti and the BEAST 1.7', *Mol Biol Evol*, vol. 29, no. 8, pp. 1969–1973, Aug. 2012, doi: 10.1093/molbev/mss075.

[121] D. L. Ayres *et al.*, 'BEAGLE: An Application Programming Interface and High-Performance Computing Library for Statistical Phylogenetics', *Syst Biol*, vol. 61, no. 1, pp. 170–173, Jan. 2012, doi: 10.1093/sysbio/syr100.

[122] H. A. Ogilvie, J. Heled, D. Xie, and A. J. Drummond, 'Computational Performance and Statistical Accuracy of *BEAST and Comparisons with Other Methods', *Systematic Biology*, vol. 65, no. 3, pp. 381–396, May 2016, doi: 10.1093/sysbio/syv118.

[123] A. Stamatakis, 'RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies', *Bioinformatics*, vol. 30, no. 9, pp. 1312–1313, May 2014, doi: 10.1093/bioinformatics/btu033.

[124] A. M. Kozlov, A. J. Aberer, and A. Stamatakis, 'ExaML version 3: a tool for phylogenomic analyses on supercomputers', *Bioinformatics*, vol. 31, no. 15, pp. 2577–2579, Aug. 2015, doi: 10.1093/bioinformatics/btv184.

[125] S. Guindon and O. Gascuel, 'A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood', *Systematic Biology*, vol. 52, no. 5, pp. 696–704, Oct. 2003, doi: 10.1080/10635150390235520.

[126] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, 'New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0', *Systematic Biology*, vol. 59, no. 3, pp. 307–321, Mar. 2010, doi: 10.1093/sysbio/syq010.

[127] L.-T. Nguyen, H. A. Schmidt, A. von Haeseler, and B. Q. Minh, 'IQ-TREE: A Fast and Effective Stochastic Algorithm for Estimating Maximum-Likelihood Phylogenies', *Molecular Biology and Evolution*, vol. 32, no. 1, pp. 268–274, Jan. 2015, doi: 10.1093/molbev/msu300.

[128] M. N. Price, P. S. Dehal, and A. P. Arkin, 'FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments', *PLoS ONE*, vol. 5, no. 3, p. e9490, Mar. 2010, doi: 10.1371/journal.pone.0009490.

[129] A. Dobin *et al.*, 'STAR: ultrafast universal RNA-seq aligner', *Bioinformatics*, vol. 29, no. 1, pp. 15–21, Jan. 2013, doi: 10.1093/bioinformatics/bts635.

[130] D. Kim, J. M. Paggi, C. Park, C. Bennett, and S. L. Salzberg, 'Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype', *Nat Biotechnol*, vol. 37, no. 8, pp. 907–915, Aug. 2019, doi: 10.1038/s41587-019-0201-4.

[131] D. Kim, G. Pertea, C. Trapnell, H. Pimentel, R. Kelley, and S. L. Salzberg, 'TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions', *Genome Biol*, vol. 14, no. 4, p. R36, Apr. 2013, doi: 10.1186/gb-2013-14-4-r36.

[132] 'NovoAlign | Novocraft'. http://www.novocraft.com/products/novoalign/ (accessed Nov. 14, 2020).

[133] M. M. H. Kusel, N. H. de Klerk, P. G. Holt, T. Kebadze, S. L. Johnston, and P. D. Sly, 'Role of Respiratory Viruses in Acute Upper and Lower Respiratory Tract Illness in the First Year of Life: A Birth Cohort Study', *The Pediatric Infectious Disease Journal*, vol. 25, no. 8, pp. 680–686, Aug. 2006, doi: 10.1097/01.inf.0000226912.88900.a3.

[134] M. M. H. Kusel *et al.*, 'Early-life respiratory viral infections, atopic sensitization, and risk of subsequent development of persistent asthma', *Journal of Allergy and Clinical Immunology*, vol. 119, no. 5, pp. 1105–1110, May 2007, doi: 10.1016/j.jaci.2006.12.669.

[135] 'Babraham Bioinformatics - Trim Galore!' https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/ (accessed Jul. 27, 2020).

[136] 'BBMap Guide', *DOE Joint Genome Institute*. https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbmap-guide/ (accessed Jul. 27, 2020).

[137] R. Patro, S. M. Mount, and C. Kingsford, 'Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms', *Nat Biotechnol*, vol. 32, no. 5, pp. 462–464, May 2014, doi: 10.1038/nbt.2862.

[138] B. Li and C. N. Dewey, 'RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome', *BMC Bioinformatics*, vol. 12:323, 2011, doi: 10.1186/1471-2105-12-323.

[139] C. Trapnell *et al.*, 'Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation', *Nat Biotechnol*, vol. 28, no. 5, pp. 511–515, May 2010, doi: 10.1038/nbt.1621.

[140] T. Griebel *et al.*, 'Modelling and simulating generic RNA-Seq experiments with the flux simulator', *Nucleic Acids Res*, vol. 40, no. 20, pp. 10073–10083, Nov. 2012, doi: 10.1093/nar/gks666.

[141] J. Harrow *et al.*, 'GENCODE: the reference human genome annotation for The ENCODE Project', *Genome Res*, vol. 22, no. 9, pp. 1760–1774, Sep. 2012, doi: 10.1101/gr.135350.111.

[142] O. Trott and A. J. Olson, 'AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading', *J Comput Chem*, vol. 31, no. 2, pp. 455–461, Jan. 2010, doi: 10.1002/jcc.21334.

[143] R. A. Friesner *et al.*, 'Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy', *Journal of Medicinal Chemistry*, vol. 47, no. 7, pp. 1739–1749, Mar. 2004, doi: 10.1021/jm0306430.

[144] N. Huang, B. K. Shoichet, and J. J. Irwin, 'Benchmarking Sets for Molecular Docking', *J. Med. Chem.*, vol. 49, no. 23, pp. 6789–6801, Nov. 2006, doi: 10.1021/jm0608356.

[145] A. Shehabi *et al.*, 'United States Data Center Energy Usage Report', LBNL--1005775, 1372902, Jun. 2016. doi: 10.2172/1372902.

[146] 'Efficiency – Data Centers – Google', *Google Data Centers*. https://www.google.com/about/datacenters/efficiency/ (accessed Jul. 27, 2020).

[147] Microsoft, 'Microsoft's Cloud Infrastructure, Datacenters and Network Fact Sheet'. Microsoft Corporation, Jun. 2015. [Online]. Available: http://download.microsoft.com/download/8/2/9/8297f7c7-ae81-4e99-b1db-d65a01f7a8ef/microsoft_cloud_infrastructure_datacenter_and_network_fact_sheet.pdf

[148] 'AWS & Sustainability', *Amazon Web Services, Inc.* https://aws.amazon.com/about-aws/sustainability/ (accessed Jul. 27, 2020).

[149] 'carbonfootprint.com - International Electricity Factors'. https://www.carbonfootprint.com/international_electricity_factors.html (accessed Jan. 21, 2021).

[150] L.-P. P.-V. P. Clément, Q. E. S. Jacquemotte, and L. M. Hilty, 'Sources of variation in life cycle assessments of smartphones and tablet computers', *Environmental Impact Assessment Review*, vol. 84, p. 106416, Sep. 2020, doi: 10.1016/j.eiar.2020.106416.

[151] 'Apple Product Environmental Reports', *Apple Environment*. https://www.apple.com/environment/

[152] D. C. Wilson, C. Velis, and C. Cheeseman, 'Role of informal sector recycling in waste management in developing countries', *Habitat International*, vol. 30, no. 4, pp. 797–808, Dec. 2006, doi: 10.1016/j.habitatint.2005.09.005.

[153] A. Sepúlveda *et al.*, 'A review of the environmental fate and effects of hazardous substances released from electrical and electronic equipments during recycling: Examples from China and India', *Environmental Impact Assessment Review*, vol. 30, no. 1, 2010, doi: 10.1016/j.eiar.2009.04.001.

[154] M. Proske, D. Sánchez, C. Clemm, and S.-J. Baur, 'LIFE CYCLE ASSESSMENT OF THE FAIRPHONE 3', p. 69.

[155] 'Country Specific Electricity Grid Greenhouse Gas Emission Factors', *carbonfootprint.com*, Sep. 2020. https://www.carbonfootprint.com/international_electricity_factors.html

[156] J. Bergstra and Y. Bengio, 'Random search for hyper-parameter optimization', *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 281–305, Feb. 2012.

[157] J. Dodge, S. Gururangan, D. Card, R. Schwartz, and N. A. Smith, 'Show Your Work: Improved Reporting of Experimental Results', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 2185–2194. doi: 10.18653/v1/D19-1224.

[158] R. Schwartz, G. Stanovsky, S. Swayamdipta, J. Dodge, and N. A. Smith, 'The Right Tool for the Job: Matching Model and Instance Complexities', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, 2020, pp. 6640–6651. doi: 10.18653/v1/2020.acl-main.593.

[159] 'carbonfootprint.com - Carbon Offset Projects'. https://www.carbonfootprint.com/carbonoffsetprojects.html (accessed Apr. 22, 2020).

[160] 'The Gold Standard'. https://www.goldstandard.org/ (accessed Apr. 22, 2020).

[161] 'Verra', *Verra*. https://verra.org/ (accessed Apr. 22, 2020).

[162] 'American Carbon Registry'. https://americancarbonregistry.org/ (accessed Apr. 22, 2020).

[163] Y. Qin *et al.*, 'Combined effects of host genetics and diet on human gut microbiota and incident disease in a single population cohort', *Nat Genet*, vol. 54, no. 2, pp. 134–142, Feb. 2022, doi: 10.1038/s41588-021-00991-z.

[164] L. Lannelongue and M. Inouye, 'Construction of in silico protein-protein interaction networks across different topologies using machine learning'. bioRxiv, p. 2022.02.07.479382, Feb. 09, 2022. doi: 10.1101/2022.02.07.479382.

[165] M. R. Allen *et al.*, 'A solution to the misrepresentations of CO 2 -equivalent emissions of short-lived climate pollutants under ambitious mitigation', *npj Climate and Atmospheric Science*, vol. 1, no. 1, Art. no. 1, Jun. 2018, doi: 10.1038/s41612-018-0026-8.

[166] I. Cutress, 'Why Intel Processors Draw More Power Than Expected: TDP and Turbo Explained'. https://www.anandtech.com/show/13544/why-intel-processors-draw-more-power-than-expected-tdp-turbo (accessed Jan. 22, 2020).

[167] 'SA.GOV.AU - SA's electricity supply and market'. https://www.sa.gov.au/topics/energy-and-environment/energy-supply/sas-electricity-supply-and-market (accessed Jul. 13, 2020).

[168] R. C. Zoie, R. D. Mihaela, and S. Alexandru, 'An analysis of the power usage effectiveness metric in data centers', in *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, Oct. 2017, pp. 1–6. doi: 10.1109/ISEEE.2017.8170650.

[169] J. Yuventi and R. Mehdizadeh, 'A critical analysis of Power Usage Effectiveness and its use in communicating data center energy consumption', *Energy and Buildings*, vol. 64, pp. 90–94, Sep. 2013, doi: 10.1016/j.enbuild.2013.04.015.

[170] 'ISO/IEC 30134-2:2016(en), Information technology — Data centres — Key performance indicators — Part 2: Power usage effectiveness (PUE)'. https://www.iso.org/obp/ui/#iso:std:iso-iec:30134:-2:ed-1:v1:en (accessed Nov. 28, 2020).

[171] S. F. AltschuP, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, 'Basic Local Alignment Search Tool', *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990, doi: https://doi.org/10.1016/S0022-2836(05)80360-2.

[172] D. V. D. Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen, 'GROMACS: Fast, flexible, and free', *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1701–1718, 2005, doi: https://doi.org/10.1002/jcc.20291.

[173] A. McKenna *et al.*, 'The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data', *Genome Research*, vol. 20, no. 9, pp. 1297–1303, Sep. 2010, doi: 10.1101/gr.107524.110.

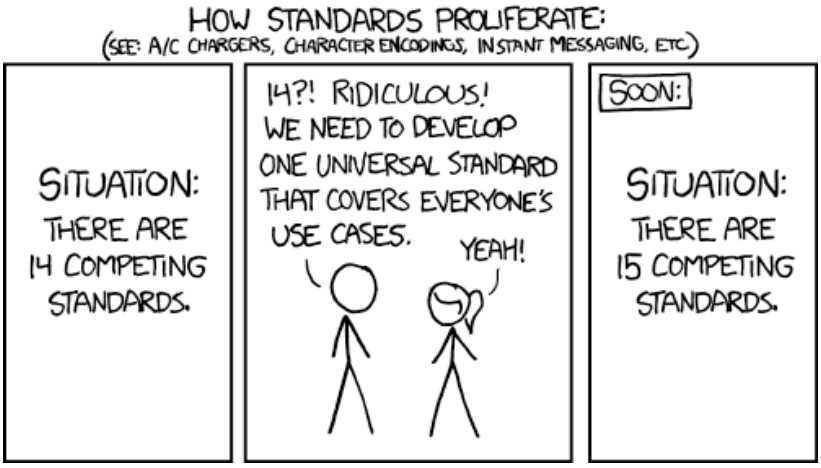
[174] J. Mbatchou *et al.*, 'Computationally efficient whole-genome regression for quantitative and binary traits', *Nat Genet*, vol. 53, no. 7, pp. 1097–1103, Jul. 2021, doi: 10.1038/s41588-021-00870-7.

CONCLUSION

This thesis presents two distinct, yet intertwined, frameworks for computational biology. B4PPI is a collection of open-source datasets and guidelines for standardised, trustworthy and reproducible PPI predictions. Analyses within this framework shed light on the different mechanisms underpinning popular PPI prediction tools and the impact of network topology on these machine learning models. B4PPI also includes appropriate accuracy metrics to consider when benchmarking prediction algorithms, such as ROC and PR curves, as well as efficiency metrics. Often omitted but yet essential, especially in light of the global warming crisis, these metrics quantifying the energy needs of computations help communicate the financial and environmental cost of algorithms to ensure a reasoned use of resources. However, no comprehensive and practical framework existed, so to include carbon footprint in B4PPI, I had to first develop one. The Green Algorithms project comprises both a methodology to estimate the carbon footprint of computations and an online calculator that makes it easy for scientists to do so. Besides including GHG emissions in B4PPI, I also studied a range of analyses in physics and conducted a detailed survey of the field of bioinformatics. Moreover, this work on green computing turned out to have far-reaching implications beyond computational biology; the calculator and guidelines have been used in machine learning, physics, cosmology etc. It also revealed that most scientists wish to be mindful of their carbon footprint but, until now, lacked the tools to do so. To help them, I put together a Ten Simple Rules article summarising the most impactful ways to reduce the environmental impact of our work.

These different projects pave the way for further studies. For example, extending B4PPI to more species and using this framework to conduct an exhaustive benchmark of the best performing tools for PPI predictions. Moreover, with the release of AlphaFold's large-scale reliable 3D protein structures (in collaboration with EMBL-EBI), prediction models based on tertiary structure become more relevant, and this field of research is likely to progress rapidly in the coming years. The field of green computing is nascent and more work is needed to promote green computing practices. There are countless ways to do so, from educating researchers to promoting transparency at institutional level. For the latter,

institutions, journals and funding bodies have a central role to play, as they have the power to request acknowledgements of carbon footprints, similarly to how conflicts of interests or ethical approvals are presented. Projects such as GA4HPC could also be implemented more systematically in HPC facilities and Trusted Research Environments.



The carbon footprint of this work

It is of course important to follow my own advice and acknowledge the carbon footprint of the different analyses presented in this thesis. Using the Green Algorithms calculator (v2.1), I estimated an upper bound for this work since 2019 of 1,666 kgCO_2e, or 151 tree-years; this is based on 18k GPU hours and 2k CPU hours on CSD3 in Cambridge (PUE = 1.15). As a commitment to reducing the environmental impact of computational research, I funded tree planting in the east of England region through carbonfootprint.com. This is imperfect, but these trees are hoped to sequester 3 tonnes of CO_2 in their lifetime, almost twice the upper-bound estimate.

More acknowledgements

This PhD was supported by the University of Cambridge MRC DTP (MR/S502443/1).

Most of the manipulations were done in Python [1] with Jupyter Notebooks [2] using the Pandas library [3], [4] and Numpy [5]. The plots were drawn using Matplotlib [6], Seaborn [7] and the MetBrewer colour palettes [8]. For the work on PPIs, all the code and final data are available on GitHub (https://github.com/Llannelongue/B4PPI); some intermediary pre-processed datasets are not available online due to file size limits but they can be recreated using the code available. The Green Algorithms project is also available on GitHub (https://github.com/GreenAlgorithms/green-algorithms-tool). The platform was created using Plotly Dash [9] and hosted on Heroku [10].

References

[1] 'The Python Language Reference — Python 3.10.1 documentation'. https://docs.python.org/3/reference/ (accessed Jan. 10, 2022).

[2] 'Jupyter Project Documentation — Jupyter Documentation 4.1.1 alpha documentation'. https://docs.jupyter.org/en/latest/ (accessed Jan. 10, 2022).

[3] J. Reback *et al.*, *pandas-dev/pandas: Pandas 1.0.3*. Zenodo, 2020. doi: 10.5281/zenodo.3715232.

[4] W. McKinney, 'Data Structures for Statistical Computing in Python', Austin, Texas, 2010, pp. 56–61. doi: 10.25080/Majora-92bf1922-00a.

[5] C. R. Harris *et al.*, 'Array programming with NumPy', *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[6] J. D. Hunter, 'Matplotlib: A 2D Graphics Environment', *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

[7] M. Waskom, 'seaborn: statistical data visualization', *JOSS*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/joss.03021.

[8] 'GitHub - BlakeRMills/MetBrewer: Color palette package in R inspired by works at the Metropolitan Museum of Art in New York'. https://github.com/BlakeRMills/MetBrewer (accessed Jan. 10, 2022).

[9] 'Dash Documentation & User Guide | Plotly'. https://dash.plotly.com/ (accessed Mar. 03, 2022).

[10] 'Documentation | Heroku Dev Center'. https://devcenter.heroku.com/categories/reference (accessed Mar. 03, 2022).

ANNEXE

Contents

Green Algorithms for Health Data Science

This is a blog article written with Jason Grealey and Michael Inouye at the beginning of the Green Algorithms project. It was released in March 2022 on the HDR UK's website:

https://www.hdruk.ac.uk/news/green-algorithms-for-health-data-science/

In the words of Dr Richard Horton, Editor in Chief of The Lancet:

> *"The climate emergency that we're facing today is the most important existential crisis facing the human species, and since medicine is all about protecting and strengthening the human species it should be absolutely foundational to the practice of what we do every single day."*

We, in the field of health data science, also abide by the Hippocratic Oath and should rise to the challenge of our climate emergency.

We tend to think that our main contribution is to design algorithms to solve the big health challenges of our time. Although there is little doubt that data science will be a key tool to tackle climate change, we often forget to consider how our own work also adds to the problem. The infrastructure we use, the algorithms and code that we write, all consume large quantities of electricity whose production is responsible for significant greenhouse gas emissions.

There is a widespread underappreciation, and indeed naivete, in our community as to the effects of our algorithms on carbon emissions and global warming. In searching for "data science and climate change", one has to pass the first ten pages of Google results to find the first article addressing the potential negative impact of algorithms. This needs to change.

In today's data-driven world, a problem doesn't exist until we can obtain data on it, which makes quantifying our carbon footprint a necessary first step to understand the depth of the issue before taking steps to reduce our impact.

Less than a handful of studies have tried to quantify algorithmic emissions (here, here and here). Those have concluded that more sustainable research is needed, particularly in image analysis and natural language processing as these models are famously power-

hungry and expensive to train. For example, the latest of Google's chatbots is trained continuously for 30 days, using over 2,000 TPU cores. While some initial steps have been made, we remain largely in the dark for much of the computational research carried out worldwide.

With that in mind, we propose a global framework enabling all scientists to easily measure their carbon emissions using an online calculator, available at www.green-algorithms.org. We use carbon-dioxide equivalent[1] (CO2e) as a single indicator of carbon impact. However, "2.3 kg of CO2e" is difficult to relate to, so we contextualise emissions calculations by also providing "tree-months": the amount of CO2 a mature tree is able to sequester (absorb) in one month, which is estimated at approximately 1kg[2]. If a project's emissions correspond to 120 tree-months, this means that 10 trees would need one full year to absorb its emissions. As we have found out, this is worryingly quick to achieve.

Many health data scientists are involved in genetic analyses, where performing a genome-wide association study (GWAS) is a frequent, sometimes daily, occurrence. For popular resources such as the UK Biobank, having 1,000 researchers worldwide each run a GWAS on 100 traits would release 290 tonnes of CO2e into the atmosphere. It would take 1,000 mature trees over 25 years to absorb this amount of CO2. Two hundred and ninety tonnes of CO2e is also approximately equivalent to each of the 1,000 scientists driving 1,000 miles (or 1600 km) in an average European car. Other tools and analyses, such as *de novo* human genome assembly or molecular dynamics simulations offer similar jarring carbon emissions when used at scale.

Now that we have a measurement tool, what can be done? One approach might be to better capture the cost of computation, not just a financial cost but a carbon cost. This could inform the design, selection, and implementation of algorithms as well as which analyses one needs to run. While financial costs are largely disclosed in grants, we would

[1] The amount of CO2 that would have the same effect on global warming.

[2] 950g to be exact, based on an estimation of 11.4kg per year.

posit that there is a public interest for transparency of carbon emissions: CO_2e estimates could be noted in academic publications.

Finally, there are ways for individual health data scientists themselves to limit their carbon impact:

- Only request the minimum memory you need. Power consumption depends on requested memory, not the actual memory usage. Requesting more memory "to be safe" is common but easily fixable.
- Test the algorithm or pipeline on small subsets of data. Thus, one may only run a power-hungry analysis once.
- Optimise the algorithm. Limit the running time, CPU and memory requirements. If using established software, many times this simply involves updating to the most recent version.
- Favour energy efficient data centres. Many data centres make their power usage effectiveness (PUE) available so that researchers can make informed decisions.
- Carefully choose the location of the servers if possible. For example, running the same algorithm on the same task would produce 64 times more CO2e in Australia than it would in Switzerland.

Perhaps most simply, run only the analysis you need. Overall, the environmental impacts of data science and computational research is an emerging and rapidly evolving field with seemingly infinite research areas, including infrastructure, algorithms, software, best-practice etc. We are hopeful that these initial steps lay the foundation for future studies and promote debate amongst researchers to work towards greener algorithms.

Carbon footprint, the (not so) hidden cost of High-Performance Computing

This is a blog/magazine article written for BCS IT Now's "Green IT" series for the COP26. It covers the material presented in ***Chapter 4*** *in a more informal and concise format.*

Lannelongue, Loïc. 2022.
'Carbon Footprint: The (Not so) Hidden Cost of High Performance Computing'.
ITNOW 63 (4): 12–13. https://doi.org/10.1093/itnow/bwab100.

There is something a bit surreal about having dozens of CPUs and gigabytes of memory at your fingertips. Yet, this is what High-Performance Computing (HPC) is all about: by centralising servers in data centres and opening safe channels of communications with the user, HPC facilities have drastically reduced the burden of purchasing and maintaining supercomputers. For users, it made it easier than ever to access these resources; any laptop, tablet or even smartphones can do it painlessly. Cloud computing platforms, such as Amazon's AWS, Google's GCP or Microsoft's Azure are the most popular options, but most research institutions and companies have their own data centre and follow the same principles.

On the plus side, such developments in computing have enabled discoveries like the first direct image of a black hole 55 million light-years away, weather forecasts more accurate than ever and the discovery of thousands of genetic variants related to diseases. To measure the magnitude of HPC usage, we can look at the usage statistics of XSEDE, the Extreme Science and Engineering Discovery Environment, a network of data centres in American universities used for scientific research. In 2020, every hour, one million compute hours were performed on the network, for a total of 9 billion compute hours.

What's the issue with this? After all, this leads to ground-breaking discoveries. It also requires mountains of hardware and the electricity to power it, which comes at a significant, yet largely ignored, environmental cost. It is estimated that data centres have a yearly carbon footprint of 100 megatons of CO2e[3] just from electricity production, similar

[3] Carbon Dioxide equivalent (CO2e) is a way to quantify the global warming impact of an activity.

to the entire American commercial aviation. Unsurprisingly, this will only increase in the next few years, by 2- to 9- fold in the next decade some studies say.

A variety of environmental impacts

There are numerous ways in which large computing facilities impact the environment: energy production, hardware manufacturing, long-term storage management, cooling, maintenance etc.

Energy usage is perhaps the aspect most discussed in the media. The environmental cost of powering data centres depends directly on the carbon footprint of energy production, which varies greatly with the energy mix of the country where the data centre is located. For example, producing 1 kWh of electricity in Switzerland (powered mainly by hydro) emits 12 gCO2e on average, but 253 gCO2e in the UK and 880 gCO2e in Australia (where coal is the main source of energy). Concretely, this means that exactly the same task will have a carbon footprint 73 times greater in Australia compared to Switzerland.

However, this is far from being the only downside. Manufacturing IT hardware is notoriously bad for the environment due to the extraction of precious metal. In the case of consumer devices, as much as 70% to 80% of the total carbon footprint is from manufacturing, with usage and disposal only responsible for the remaining 20 to 30%. This shows the importance for us to try to keep, repair and reuse our devices as much as possible. In data centres, it's also important to factor in environmental impact in the renewing cycle of hardware.

Artificial Intelligence: big models, huge carbon footprint

"Training a single AI model can emit as much carbon as five cars in the lifetimes." This headline was on the front of all the tech newspapers in the summer of 2019, from the MIT Technology Review to New Scientist. This followed an article studying the carbon footprint of algorithms trying to apprehend natural language, a task extremely difficult for which it's common for algorithms to run for days or even weeks. Similar concerns were raised in articles such as "Green AI" and "On the dangers of stochastic parrots: can language models be too big?", in which researchers discuss the issue of accessibility (what happens to research if only a handful of tech companies can afford to develop such models?) and the

risks posed by these technologies. In particular, they point out that the populations suffering most from the environmental cost of AI also benefit the least from innovations such as Apple's Siri or Amazon's Alexa. Notwithstanding the ethical issues arising if such tech companies are the only ones overseeing language models underpinning numerous aspects of society. Almost to illustrate this point, Google disbanded the *Ethical AI* team who participated in writing the stochastic parrots article shortly after its release.

These different studies led to the development of several tools aimed specifically at estimating the carbon footprint of machine learning models, such as the ML CO2 impact and the Experiment Impact Tracker.

AI may be the tree that hides the forest

The ever-growing computational needs of Artificial Intelligence are a real source of concerns, but we shouldn't forget about all the others fields of science that also rely heavily on computations. As discussed at the start, complex algorithms are everywhere, from genomics to physics and astronomy. This realisation that carbon footprint should be accounted for by all scientists using algorithms, not only in AI, is what led the development of a broad initiative, the Green Algorithms project. This is a theoretical framework (another one!) to estimate the carbon footprint of any computation, but more importantly, an online calculator to do the estimations easily. This helped shedding light of the carbon footprint of bioinformatics for example, where tools in genomics or molecular simulations emit kilograms of CO2e at each utilisation.

Since then, Australian astronomers have also investigated their environmental impact, and found that when taking into account all sources of emissions (travels, computations, offices etc.), the carbon footprint of an astronomer averages over 37 tCO2e per year, 5 times the global average. In several recent publications (here and here for example), astronomers urge their community to reduce this impact.

What about cryptocurrencies? The power usage of the so-called "mining farms" is estimated by the Cambridge Bitcoin Electricity Consumption Index at ~100 TWh per year. For comparison, if Bitcoin was a country, it would rank just behind the Netherlands in terms of energy usage, or alternatively, these 100 TWh could power all the British tea kettles for

22 years. There is also the rapid renewing cycle of the dedicated application-specific integrated circuits (ASIC) that creates mountains of technological waste difficult to recycle. This massive carbon footprint can't be ignored and similarly to other computation, it will be important to weigh the pros and cons of this technology moving forward.

Why don't we care?

When the computer is a loud tower in a corner of the room, the countless cables and the noisy fan are a reminder of the sort of energy needed to power it. On the other hand, the physical distance between users and data centres, which makes running demanding algorithms easier than ever, also make us forget about the resources used behind the scenes.

Besides, energy efficiency and carbon footprint are rarely a topic of discussion for computing hardware. Fridges, dishwashers, TVs, cars etc., all these devices promote their low energy needs and sustainability: being (or claiming to be) environmentally friendly is now a key element of an advertising campaign (as well as a legal requirement in many cases). This is not yet the case for computers, for which mostly speed and performances are discussed, and when energy efficiency is brought up in non-IT-specialist communities, it's usually to promote battery life and fast charging, not sustainability.

If you add to that that computing time is virtually free for a lot of scientists, it's no surprise that it hasn't been addressed earlier. Notwithstanding that even for scientists who would like to estimate their carbon footprint, until recently it was excessively difficult to do so and required extensive research on the environmental impact of each hardware component. Without data, the problem has mostly been ignored, but hopefully with tools such as Green Algorithms, this will start to change.

What now?

Does that mean that scientists should stop using algorithms for their work? Of course not, it has and will continue to enable amazing discoveries, including in the fight against climate change. However, similarly to the way the financial cost of a project is included a cost-benefit analysis prior to any projects, the environmental cost should similarly be considered.

Besides estimating and acknowledging their impact, there is a number of things scientists can do to limit the carbon footprint of their work, such as factoring in sustainability in hardware and software choices, optimise (or use optimised) code and avoid unnecessary computations. For more details, a short article presents "Ten simple rules to make your computing more environmentally sustainable".

In the coming years, High Performance Computing will certainly both continue to contribute to global warming and help fighting it. Only by being transparent about the carbon footprint of computation, and doing our best to reduce it, we can ensure that the net result benefits society globally.

Supplementary material – Chapter 4

Supplementary Table 4: Hardware details of the bioinformatic tasks evaluated.

Task	Name of the tool	Number of cores	CPU or GPU	Usage factor	Memory (GB)	CPU model	Number of hours	Notes
cis-eQTL mapping	FastQTL	1	CPU	1	52	Xeon E5-2699 v3	0.01	
cis-eQTL mapping	TensorQTL	1	GPU	1	52	Tesla P100 PCIe	0.000031	
eQTL mapping	TensorQTL v1.0.2	8	GPU	1	192	Tesla P100 PCIe	1.24	
eQTL mapping	LIMIX v2.0.3	N/A	CPU	N/A	N/A	N/A	N/A	Combined limix pipeline. Total footprint is summation of rows 6,7, and 8
eQTL mapping_part1	LIMIX v2.0.3	1	CPU	1	40.71	Xeon Gold 6142	9462.44	part 1 of total limix pipeline
eQTL mapping_part2	LIMIX v2.0.3	1	CPU	1	56.51	Xeon Gold 6142	232.32	part 2 of total limix pipeline
eQTL mapping_part3	LIMIX v2.0.3	1	CPU	1	220.9	Xeon Gold 6142	10.15	part 3 of total limix pipeline
Metagenome classification (short read)	Centrifuge	1	CPU	1	12	Any	0.97	
Metagenome classification (short read)	Kraken	1	CPU	1	21	Any	0.33	
Metagenome classification (short read)	Kraken/ Bracken	1	CPU	1	154	Any	1.67	
Metagenome classification (long read)	MetaMaps	1	CPU	1	262	Any	209.89	
PBMC-RNAseq: QC	FastQC	18	CPU	1	8	Xeon E5-2683 v4	370.17	Part 1 of total RNA-seq pipeline

check on rawData								
PBMC-RNAseq: adapter removal	TrimGalore	18	CPU	1	8	Xeon E5-2683 v4	79.74	Part 2 of total RNA-seq pipeline
PBMC-RNAseq: optical duplicates removal	bbmap/clumpify	16	CPU	1	8	Xeon E5-2683 v4	11.31	Part 3 of total RNA-seq pipeline
PBMC-RNAseq: Alignment	STARv2.7.0e	32	CPU	1	8	Xeon E5-2683 v4	23.98	Part 4 of total RNA-seq pipeline
RNA-seq QC pipeline	RNA-seq QC	.	.	.	.	.	485.2	This is for the entire pipeline RNA-seq. Total footprint is summation of rows 13 to 16
GWAS	BOLT-LMM v2.3	8	CPU	1	100	Xeon E5-2683 v4	60.96	
GWAS	BOLT-LMM v1	8	CPU	1	100	Xeon E5-2683 v4	224.16	
RNA-seq read alignment (human)	STAR	16	CPU	0.47	35	Xeon E5-2665	0.1	Pragmatic scaling factors are used to scale running time to CPU time (i.e. they have the same impact as utilising the CPU time in the Barruzo et al. manuscript)
RNA-seq read alignment (human)	HISAT2	16	CPU	0.98	5	Xeon E5-2665	0.03	
RNA-seq read alignment (human)	TopHat2	16	CPU	0.75	16	Xeon E5-2665	2.24	
RNA-seq read alignment (human)	Novoalign	1	CPU	1	64	Xeon E5-2665	32.19	
RNA-seq read alignment (malaria)	STAR	16	CPU	0.81	8	Xeon E5-2665	2.46	
RNA-seq read	HISAT2	16	CPU	0.98	1	Xeon E5-2665	0.03	

alignment (malaria)								
RNA-seq read alignment (malaria)	TopHat2	16	CPU	0.9	13	Xeon E5-2665	1.42	
RNA-seq read alignment (malaria)	Novoalign	1	CPU	1	21	Xeon E5-2665	38.07	
Metagenome assembly	MetaSPAdes	4	CPU	1	60	Xeon E5-4650L	29.4	
Metagenome assembly	MEGAHIT	4	CPU	1	12	Xeon E5-4650L	15.6	
Metagenome assembly	MetaVelvet k101	8	CPU	1	130	Xeon E5-4650L	1.1	
De novo human genome assembly	ABySS	64	CPU	1	34	Xeon E7-8867 v3	20	
De novo human genome assembly	MEGAHIT	64	CPU	1	197	Xeon E7-8867 v3	26	
Molecular dynamics simulation	AMBER18	1	GPU	1	0	Tesla V100	75	
Molecular dynamics simulation	NAMD v2.13	1	GPU	1	0	Tesla V100	400	
Protein docking	AutoDock Vina	1	CPU	1	0.05	Xeon X5660	40972.22	
Protein docking	Glide v57111	1	CPU	1	0.05	Xeon X5660	1027.78	
Protein docking	rDock	1	CPU	1	0.05	Xeon X5660	12250	
Transcript isoform abundance measurement	Cufflinks	1	CPU	1	11	Any	3.5	
Transcript isoform abundance inference	Cufflinks	16	CPU	1	12	Any	1.75	
Transcript isoform abundance measurement	RSEM	1	CPU	1	9	Any	47.17	
Transcript isoform abundance inference	RSEM	16	CPU	1	21	Any	8.83	
Transcript isoform abundance measurement	Sailfish	1	CPU	1	7	Any	0.7	
Transcript isoform	Sailfish	16	CPU	1	7	Any	0.23	

abundance inference								
Genome assembly scaffolding (short read)	SGA	1	CPU	1	30	Any	7.08	
Genome assembly scaffolding (short read)	SSPACE	1	CPU	1	30	Any	0.14	
Genome assembly scaffolding (short read)	SOAPde novo2	1	CPU	1	30	Any	0.19	
Genome assembly scaffolding (long read)	SGA	1	CPU	1	30	Any	1.59	
Genome assembly scaffolding (long read)	SSPACE	1	CPU	1	30	Any	0.06	
Genome assembly scaffolding (long read)	SOAPde novo2	1	CPU	1	30	Any	0.08	
Phylogeneti cs (codon model)	Beast/B eagle	1	CPU	1	2	Xeon E5-2680 v3	7.75	
Phylogeneti cs (codon model)	Beast/B eagle	2	CPU	1	2	Xeon E5-2680 v3	4.17	
Phylogeneti cs (codon model)	Beast/B eagle	4	CPU	1	2	Xeon E5-2680 v3	2.42	
Phylogeneti cs (codon model)	Beast/B eagle	6	CPU	1	2	Xeon E5-2680 v3	1.72	
Phylogeneti cs (codon model)	Beast/B eagle	8	CPU	1	2	Xeon E5-2680 v3	1.42	
Phylogeneti cs (codon model)	Beast/B eagle	10	CPU	1	2	Xeon E5-2680 v3	1.25	
Phylogeneti cs (codon model)	Beast/B eagle	12	CPU	1	2	Xeon E5-2680 v3	1.08	
Phylogeneti cs (codon model)	Beast/B eagle	1	GPU	1	2	Tesla P100 PCIe	0.08	
Phylogeneti cs (codon model)	Beast/B eagle	2	GPU	1	2	Tesla P100 PCIe	0.06	
Phylogeneti cs (nucleotide model)	Beast/B eagle	2	CPU	1	8	Xeon E5-2680 v3	0.67	
Phylogeneti cs (nucleotide model)	Beast/B eagle	4	CPU	1	8	Xeon E5-2680 v3	0.43	

Phylogenetics (nucleotide model)	Beast/Beagle	6	CPU	1	8	Xeon E5-2680 v3	0.4	
Phylogenetics (nucleotide model)	Beast/Beagle	8	CPU	1	8	Xeon E5-2680 v3	0.39	
Phylogenetics (nucleotide model)	Beast/Beagle	10	CPU	1	8	Xeon E5-2680 v3	0.43	
Phylogenetics (nucleotide model)	Beast/Beagle	12	CPU	1	8	Xeon E5-2680 v3	0.43	
Phylogenetics (nucleotide model)	Beast/Beagle	1	GPU	1	8	Tesla P100 PCIe	0.27	
Phylogenetics (nucleotide model)	Beast/Beagle	2	GPU	1	8	Tesla P100 PCIe	0.19	
Phylogeographic modelling	Beast/Beagle	2	CPU	1	8	Xeon E5-2680 v3	3.86	
Phylogeographic modelling	Beast/Beagle	4	CPU	1	8	Xeon E5-2680 v3	3.73	
Phylogeographic modelling	Beast/Beagle	6	CPU	1	8	Xeon E5-2680 v3	3.69	
Phylogeographic modelling	Beast/Beagle	8	CPU	1	8	Xeon E5-2680 v3	3.71	
Phylogeographic modelling	Beast/Beagle	10	CPU	1	8	Xeon E5-2680 v3	3.68	
Phylogeographic modelling	Beast/Beagle	1	GPU	1	8	Tesla P100 PCIe	0.64	
Phylogeographic modelling	Beast/Beagle	2	GPU	1	8	Tesla P100 PCIe	0.54	

Supplementary Table 5: The carbon footprint of hardware changes and parallelisation, using benchmarks from Beale et al.

Task	Algorithm	Number of CPU cores or GPU devices	Running time (hours)	Carbon footprint ($kgCO_2e$)
Codon substitution modelling	BEAST/ BEAGLE	1 2 4 6 8 10 12	7.75 4.17 2.42 1.72 1.42 1.25 1.08	0.066 0.069 0.078 0.083 0.091 0.10 0.10
		1 GPU 2 GPU	0.08 0.06	0.017 0.023
Nucleotide substitution modelling	BEAST/ BEAGLE	2 4 6 8 10 12	0.67 0.43 0.40 0.39 0.43 0.43	0.012 0.015 0.020 0.026 0.035 0.042
		1 GPU 2 GPU	0.27 0.19	0.054 0.076
Phylogeographic modelling	BEAST/ BEAGLE	2 4 6 8 10	3.86 3.73 3.69 3.71 3.68	0.070 0.13 0.18 0.24 0.30
		1 GPU 2 GPU	0.64 0.54	0.13 0.22

Supplementary Table 6: The ratio of RNA reads per million and ratio of CPU time of 10 random in-house PBMC samples, from the RNA sequencing quality control pipeline task.

FastQC				
Sample_name	reads per million (raw reads)	CPU time taken (mins)	ratio of reads per million wrt sample 1	ratio of CPU time taken wrt sample 1
1	18.931429	4.6833	1	1
2	21.649703	5.2333	1.143585252	1.117438558
3	29.986196	7.1833	1.583937272	1.533811629
4	45.932577	10.65	2.426260427	2.274037538
5	51.760103	12.0667	2.734083254	2.576537911
6	55.506794	12.75	2.931991769	2.722439306
7	66.599554	15.6167	3.517935915	3.334550424
8	66.905233	15.9167	3.534082557	3.398607819
9	70.699164	16.6333	3.734486393	3.551619584
10	71.487848	16.8167	3.776146428	3.590780006
11	129.240842	30.9833	6.826787455	6.615698332

TrimGalore				
Sample_name	reads per million (raw reads)	CPU time taken (mins)	ratio of reads per million wrt sample 1	ratio of CPU time taken wrt sample 1
1	18.931429	70.5167	1	1
2	21.649703	72.4833	1.143585252	1.027888429
3	29.986196	100.35	1.583937272	1.42306716
4	45.932577	159.5833	2.426260427	2.263056836
5	51.760103	176.85	2.734083254	2.507916564
6	55.506794	197.8333	2.931991769	2.805481538
7	66.599554	201.7	3.517935915	2.860315358
8	66.905233	228.4167	3.534082557	3.239185895
9	70.699164	221.4833	3.734486393	3.140863086
10	71.487848	267.5667	3.776146428	3.794373531
11	129.240842	440.5	6.826787455	6.246747224

Bbmap/clumpify				
Sample_name	reads per million (raw reads)	CPU time taken (mins)	ratio of reads per million wrt sample 1	ratio of CPU time taken wrt sample 1
1	18.931429	29.2	1	1
2	21.649703	33.45	1.143585252	1.145547945
3	29.986196	44.8167	1.583937272	1.534818493
4	45.932577	75.2667	2.426260427	2.577626712
5	51.760103	85.0167	2.734083254	2.911530822
6	55.506794	91.4	2.931991769	3.130136986
7	66.599554	90.9333	3.517935915	3.11415411
8	66.905233	104.8833	3.534082557	3.591893836
9	70.699164	106.1	3.734486393	3.633561644
10	71.487848	116.7	3.776146428	3.996575342
11	129.240842	212.8167	6.826787455	7.288243151

Star alignment				
Sample_name	reads per million (after dedup)	CPU time taken (mins)	ratio of reads per million wrt sample 1	ratio of CPU time taken wrt sample 1
1	16.688146	57.0667	1	1
2	19.040731	65.6	1.140973419	1.149532039
3	26.248112	85.8333	1.572859681	1.504087322
4	39.827254	125.1833	2.386559538	2.193631312
5	45.998551	140.0833	2.756360772	2.45472929
6	48.654564	163.65	2.915516439	2.867696923
7	55.983788	274.1	3.354703872	4.8031514
8	58.466638	177.0333	3.503483131	3.10221723
9	61.188251	244.1333	3.666569732	4.2780343
10	61.950514	194.3333	3.712246645	3.405371259
11	113.215654	349.3833	6.78419604	6.122367335

Supplementary Note 1: Scripts used to perform the in-house analyses.

```
#########################################################################

#### Input file: sample1_Read1_fastq.gz, sample1_Read2_fastq.gz   ####

#### Output directory: output/                                    ####

#########################################################################
```

1. Perform quality control checks on raw sequence data using FastQC

%fastqc <input_fastq_file_name> -o <output_directory>

%fastqc sample1 -o output/

2. Adapter trimming using TrimGalore

%trim_galore --illumina --paired <input_fastq_file_Read1> <input_fastq_file_Read2> -o <output_directory>

%trim_galore --illumina -–paired sample1_Read1_fastq.gz sample1_Read2_fastq.gz -o output/

3. Remove cluster reads generated by the sequencer using bbmap v38.68/clumpify.sh, where the user needs to specify a distance in the parameter and it is platform-specific. The RNA sequencing of our in-house data) was sequenced using Illumina NovaSeq. The recommended distance parameter for Illumina NovaSeq was used.

%bbmap/clumpify.sh in1=<input_fastq_file_Read1> in2=<input_fastq_file_Read2> out1=<output_fastq_file_Read1> out2=<output_fastq_file_Read2> dedupe allduplicates subs=0 optical=t dupedist=12000

%bbmap/clumpify.sh in1=sample1_Read1_val_fastq.gz in2=sample1_Read2_val_fastq.gz out1=sample1_Read1_dedupe_fastq.gz out2=sample1_Read2_dedupe_fastq.gz dedupe allduplicates subs=0 optical=t dupedist=12000

4. Reads alignment using STAR2.7.0.e

%STAR --runMode alignReads --genomeDir <genome_file> --readFilesCommand zcat --readFilesIn <input_fastq_file_Read1> <input_fastq_file_Read2> --runThreadN 8 --outFileNamePrefix <output_directory/input_name> --outSAMtype BAM SortedByCoordinate --quantMode Genecounts --sjdbGTFfile <GTF_file_with_annotations>

%STAR --runMode alignReads --genomeDir ref/GRCh38/STARindex --readFilesCommand zcat --readFilesIn sample1_Read1_dedupe_fastq.gz sample1_Read2_dedupe_fastq.gz --runThreadN 8 --outFileNamePrefix output/sample1. --outSAMtype BAM SortedByCoordinate --quantMode Genecounts --sjdbGTFfile Ensemble/Homo_sapiens.GRCh38.98.gtf

Python script to run cis-eQTL mapping

```
import sys
import pandas as pd
import tensorqtl
from tensorqtl import genotypeio, cis, trans

# Specify root directory for analysis
path = "..."

# Specify output path (and prefix if desired). Can be set to any folder if you want it
outside the analysis directory
outpath = path + "results/cis_eQTLs/"

# Get analysis chromosome from command line arguments, this in turn comes from
SLURM array ID
chr = str(sys.argv[1])

# Set up file paths
phenotype_bed_file = path + "phenotypes/INTERVAL_RNAseq_phase1-
2_filteredSamplesGenes_TMMNormalised_FPKM_Counts_foranalysis_chr" + chr +
".bed.gz"
covariates_file = path + "covariates/INTERVAL_RNAseq_phase1-
2_fullcovariates_foranalysis.txt"
plink_prefix_path = path + "genotypes/INTERVAL_RNAseq_Phase1-
2_imputed_b38_biallelic_MAF0.005_chr" + chr

# Read in phenotypes
phenotype_df, phenotype_pos_df = tensorqtl.read_phenotype_bed(phenotype_bed_file)

# Read in covariates and make subset to only ids that are in the phenotype file
covariates_df = pd.read_csv(covariates_file, sep='\t', index_col=0)
covariates_df = covariates_df[phenotype_df.columns].T

# Read in genotypes
pr = genotypeio.PlinkReader(plink_prefix_path)

# load genotypes and variants into data frames
genotype_df = pd.DataFrame(pr.load_genotypes(), index=pr.bim['snp'],
columns=pr.fam['iid'])
variant_df = pr.bim.set_index('snp')[['chrom', 'pos']]

# Cis gene-level mapping
cis_df = cis.map_cis(genotype_df, variant_df, phenotype_df, phenotype_pos_df,
covariates_df)
tensorqtl.calculate_qvalues(cis_df, qvalue_lambda=0) # Lambda of 0 is equivalent to BH
correction. Prevents crashes for chr 9, 18, 22.
```

```
cis_df.to_csv(outpath + "tensorqtl_cis_MAF0.005_cisPerGene_chr" + chr + ".csv",
index=True, index_label = "Phenotype")

# Cis nominal mapping
cisnom_df = cis.map_nominal(genotype_df, variant_df, phenotype_df,
phenotype_pos_df, covariates_df, prefix=outpath +
"tensorqtl_cis_MAF0.005_cisNominal_chr" + chr)
cisnom_df2 = pd.read_parquet(outpath + "tensorqtl_cis_MAF0.005_cisNominal_chr" +
chr + ".cis_qtl_pairs." + chr + ".parquet")
cisnom_df2.to_csv(outpath + "tensorqtl_cis_MAF0.005_cisNominal_chr" + chr + ".csv",
index = False)

# Conditional analysis # commented out because it times out
#indep_df = cis.map_independent(genotype_df, variant_df, cis_df, phenotype_df,
phenotype_pos_df, covariates_df, nperm=10000)
#indep_df.to_csv(outpath + "tensorqtl_cis_MAF0.005_cisIndependent_chr" + chr +
".csv", index=True, index_label = "Phenotype")

# GxE
#interaction_s = pd.read_csv("INTERVAL_RNAseq_phase1_GxE_neutPCT.txt", sep = "\t",
index_col=0, squeeze = True).T
#cisGxE_df = cis.map_nominal(genotype_df, variant_df, phenotype_df,
phenotype_pos_df, covariates_df, prefix="Test_gxe", interaction_s=interaction_df)
#cis.map_nominal(genotype_df, variant_df, phenotype_df, phenotype_pos_df,
covariates_df,
prefix="tensorqtl_cis_MAF0.005_cisGxE_chr1",interaction_s=interaction_s,
maf_threshold_interaction=0.05,group_s=None, run_eigenmt=True,
output_path=outpath)
```