

In silico discovery of copper oxidases for the degradation of polyethylene

Promoter:
Prof. Vera Van Noort
Microbial and Molecular Systems
Centre of Microbial and Plant Genetics

Dissertation presented in
fulfillment of the requirements
for the degree of Master of Science:
Bioinformatics

Miguel BILTJES



*Copyright Information:
student paper as part of an academic education
and examination.
No correction was made to the paper
after examination.*

Acknowledgments

My gratitude goes out to my supervisor Maarten Langen. From the very start he has granted me complete freedom in the choice of topic. I was allowed to pursue my own interests, and he was always receptive and willing to listen to the ideas I put forth. He proofread the thesis multiple times, helped me implement the methods, gave feedback on presentation skills, and took the time to discuss my progress on a weekly basis. I also want to express my gratitude to every other member of the lab for letting me present my work, question it, and for simply shaping the environment in which all of this took place: Stefaan, Hannelore, Nand, Jaldert, and Sophie.

I also thank prof. Vera van Noort for providing valuable feedback on multiple occasions, advising me on which methods to use, and what to focus on during the writing process. I am especially grateful for the opportunity she provided in reaching out to a company, *B4Plastics*, who were willing to listen to the work I performed and discuss future job opportunities.

All the authors listed in the bibliography also deserve a word of appreciation. This work could not have been written without their contributions to the conversation of copper oxidases, plastics degradation, and many other things. As is the case with any scientific work, I too stood on the shoulders of giants.

Last but not least, none of this would have been possible without my loving and supportive parents, as well as my equally loving and supportive partner. Despite understanding very little of what is written in the following pages, they were still willing to read it and provide feedback.

Abstract

Polyethylene (PE) is the most abundantly produced and discarded synthetic polymer globally. Its accumulation in the environment poses significant ecological risk for both humans and non-human species. The majority of PE waste is either disposed at land fills or incinerated, and the amount actually upcycled into valuable chemicals is less than 1%. Enzymes have been proposed as eco-friendly catalysts for the deconstruction of PE. However, due to this plastic's inert nature, only a limited amount of enzymes are known today, and none of them achieve a complete depolymerization. Currently, laccase-like multicopper oxidases (LMCOs) are the dominant enzymes involved in PE degradation. Recently, a novel LMCO has been isolated from *Rhodococcus* bacteria, with its degradative properties attributed to a hydrophobic binding region rich in Met residues.

This thesis aims to expand the current repertoire of PE-degrading enzymes by identifying candidates worthy of screening in an experimental setting. LMCOs known to degrade PE were extracted from the literature (three sequences). All multicopper oxidases in the Uniprot database were fetched (64,358 sequences). Two metagenomic studies from environments enriched in plastic debris were gathered from the MGNIFY database (9,592,718 sequences), and similar enzymes were identified through hidden Markov Models and the MMSEQS2 *search* module. A computational screening method was then developed to identify LMCOs containing a binding region that is (i) rich in Met residues, (ii) hydrophobic, and (iii) surface-accessible.

In the MGNIFY data set, 469 LMCOs were predicted, 23 of which fulfilled all three criteria. Forty-eight candidates were identified in the UniProt data set, 31 of which are from fungal origin. In total, they represent 71 candidate LMCOs with hydrophobic, surface-accessible regions rich in Met residues. Clustering at 90% sequence identity revealed 53 cluster representatives, indicating considerable sequence diversity among the candidates. Variety was observed concerning the location, composition, and length of the identified binding regions. The main claim of this thesis is that the 71 candidates can be tested in an experimental setting to determine their PE-degrading capabilities, and to evaluate the real-life value of this screening strategy.

Contents

1	Introduction	1
1.1	Context and aims	1
1.1.1	The plastic pollution problem	1
1.1.2	Aims and motivation	2
1.2	Properties of PE	3
1.2.1	PE has a C-C backbone	3
1.2.2	PE contains various degrees of branching	4
1.2.3	PE is a semi-crystalline polymer	5
1.2.4	PE is a polydisperse polymer	5
1.3	Degradation of PE	5
1.3.1	Experimental evidence of enzymatic degradation	6
1.3.2	Metagenomic insights	13
1.4	Laccase-like multicopper oxidases	16
1.4.1	Clarification of terminology	16
1.4.2	Biological distribution and function	18
1.4.3	Structural properties	19
2	Materials and Methods	25
2.1	Literature study	26
2.2	Data sets	26
2.2.1	Experimentally verified PE-degrading enzymes (PE)	26
2.2.2	Representative LMCOs (RE)	26
2.2.3	MCOs in UniProt (UP)	28
2.2.4	Metagenomic studies related to plastic (MG)	28
2.3	Alignment of the PE-RE data set	28
2.4	Searching LMCOs in the MG data set	29
2.4.1	Regular expression patterns	29
2.4.2	Hidden Markov model	29
2.4.3	MMSEQS2	30
2.5	Candidate screening	30
2.5.1	Predicting regions enriched in Met	30
2.5.2	Scoring hydrophobicity	32
2.5.3	Scoring surface accessibility	32
2.6	Further analysis of the candidates	33
2.6.1	Predicting the T1 Cu axial ligand	33
2.6.2	Clustering and alignment	33
2.6.3	Structure prediction	33
3	Results	34
3.1	Data exploration	34
3.1.1	Raw UP data set	34
3.1.2	Filtered UP data set	36

3.2	Alignment of the PE-RE data set confirms homology	38
3.3	HMM profile	42
3.4	469 predicted LMCOs in the MG data set	42
3.5	Parameter optimization	43
3.5.1	Identifying MR regions	43
3.5.2	Scoring hydrophobicity	45
3.5.3	Scoring surface accessibility	45
3.6	71 Candidates found	46
3.6.1	297 LMCOs with MR regions	46
3.6.2	71 LMCOs pass hydrophobicity and RSA thresholds	47
3.7	Further analysis of the candidates	48
3.7.1	Taxonomy and EC number of the UP candidates	48
3.7.2	One LMCO with Leu as axial ligand	48
3.7.3	Clustering and alignment	49
3.7.4	Predicted structures	49
4	Discussion	52
4.1	Interpretation of results	52
4.1.1	UP LMCOs not matching RegEx patterns	52
4.1.2	Enrichment of MR regions in metagenomic sequences	53
4.1.3	Limitations of MR region prediction	53
4.2	Comparison to the ACTPAC approach	54
4.3	Future perspectives	55
4.3.1	Pocket analysis can provide more informed decisions	55
4.3.2	Experiment proposal	56
4.3.3	Towards industrial applications	58
A	Code snippets	66
A.1	Commands executed	66
A.2	MG data retrieval	67
A.3	PE-RE pairwise alignment	68
A.4	Q-stat implementation	70
A.5	Sliding window implementation	73
B	ESMFold plDDT plots	76
C	Additional background information	77
C.1	A brief history of plastics	77
C.2	Techniques to measure degradation	78
D	GenAI statement	81

List of Abbreviations

- 2dMCO* Two domain multicopper oxidase
3dMCO Three domain multicopper oxidase
6dMCO Six domain multicopper oxidase
ABTS 2,2'-azino-bis(3-ethylbenzothiazoline-6-sulfonic acid)
ADH Alcohol dehydrogenase
AFM Atomic force microscopy
AH Alkane hydroxylase
ATR – FTIR Attenuated total reflection Fourier-transform infrared spectroscopy
BRENDA Braunschweig enzyme database
BVMO Baeyer–Viller mono-oxygenase
CI Carbonyl index
DFT Density functional theory
DMP Dess–Martin periodinane
DSC Differential scanning calorimetry
EC Enzyme Commission
FTIR Fourier-transform infrared spectroscopy
GFP Green fluorescent protein
GPC Gel permeation chromatography
HBT Hydroxybenzotriazole
HDPE High density polyethylene
HMM Hidden Markov model
HT – GPC High temperature gel permeation chromatography
ITS Internal transcribed spacers
LDPE Low density polyethylene
LiP Lignin peroxidase
LLDPE Linear low density polyethylene

<i>LMCO</i>	Laccase-like multicopper oxidase
<i>LMWPE</i>	Low molecular weight polyethylene
<i>MCO</i>	Multicopper oxidase
<i>mCPBA</i>	meta-Chloroperoxybenzoic acid
<i>MG</i>	MGNIFY database
<i>MnP</i>	Manganese peroxidase
<i>MP</i>	Methionine proportion
<i>MR</i>	Methionine-rich
<i>NCBI</i>	National center for biotechnology information
<i>NHE</i>	Normal hydrogen electrode
<i>OTU</i>	Operational taxonomic unit
<i>PAH</i>	Polycyclic aromatic hydrocarbons
<i>PAZy</i>	Plastics active enzyme database
<i>PDB</i>	Protein data bank
<i>PE</i>	Polyethylene
<i>PET</i>	Polyethylene terephthalate
<i>PP</i>	Polypropylene
<i>RE</i>	Relative elongation
<i>RegEx</i>	Regular expression
<i>ROS</i>	Reactive oxygen species
<i>RSA</i>	Relative surface accessibility
<i>RTS</i>	Relative tensile strength
<i>SEC</i>	Size exclusion chromatography
<i>SEM</i>	Scanning electron microscopy
<i>T1, T2, T3</i>	Type 1, 2 and 3
<i>TEMPO</i>	(2,2,6,6-Tetramethylpiperidin-1-yl)oxyl
<i>TNC</i>	Trinuclear cluster
<i>UP</i>	UniprotKB database
<i>UV</i>	Ultraviolet
<i>WCA</i>	Water contact angle
<i>WL</i>	Weight loss
<i>XPS</i>	X-ray photoelectron spectroscopy
<i>XRD</i>	X-ray diffraction

List of Figures

1.1	Molecular structure of PE, PP and PET.	4
1.2	Difference between HDPE, LDPE and LLDPE.	4
1.3	Plastic degrading species and proteins listed in the plasticDB per polymer type.	7
1.4	Predicted structures of two PE-degrading LMCOs.	10
1.5	Presumed reaction mechanism of PE oxidation by an LMCO.	11
1.6	Enzymatic cascade for the deconstruction of PE into low molecular weight carboxylic acids	12
1.7	SEM images of micro-organisms colonizing marine plastic debris.	13
1.8	Proposed plastic degradation pathway inferred from metagenomic study.	15
1.9	GO term annotations for two MGNIFY studies.	16
1.10	General reaction mechanism of an LMCO.	17
1.11	Schematic illustration of the cupredoxin fold and overall LMCO architecture.	20
1.12	Copper site architecture of a bacterial and a fungal LMCO.	21
1.13	His-Cys-His electron transfer pathway in LMCOs	22
1.14	MR regions in <i>B. subtilis</i> and <i>T. thermophilus</i> LMCO.	22
1.15	MR region in <i>E.coli</i> allowing rapid immobilization on carbon nanotubes.	23
2.1	Schematic illustration of the entire workflow.	25
2.2	Example of q-stat increasing when Met residues cluster together.	31
2.3	Q-stat distribution after 10,000 random shuffles.	31
3.1	Top five EC numbers in the raw UP data set	34
3.2	Length and mass distribution of the raw UP data set	35
3.3	Taxonomy of the raw UP data set.	35
3.4	Top 5 EC numbers in the filtered UP data set.	36
3.5	Taxonomy of the filter UP data set.	37
3.6	Met proportion of the filtered UP data set.	37
3.7	Axial ligands in the filtered UP data set by kingdom.	38
3.8	Alignment of the PE-RE data set.	39
3.9	Alignment of the PE-RE data set (continued).	40
3.10	Logo of the HMM profile derived form the MSA of the PE-RE data set.	42
3.11	Venn diagram of predicted LMCOs in the MG data set by RegEx patterns, HMM and MMSEQS2.	43
3.12	Example of NetSurfP output.	45
3.13	Summary of the screening process for the UP and MG data set.	46
3.14	Venn diagram of MR regions predicted by q-stat and both sliding windows.	47
3.15	Distribution of length and MP for MR regions in the UP and MG data set.	47
3.16	Distribution of hydrophobicity score and RSA for MR regions in the UP and MG data set.	48
3.17	Part of MSA of candidate LMCOs.	49
3.18	MR regions for three candidates.	50
3.19	Predicted structures of two UP candidates	51
3.20	Predicted structure of MG candidate	51

4.1 Comparison to ACTPAC’s pathway for PE depolymerization.	55
B.1 plDDT plots for ESMFold structure predictions.	76

List of Tables

1.1	Experimental evidence of enzymatic PE degradation.	8
2.1	Representative LMCOs.	27
2.2	Hydrophobicity scores.	32
3.1	Sequence identity matrix of the PE-RE data set	41
3.2	Static sliding window test.	44
3.3	Dynamic sliding window test.	44

Chapter 1

Introduction

1.1 Context and aims

1.1.1 The plastic pollution problem

It has been estimated that since 1950, 9.2 billion tonnes of plastics have been produced (approximately the same weight as 900,000 Eiffel towers or 1.2 billion elephants), of which 5.3 billion tonnes has been discarded into the environment, and 1 billion tonnes incinerated (Geyer, 2020). Polyethylene accounted for approximately 2.2 billion tonnes (41%) of the discarded plastics, while at the same time being one of the most persistent and recalcitrant synthetic polymers. Recent estimates state that 414 million tonnes of plastics were manufactured in 2023, of which 374 million tonnes were fossil-based (i.e., not recycled or made from sources other than oil/gas reservoirs) (Plastics Europe, 2024). The packaging industry is the major contributor to plastics production, accounting for about 40% since 1950, and approximately 90% of packaging is made of PE, polypropylene (PP) or polyethylene terephthalate (PET) (Geyer, 2020).

Since plastics are designed to be durable and resistant to degradation, the majority has been accumulating—in their polymer form—in marine and terrestrial environments. Indeed, an estimated 80% of all marine waste is thought to be derived from land, with 50% of marine litter consisting of single-use plastic items (Crawford & Quinn, 2017; European Commission, 2021). The persistence of these materials poses significant ecological risk to marine species and humans, as well as financial burdens through loss of tourism and damage to marine ecosystems (European Commission, 2021).

Once discarded, macroplastic debris fragments into smaller particles called microplastics and nanoplastics, primarily due to mechanical stresses (e.g., waves, collisions with rocks) (Crawford & Quinn, 2017). Currently, microplastics are so ubiquitous that they can be found at nearly every beach, sediment layer, ocean, lake, and even within polar ice caps (Crawford & Quinn, 2017). Microplastics are also known to adsorb and absorb persistent organic pollutants (POPs) such as polycyclic aromatic hydrocarbons (PAH) and perfluoralkylates at concentrations 1000 times higher than their surroundings, which exhibit considerable toxicity when ingested (Crawford & Quinn, 2017). Additionally, biofilms containing pathogenic micro-organisms (e.g., *Vibrio* species) can grow on plastic debris that can be transported over long distances, possibly arriving at public beaches (Amaral-Zettler et al., 2020). Thus, microplastics can act as sponges and vectors for both POPs and pathogens.

Numerous marine species such as fish, mussels, turtles and seabirds have been observed to consume both macro- and microplastics (Crawford & Quinn, 2017). Studies at European coastlines found microplastics in every organism that was collected (e.g., 1.2 microplastics/g in the lugworm *A. marina*) (Crawford & Quinn, 2017). In some cases, ingestion leads to intestinal blockage with death as a result. Humans are exposed to microplastics as well, either through diet, inhalation or dermal penetration (Sun & Wang, 2023). However, the exact quantities and health effects are still a matter of investigation. It is estimated that the European individual ingests up to 11,000 microplastics per year through eating shellfish (Crawford & Quinn, 2017). In a study testing human feces

on the presence of microplastics, 95% of samples tested positive, with concentrations up to 138.9 items/g in the form of fragments, films and fibers (Sun & Wang, 2023). Furthermore, PE included in the diet of mice has been shown to exhibit immuno- and neurotoxic effects (Sun & Wang, 2023).

The above issues have led to the EU banning certain single-use plastic packaging in 2021 (European Commission, 2021), as well as funding a recent initiative—ACTPAC—specifically aiming to develop new catalysts for the deconstruction of PE into value-added chemicals (ACTPAC, 2025). Clearly, efficient ways of processing plastic waste, other than disposal at landfills or burning, are in demand. Especially concerning are the C-C backbone polymers such as PE and PP, which make up the majority of commercially produced plastics, while at the same time being the most recalcitrant due to their inert chemical nature.

Currently, the amount of PE actually upcycled into value-added products is less than 1% (ACTPAC, 2025), and no industrially viable solution for transforming PE into low molecular weight compounds exists. Most approaches rely on pyrolysis (burning in the absence of oxygen) to produce fuels and waxes of different kinds (Guo, 2025; Tournier et al., 2023). However, this process requires extremely high temperatures ($> 400^{\circ}\text{C}$) and/or pressures, and is usually of low selectivity, resulting in a broad spectrum of end products (Tournier et al., 2023). Hence, pyrolysis is considered more as a downcycling method due to its high energy demand and low commercial viability.

Alternatively, micro-organisms and the enzymes they produce have been proposed as an eco-friendly solution to catalyze the degradation of plastics (Tournier et al., 2023). Indeed, the enzymatic deconstruction of PET into its respective monomers has already been successfully industrialized by *Carbios*, a French company. However, PET contains reactive functional ester groups suited for enzymes (esterases) to attack. PE, on the other hand, is considered non-degradable and highly inert due to the absence of functional groups, extreme hydrophobicity, and large molecular dimensions (Ghatge et al., 2020). Hence, only a limited amount of PE-degrading enzymes are known today, and not a single one is able to achieve a complete depolymerization. Thus far, only minor weight losses and surface modifications of milligram-scale amounts have been reported.

1.1.2 Aims and motivation

This thesis aims to expand the currently available repertoire of PE-degrading enzymes through computational screening methods. Enzymes currently known to act on PE will be extracted from the literature. Then, larger data sets will be screened for proteins with high sequence similarity, and filtered based on a set of criteria which mostly relate to the presence of hydrophobic binding regions. The main claim of this thesis is that the sequences passing all criteria are worthy to be subjected to an experiment. Hence, at the end of this work, candidate enzymes will be proposed, saving both time and money for the initial screening process in future laboratory settings, and bringing us one step closer to industrial applications.

The motivation for this work is threefold: Firstly, the ongoing mass production and disposal of PE and other plastics is detrimental to ecosystems in numerous ways, and possibly affects human health as well. In order to prevent further irreversible damage, innovative and ecologically friendly ways of processing plastic waste are required.

Secondly, if it were possible to transform PE waste into economically valuable, non-toxic compounds, this would contribute to the realization of a circular economy. That is; an economy where waste products are upcycled and reused. Currently, millions of tonnes of PE, now regarded as waste, can be harvested from the environment and could be turned into valuable resources if a proper and fast depolymerization can take place. As mentioned above, the deconstruction of PET into its respective monomers has already been commercialized by *Carbios*. For them, PET waste drives the business model. It is only a matter of time for PE and other synthetic polymers to follow.

Thirdly, a more intrinsic motivation; the (bio)chemical nature of the problem at hand is rather appealing (at least to the author of this work) and challenging. Is it possible to enzymatically transform the inert C-C bonds in PE? Can nature provide a solution for the breakdown of molecules specifically designed not to be broken down? Million dollar industries have been built around these synthetic polymers, and they surround us throughout our everyday lives. It is simply interesting to gain a better understanding of their chemistry through this work.

Section 1.2 highlights the key chemical and physical properties of PE related to its recalcitrance. Section 1.3 reviews the experimental evidence of PE degradation and the enzymes involved, along with some insights from metagenomic studies of plastic-enriched environments. Section 1.4 discusses the biological function and structure of one of these enzymes: laccase-like multicopper oxidases (LMCOs). At the end of the introduction, the reader should primarily take away that (i) LMCOs are the dominant enzymes involved in PE degradation, and that (ii) the presence of hydrophobic binding regions rich in Met residues facilitate the formation of an LMCO-PE complex, allowing oxidation and subsequent depolymerization to occur.

1.2 Properties of PE

The main appealing features of PE are its low cost, resistance to chemical attack, processability, toughness, flexibility and electrical insulation properties (Ronca, 2017). PE arguably has one of the simplest molecular structures of all plastics currently produced. Essentially, it can be thought of as a tightly packed matrix of long hydrocarbon chains; a homogeneous repetition of CH_2 motifs. There are no functional groups, aromatic rings, or other complex chemical patterns. Currently, hundreds of different grades of PE exist, mainly distinguished based upon the variation in (i) degree of short-chain branching, (ii) degree of long-chain branching, (iii) average molar mass and (iv) presence of co-monomers or impurities (Ronca, 2017).

The chemical properties of PE are to a large extent comparable to those of alkanes, since PE is essentially an alkane of high molecular weight. Alkanes are notorious for their inertness due to the highly stable sp^3 -hybridized C-C bonds and lack of atoms with free electron pairs (Engbersen & de Groot, 2017). Due to the low difference in electronegativity, breaking the bonds most often occurs through a homolytic process, resulting in free radical intermediates (Engbersen & de Groot, 2017). An important consequence of this inertness is that alkanes from oil spills remain present in the environment for long periods of time, which is also the case for PE. Another shared characteristic is the extreme hydrophobicity caused by the little amount of polarization in the molecule. Hence, no solvents, not even non-polar ones, exist for PE at room temperature (Ronca, 2017).

Additionally, efforts have been made throughout history to increase the flexibility and recalcitrance of plastics by implementing additives such as anti-oxidants, flame retardants and antimicrobial agents, which are non-covalently bound to the plastic (Tournier et al., 2023). After all, plastics are produced for their performance and durability, not to be recycled or degraded. Even though the impact of these additives is of significance, they will be disregarded for this thesis, as they are only present in relatively small amounts.

1.2.1 PE has a C-C backbone

Plastics are often categorized by the nature of their polymer backbone (Tournier et al., 2023). Either it consists of C-C bonds, or C-X bonds, where X represent nitrogen or oxygen in the majority of cases. Figure 1.1 shows this distinction for three commonly produced plastics. PE and PP have C-C backbones, while PET has a C-O backbone.

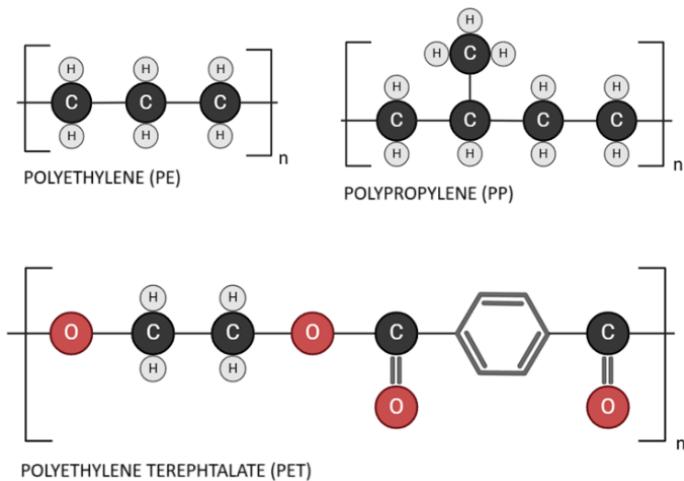


Figure 1.1: Three common plastics and their simplified monomer structure. The 'n' means that the motif is repeated many times to form the polymer. PE and PP have a C-C backbone. PET has a C-O backbone. This figure was created with BioRender.com.

The nature of the polymer backbone has significant implications for its rate of degradation. C-X backbones contain functional groups with reactive and accessible electrons. Hence, it is generally considered easier to break down C-X backbone plastics such as PET.

1.2.2 PE contains various degrees of branching

PE can be categorized according to the amount of branching into high density PE (HDPE), low density PE (LDPE), linear low density PE (LLDPE), and theoretically anything in between (Crawford & Quinn, 2017; Ronca, 2017). HDPE has very few branches along the main polymer chain and allows for tighter and more compact packaging of the chains. LDPE and LLDPE have a higher frequency of branching, but the nature of the branches differ. In LDPE, branches are typically longer and might contain sub-branches, whereas LLDPE only contains short linear branches. The effect of these branches is a reduction in compressibility and density, and an increase in flexibility of the material (Crawford & Quinn, 2017). Figure 1.2 shows the difference between HDPE and (L)LDPE.

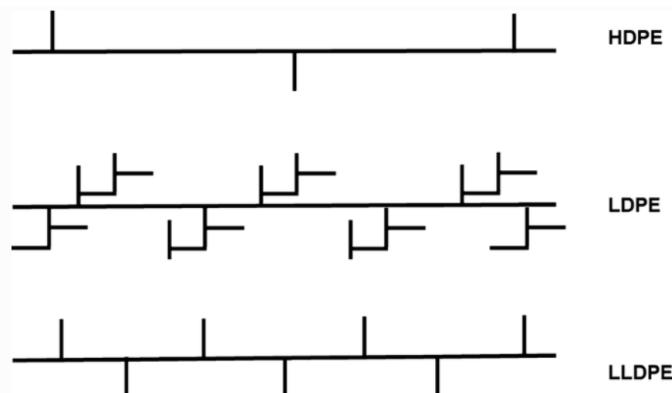


Figure 1.2: Different types of PE according the amount and nature of branching. Simplified representation. Figure taken from Ghatge et al. (2020)

1.2.3 PE is a semi-crystalline polymer

The crystallinity of polymers refers to the degree of order in the stacking of the chains. This property directly affects the polymer's mechanical, chemical, optical and thermal properties (Crawford & Quinn, 2017). For PE, the degree of crystallinity is tightly linked to the amount of branching as described above. When the PE polymer is in a melted state (f.e. during the production process), the chains are highly motile and arrange themselves in mostly random, entangled coil-like structures. After cooling and settling down, some chains will remain in this disordered amorphous state, whereas others will align themselves and stack on top of each other in crystalline regions (Crawford & Quinn, 2017). Depending on the manufacturing process and the degree of branching, 10-80% of the polymer can be in a crystalline state (Crawford & Quinn, 2017).

Crystalline regions have stronger intermolecular forces keeping them together. Hence, it is considered more difficult to degrade highly crystalline PE, as opposed to amorphous PE (Wei & Zimmerman, 2017). Since crystallinity is strongly correlated with the amount of branching (i.e., more branches means lower degree of crystallinity), LDPE is considered to degrade faster than HDPE.

1.2.4 PE is a polydisperse polymer

The molecular weight of a CH_2 motif is around 14 g/mol. Thus, a single molecule with a thousand repetitions of this motif has a molecular weight of approximately 14,000 g/mol. However, during the production of PE, many polymer chains of differing lengths are synthesized, and there thus exists a distribution of molecular weights. Polymers where every chain is of equal length and mass, are called homodisperse polymers, whereas polymers with variable chain lengths are called polydisperse (Whisnant, 2025). PE, unless it has been purified, is a polydisperse polymer.

Two quantities are often used to describe the molecular weight of a given polymer material; number-average molecular weight (\bar{M}_n) and weight-average molecular weight (\bar{M}_w) (Ronca, 2017; Whisnant, 2025). \bar{M}_n is calculated by summing the amount of molecules N_i in every given molecular weight fraction M_i , multiplied by the molecular weight, and dividing by the total number of molecules N_j , thus giving the simple arithmetic mean of the molar mass distribution (Whisnant, 2025).

$$\bar{M}_n = \frac{\sum_{i=0}^{\infty} N_i M_i}{\sum_{j=0}^{\infty} N_j}$$

\bar{M}_w is calculated in a similar way, except that higher molecular weight fractions are given more importance. The amount of molecules in a given molecular weight fraction are again multiplied by their respective molecular weight before dividing by the total mass of the sample, thus giving more weight to longer, heavier molecules (Whisnant, 2025). Hence, \bar{M}_w always exceeds \bar{M}_n .

$$\bar{M}_w = \frac{\sum_{i=0}^{\infty} N_i M_i^2}{\sum_{j=0}^{\infty} N_j M_j}$$

In commercially available research PE, \bar{M}_w ranges from 15,100 g/mol to 136,400 g/mol (Cuthbertson et al., 2024). The ratio \bar{M}_w/\bar{M}_n , also known as the polydispersity index, is a measure of how widely distributed the molecular weights are in the mixture. A ratio of 1 indicates that the distribution is narrow, whereas values higher than 1 indicate a wide distribution (Whisnant, 2025). For LDPE, polydispersity indices typically range from 20-50, while for HDPE values range from 4-15 (Ronca, 2017). In practice, these quantities are determined by techniques such as gel permeation chromatography (GPC), where the polymers in a mixture can be separated based on size. Molecular weight also influences resistance to degradation, with high molecular weight PE being more difficult to degrade.

1.3 Degradation of PE

For the purposes of this thesis, degradation equals depolymerization; the chemical transformation of polymer to oligomers or monomers. This can be the initial monomer of which the plastic was

made (ethylene gas in the case of PE), or a different low molecular weight compound. Complete degradation entails that the entire polymer is converted to molecules of at most 30 carbons long, which can then be metabolized or repurposed. Partial degradation entails that some of the material has depolymerized, but the majority still exists in its polymer form.

Once disposed in the environment, a combination of abiotic and biotic factors is thought to initiate the degradation process. UV exposure, fluctuating temperatures, mechanical stresses, as well as various life forms all contribute to the partial degradation of plastics, albeit at a rather slow rate (Chamas et al., 2020; Wei & Zimmerman, 2017). PE is particularly susceptible to photo-oxidation, which causes the formation of free radicals, incorporation of oxygen, and subsequent chain scission through Norrish type 1 and type 2 reactions (Albertsson et al., 1987; Hawkins, 1984). Waves and collisions with rocks cause macroplastic debris to fragment into smaller particles called microplastics and nanoplastics, thereby increasing the available surface area (Crawford & Quinn, 2017).

Micro-organisms then proceed to colonize the macro- and microplastics, further accelerating the degradation process through metabolic activity, production of reactive oxygen species (ROS), and the secretion of enzymes (Crawford & Quinn, 2017). Larger organisms also contribute through ingestion, where the plastic is exposed to acidic environments and gut microbes. In a second stage, the breakdown products of these processes are able to pass the cell membrane, where they can be mineralized to biomass, H_2O and CO_2 . Indeed, a ten-year experiment on PE biodegradation indicated that it degrades faster in a living environment than in a sterile one (Albertsson & Karlsson, 1990). However, even in this period of ten years, unpretreated radioactively labeled PE only lost 0.3% of its weight to $^{14}CO_2$. Adding photosensitizers and pretreating with UV for 42 hours increased the weight loss to 8.4%.

In this thesis, the focus is on isolated and purified enzymes acting on PE. We are not concerned with reports of living cells growing on PE, which would be termed biodegradation. The distinction also affects end products: enzymatic degradation yields oligomers or monomers, while biodegradation can lead to complete mineralization into biomass, H_2O and CO_2 .

1.3.1 Experimental evidence of enzymatic degradation

Enzymes currently known to degrade PE are limited to a handful of oxidoreductases in the Enzyme Commission (EC) category 1, and mostly rely on some form of pretreatment. To the author's knowledge, no reports exist on the complete enzymatic degradation of PE. Furthermore, the limited amount of publications treat LDPE, which is considered easier to degrade due to its branching properties and reduced crystallinity. The enzymatic degradation of HDPE has, to the author's knowledge, not been reported.

PlasticDB and PAZy

Two key resources exist for gathering information on enzymes involved in plastic degradation; (i) PlasticDB (Gambarini et al., 2022), and (ii) The Plastics Active Enzyme database (PAZy) (Buchholz et al., 2022).

PlasticDB presents a general overview of all publications, (micro-)organisms, and enzymes involved in plastic degradation. Both natural and synthetic polymers are included. Searching this database for PE returns 129 results, which also includes reports on biodegradation (i.e., without specifically mentioning an isolated enzyme). One can also query for proteins only, and for PE, this returns four results; one alkane hydroxylase and three copper oxidases. Upon closer inspection, the alkane hydroxylase was not isolated and tested on PE. Rather, it was recombinantly expressed in *E. coli* growing on PE (Gyung Yoon et al., 2012). Hence, this publication was dismissed. One copper oxidase is described by A. Zhang et al. (2022), and the other two are described by Zampolli et al. (2023).

Figure 1.3 shows the amount of species and proteins reported to degrade a certain plastic in this database. It can be seen that many micro-organisms degrading PE and LDPE have been identified, while a relatively limited number of proteins exist. This thesis strives to fill this gap.

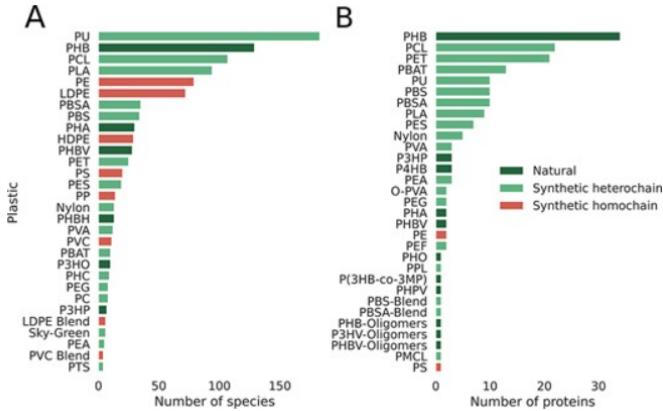


Figure 1.3: Amount of species and proteins listed in the PlasticDB per polymer type. Notice the relatively large amount of species reported for PE and LDPE, compared to the small amount of proteins. Figure taken from Gambarini et al. (2022)

The PAZy database is a more valuable resource, since it is manually curated and exclusively reports enzymes for which sufficient experimental evidence of plastic degradation exists. In total, PAZy contains 255 unique enzymes known to act on various plastics (both fuel-based and from renewable resources), 124 of which act on PET. For PE, it lists two "unspecific oxidative enzymes" belonging to the family of laccases/laccase-like multicopper oxidases (LMCO, EC 1.10.3.2), the same ones found in the plasticDB described by Zampolli et al. (2023). Together, these databases list three isolated enzymes from two studies involved in PE degradation. All of them are LMCOs.

Literature review

Aside from the above two databases, a literature study was performed primarily using Google Scholar and PubMed (see methods section). It was found that, aside from LMCOs, lignin peroxidases (LiP, EC 1.11.1.14) and manganese peroxidases (MnP, EC 1.11.1.13) were also often reported (Tournier et al., 2023; Wei & Zimmerman, 2017). LMCOs and peroxidases are both metalloenzymes involved in the degradation of lignin (Youn et al., 1995). This is no coincidence. Lignin, a complex polymer composed of a variety of phenolic monomers, is the second most abundant natural polymer after cellulose. It is found in the cell walls of plants where it provides rigidity and a protective chemical barrier. LMCOs and peroxidases are secreted by white-rot fungi and bacteria to break down lignin polymers (Granja-Travez et al., 2018; Youn et al., 1995). It is thought that they can also cleave lignin C-C bonds (Andler et al., 2022). However, these cannot be compared to the C-C bonds found in PE, as they are in a completely different chemical context and exhibit higher bond dissociation energies (Andler et al., 2022).

Table 1.1 lists all retrieved publications in which PE was treated with an isolated enzyme. This is meant as an exhaustive list, but it is certainly possible that some publications have been missed. The techniques used to measure degradation are explained in appendix C.2. It can be seen that all experiments either use an LMCO or a peroxidase, with LMCOs being the most prominent. There is, however, a clear tendency towards milligram amounts of LDPE (given that the authors even mention the type of PE to begin with), and none of the publications report on a complete depolymerization. There also seems to be no standardization of pretreatment, reaction conditions, or degradation assays, which makes it difficult to compare the results. Additionally, the publications often lack detailed reports on PE properties such as \bar{M}_n and \bar{M}_w , crystallinity and branching. Only four papers mention under which identifier the protein sequence can be found in a public database (A. Zhang et al., 2022; Oiffer et al., 2024; Zampolli et al., 2023; Y. Zhang et al., 2023). It can thus be noted that experimental evidence is rather scarce.

The first report found on enzymatic PE degradation appeared in 1998 (Iiyoshi et al., 1998). Researchers isolated MnP from the lignin-degrading fungus *Trametes versicolor* and treated PE

Table 1.1: Experimental evidence of enzymatic PE degradation.

Ref.	PE type	Amount	Enzyme	Pretreatment	Reaction conditions	Degradation assays ¹	Organism	Seq. ID
Iiyoshi et al., 1998	PE strips ²	1 cm x 6 cm x 100 µm	MnP	None	8d, 37°C Tween-80 GOD ³	RE RTS GPC	<i>T. versicolor</i>	NA
Fujisawa et al., 2001	PE strips	1 cm x 6 cm x 100 µm	LMCO	None	3d, 30°C Mediator (HBT)	RE RTS GPC	<i>T. versicolor</i>	NA
Santo et al., 2013	LDPE films	3 cm x 3 cm x 0.2 mm 100 mg	LMCO	UV 60h	2w, 37°C	WL ATR-FTIR DSC GPC	<i>R. ruber</i> C208	In paper
Yao et al., 2022	LDPPE films	3 cm x 3 cm x 20 µm 50 mg	LMCO	UV 3d 70°C	5d, 30°C Mediators	FTIR HT-GPC GC-MS SEM	<i>B. acclada</i> + <i>B. subtilis</i>	NA
A. Zhang et al., 2022 [∞]	PE particles PE films	1000 µm (particles) 30 mm x 20 mm (films)	LMCO	None	1d, 30°C (particles) 7d, 30°C (films)	WL ATR-FTIR SEM WCA XRD	<i>Psychrobacter</i> NJ228	sp. OM891104
Y. Zhang et al., 2022	LDPPE films	1 cm x 1 cm	LMCO	UV 4h 50°C	12h, 42°C	WL ATR-FTIR SEM WCA GPC	<i>A. baumanii</i>	NA
Y. Zhang et al., 2023	LDPPE films	4 cm x 4 cm	LMCO	None	8w, 28°C	WL FTIR XPS SEM	<i>L. fujiformis</i>	WP_193831439.1
Zampolli et al., 2023	LDPE powder	1% in 1 ml	LMCO	None	48h, 60°C	FTIR GC-MS	<i>R. opacus</i> R7	WP_128643863.1
Oiffer et al., 2024	LMWPE powder	15 g	MnP + others	mCPBA Ultrasonic waves	24h, 20°C 80°C	AFM GC-MS	<i>R. picketti</i>	WP_012436375.1

¹ The reader is referred to the list of abbreviations and appendix C.2 for this column.² If 'PE' is written in the table, this means that the type of PE was not mentioned by the authors.³ Tween-80 is a surfactant. GOD is glucose oxidase.

strips for up to eight days at 37°C. It was shown that Tween-80, a surfactant, improves performance, and that glucose oxidase (used as a source for H_2O_2 for the peroxidase to work) was not necessary. The authors report on reductions in relative elongation (RE) from 100 to 0 and reductions in relative tensile strength (RTS) from 100 to 52, as well as reductions in \bar{M}_w and \bar{M}_n from 716,000 g/mol to 89,500 g/mol, and 108,000 g/mol to 38,800 g/mol respectively.

The first experiments with an LMCO, also isolated from the fungus *T. versicolor*, appeared three years later (Fujisawa et al., 2001). Unpretreated PE membranes of several centimeters long were incubated with the purified LMCO and 1-hydroxybenzotriazole (HBT). Here, HBT functioned as a synthetic mediator, meaning that the LMCO first oxidizes HBT, and the free radical intermediates go on to further oxidize PE. The enzyme thus acts indirectly. It was, however, shown that the LMCO can also act without a mediator, albeit with reduced effectiveness. The reaction was performed at 30°C for three days, with the LMCO and HBT added each day. Results showed that the PE membrane could visually be seen to disintegrate, and the treated membrane exhibited no elongation after treatment. RTS dropped by 60%, and \bar{M}_w and \bar{M}_n dropped from 242,000 to 28,300, and 36,600 to 12,100 respectively after three days.

These studies show that the mechanical properties of PE are altered and molecular weight can be reduced after treatment with a peroxidase or an LMCO. However, both authors fail to report on the type of PE used, and the process either relies on the presence of a surfactant or the use of mediators.

Santo et al. (2013) isolated crude extracts containing an LMCO from the bacterium *Rhodococcus ruber* and tested its activity towards 0.2 mm thick LDPE films (average molecular weight of 191,100, the authors do not mention if this is a weight- or number average) after UV pretreatment. Attenuated total reflection Fourier-transform infrared spectroscopy (ATR-FTIR) showed a 40% increase of the carbonyl peak, and GPC showed a 20% and 15% reduction in \bar{M}_w and \bar{M}_n respectively, with an overall weight loss of 2%. It was also found that *T. versicolor*'s LMCO did not affect the FTIR spectrum, which is in disagreement with Fujisawa et al. (2001). The IR spectra obtained are, however, not shown. Additionally, differential scanning calorimetry (DSC) showed an increase in crystallinity for the films incubated with living *Rhodococcus* cells and LDPE as the sole carbon source. This suggests that amorphous regions are degraded first, resulting in a relative increase in crystallinity. Lastly, the authors mention that chain breakage likely occurs at branches rather than at the center of the polymer, supported by the insignificant changes in polydispersity index (Santo et al., 2013).

Yao et al. (2022) isolated one fungal and one bacterial LMCO to determine their activity towards UV-pretreated LDPE films ($\bar{M}_n = 777$, $\bar{M}_w = 25,992$) with the addition of three different synthetic mediators (HBT, ABTS and TEMPO). Scanning electron microscopy (SEM) pictures of the treated film surfaces showed clear signs of cracking and abrasion compared to the control. ATR-FTIR showed new peaks of oxygen-containing groups, and GPC showed a reduction of \bar{M}_w up to 52% with the TEMPO mediator, whereas other mediators showed increases in \bar{M}_n . Lastly, the authors also used gas chromatography and mass spectrometry (GC-MS) to identify the breakdown products, and mostly found alcohols, esters and ketones of three to twenty carbons long, suggesting that some depolymerization has occurred. However, a clear discrepancy could be seen between the chemical formulas and compound structures presented in the table reporting on the GC-MS end products, leading to skepticism regarding the trustworthiness of the paper.

A. Zhang et al. (2022) purified a cold-adapted LMCO from *Psychrobacter* living in Antarctic sea ice and tested its activity towards unpretreated PE particles of 1000 μm in diameter and strips of 30 mm x 20 mm. This enzyme was also reported in the PlasticDB, but not in the PAZy database. A weight loss of 13.2% was observed with the purified enzyme. FTIR spectra showed barely noticeable increases in carbon double bonds, C=N and C=O peaks. Water contact angles (WCA) dropped by roughly 20°, and SEM showed several surface modifications compared to the control. X-ray diffraction (XRD) highlighted a decrease in crystallinity from 79% to 55%, an unexpected result in disagreement with Santo et al. (2013), suggesting that the enzyme can attack crystalline regions.

Y. Zhang et al. (2023) developed a computer-aided screening process to identify enzymes from the genomes of known PE-degrading micro-organisms. They hypothesized that the enzyme must (i) contain a metal catalytic center, and (ii) possess an open substrate binding pocket. After this

screening based on structural characteristics, nine enzymes were expressed and functionally characterized. One enzyme, an LMCO from *Lysinibacillus fusiformis*, was chosen to be tested on PE. After eight weeks of treatment (considerably longer than all other papers), SEM showed visible changes in surface morphology. FTIR again showed an increase in C=C, C=O, and C=N peaks, further confirmed by X-ray photoelectron spectroscopy (XPS) experiments. A weight loss of 3.75% and 2.79% was reported for treatment with the crude cell extract and the purified enzyme respectively. The activity towards PE is explained by the presence of flexible loops near the open substrate binding site (as was enforced by their computational screening).

Meanwhile, Zampolli et al. (2021) already performed transcriptomic experiments on *Rhodococcus* bacteria growing on LDPE as a sole carbon source (Zampolli et al., 2021). Several up regulated genes were identified, including one LMCO. Two other putative LMCOs in the genome that were not up regulated were also picked up by the authors. The two dormant LMCOs, *LMCO2* and *LMCO3*, were chosen for further analysis in a subsequent publication (Zampolli et al., 2023). Why these LMCOs were not up regulated in the initial experiment remains unclear.

LMCO2 and *LMCO3* were incubated at 60°C for two days with unpretreated commercial LDPE powder ($M_n = 1700, M_w = 4000$). FTIR revealed the introduction of carboxyl functional groups and a reduction in the abundance of CH-bands. GC-MS indicated that end products were mainly carboxylic acids between 8 and 30 carbons long, as well as alcohols and ketones of varying length. This is in disagreement with Yao et al. (2022), who did not obtain carboxylic acids. However, as previously mentioned, their results are questionable. Additionally, the absence of any mediators in this experiment, and the elevated temperatures might have caused the difference in end products. The most striking difference in the GC-MS spectrum compared to the control was the increase in C8 and C16 carboxylic acids.

The authors further predicted the three-dimensional structure of both LMCOs. They propose that a hydrophobic methionine-rich (MR) loop in *LMCO2* facilitates binding to the PE polymer, and draws the catalytic center in close proximity to PE (Zampolli et al., 2023). This MR region is 39 amino acids long, and contains 7 Met residues (Figure 1.4). It can be divided up into two parts; a nonpolar part where the Met residues aggregate, along with numerous flexible Gly residues in between. The second part contains no Met residues and more polar amino acids such as Asp and Glu. Notably, the authors also predicted the structure of the cold-adapted LMCO from *Psychrobacter* mentioned above (A. Zhang et al., 2022), and saw that the T1 copper site is similarly surrounded by numerous Met residues, further supporting the idea that MR regions might be involved in PE binding. Both structures are shown in Figure 1.4

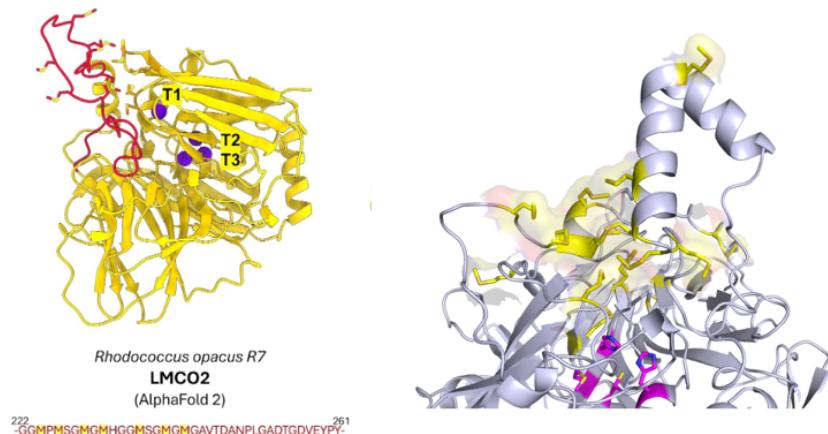


Figure 1.4: **Left:** Predicted structure of *LMCO2* with its MR region depicted in red. Purple spheres represent the copper ions. T1 subtracts the electron from the substrate. **Right:** Predicted structure of the cold-adapted LMCO from A. Zhang et al. (2022). Copper binding ligands are depicted in purple. In both cases, an MR region close to the catalytic site is thought to facilitate the binding to PE. Both figures adopted from Orlando et al. (2025). Figure on the right can be found in the supplementary material.

A possible reaction mechanism supported by density functional theory (DFT) was also proposed. Results do not rule out that the LMCO functions by means of an electron-proton transfer, resulting in the formation of a free radical carbon, which is subsequently attacked by water to form an alcohol. This alcohol is then further oxidized by the LMCO to a ketone. This process is depicted in Figure 1.5. The authors note that their calculations are far from an exhaustive description of the process, and the formation of carboxylic acids, their main end product, is not explained by their reaction scheme.

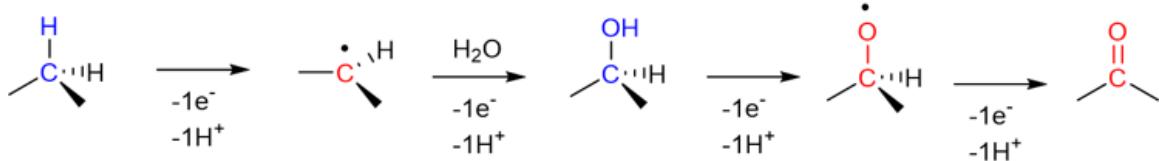


Figure 1.5: Possible reaction mechanism of PE by an LMCO enzyme. Figure taken from Zampolli et al. (2023), supplementary material S8

In a subsequent paper, Orlando et al. (2025) performed docking and molecular dynamics (MD) simulations using *LMCO2* and two other known PE-degrading LMCOs to explore if the MR region is indeed capable of stabilizing the interaction with a $C_{100}H_{202}$ molecule. Results confirmed their hypothesis, and indicate that the MR region in *LMCO2* enhances the formation of a stable complex with their model PE compound, allowing the catalytic pocket to remain close to the substrate, and electron-proton transfer to occur. Additionally, several acidic residues in this MR loop were also found to remain close to the PE model molecule, which were presumed to be necessary for proton transfer. Hence, the presence of a hydrophobic MR region seems to be a key driving force for PE degradation. It should, however, be noted that the PE model compound is simply a single $C_{100}H_{202}$ molecule, contains no branching points, and is not embedded in a context of multiple polymers.

To date, the most successful account of the chemo-enzymatic degradation of PE has been reported by Oiffer et al. (2024). This is the first publication found to scale the process up to several grams (15 g) of low molecular weight PE (LMWPE, $M_n = 1800$ g/mol). In their experiments, a combination of chemical, physical and enzymatic treatments were employed. First, meta-chloroperoxybenzoic acid (mCPBA) was used for oxidation purposes, along with ultrasonic treatment. After this pretreatment, an enzymatic cascade followed; starting with a mutant MnP to further increase oxidation, succeeded by an alcohol dehydrogenase (ADH), Baeyer-Villiger mono-oxygenase (BVMO), and finally a lipase (Figure 1.6).

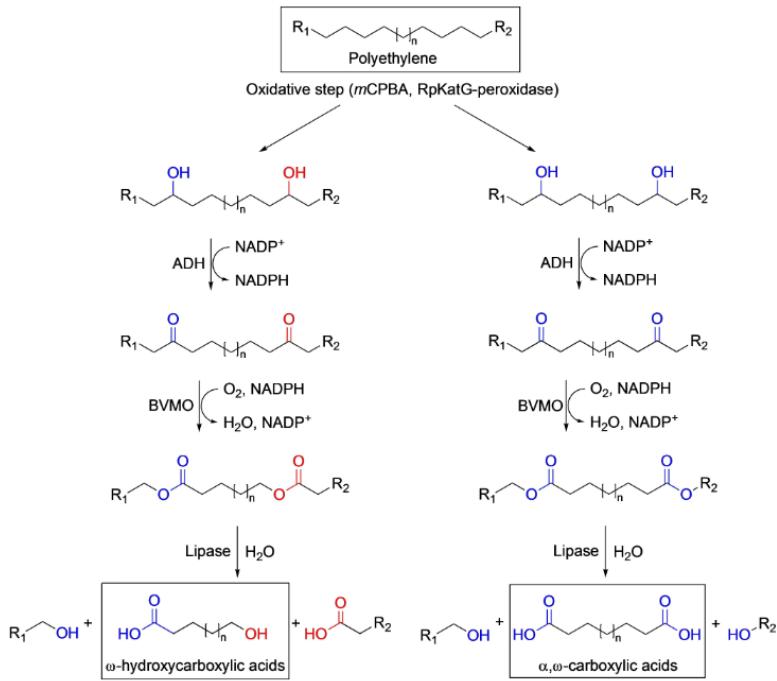


Figure 1.6: Schematic illustration of the enzymatic cascade used by Oiffer et al. (2024) to depolymerize LMWPE into diacids and hydroxycarboxylic acids. mCPBA = meta-chloroperoxybenzoic acid, ADH = alcohol dehydrogenase, BVMO = Baeyer-Viller mono-oxygenase. Figure taken from Oiffer et al. (2024)

Results showed a 27% weight loss of 6.9 g of pre-oxidized LMWPE powder, suggesting that 1.9 g of pre-treated LMWPE could be depolymerized by the enzymatic cascade. It was predicted that diacids and hydroxycarboxylic acids would be the end product of this enzymatic treatment, and this was indeed confirmed by GC-MS. Other end products such as small alcohols and diols were also detected. A higher reduction of mass was observed by gravimetric analysis than was predicted by GC-MS, suggesting that higher molecular weight products not detectable by GC-MS were also present. Lastly, atomic force microscopy (AFM) showed that the particle size decreased after treatment.

Although these experiments show that enzymatic degradation of several grams of pretreated LMWPE is possible, it still relies on the use of a toxic chemical (mCPBA) and physical pretreatment to perform the initial oxidation. Ideally, the process is purely enzymatic, and an enzyme able to oxidize PE without any pretreatment (or at most a minimal amount of UV) would greatly enhance the attractiveness of this approach. The authors comment that further research should focus on identifying much more potent peroxidases. Additionally, this study used low-molecular weight PE with an M_n of 1800 g/mol, which means that, on average, the polymer chains are 128 carbons long. The enzymatic degradation of higher molecular weight PE thus remains unsolved. The authors also do not report on the amount of branching in their PE samples.

To conclude, it seems that there is sufficient evidence to believe that peroxidases and/or LMCOs can bring about the partial degradation of small amounts of LDPE through an oxidative mechanism. Some publications suggest that no pretreatment or mediators are necessary, and that the LMCO can both oxidize and cause depolymerization (A. Zhang et al., 2022; Zampolli et al., 2023; Y. Zhang et al., 2023), whereas Oiffer et al. (2024) require both physical and chemical pretreatments followed by an enzymatic cascade after the initial oxidation. Since all publications report on LDPE, it remains unclear if oxidation can take place along the main chain, or only at branching points. The exact mechanism of chain cleavage also remains unclear. Most importantly, Zampolli et al. (2023) and Orlando et al. (2025) suggest that a hydrophobic MR binding region facilitates the formation of an LMCO-PE complex, drawing the catalytic center close to PE and allowing for electron-proton transfer to occur. However, only two LMCOs presumably functioning this way are known today (A.

Zhang et al., 2022; Zampolli et al., 2023). Hence, it seems that the presence of such a region can be set as an important criterium to identify PE-degrading LMCOs, which is precisely what this work plans on doing.

1.3.2 Metagenomic insights

This section aims to strengthen the idea that PE-degrading enzymes can be found in environments enriched in plastic debris. Indeed, the PET-degrading hydrolase produced by *Ideonella sakaiensis* was found by sampling plastic waste dump sites near a recycling facility (Yoshida et al., 2016). Upon culturing the bacterium, it was found to produce two enzymes that completely convert PET into its monomer building blocks. Hence, a similar approach can be applied to search for PE-degrading enzymes.

The plastisphere

There is no doubt that various micro-organisms colonize marine and terrestrial plastic debris (Amaral-Zettler et al., 2020). The ensemble of microbes inhabiting this plastic ecosystem has been termed "the plastisphere" (Zettler et al., 2013). However, whether or not they are actively depolymerizing, assimilating and metabolizing the polymers remains an open question (Bryant et al., 2016). Either they opportunistically attack parts that have been affected by abiotic factors, or they feed on breakdown products or other compounds adsorbed to the plastic surface. Alternatively, they simply prefer a surface-attached lifestyle, and a photosynthetic symbiont is providing nutrients.

The first metagenomic study on marine plastic debris sampled the North Atlantic Ocean, and revealed the presence of a rich microbial community comprised of both eukaryotes and prokaryotes (Zettler et al., 2013). The two samples, which contained the plastics PE and PP, exhibited community compositions significantly different from that of uncontaminated seawater, suggesting that plastic debris selects for the presence of specific microbes. Some operational taxonomic units (OTUs) specific to both PP and PE were also identified. Interestingly, these contained micro-organisms known to degrade hydrocarbons (e.g., cyanobacteria from the genus *Phormidium*). SEM images of such communities are shown in Figure 1.7. However, this study did not reveal any particular enzymes possibly involved in PE degradation.

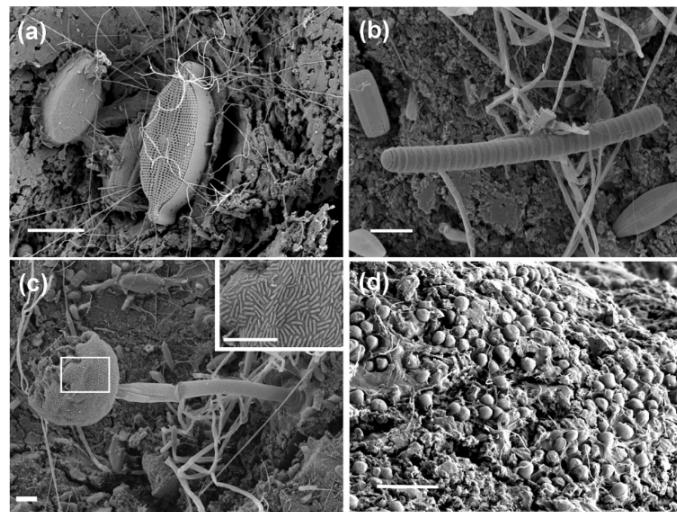


Figure 1.7: SEM images of plastic debris found in the North Atlantic Ocean. Various diatoms and filamentous bacteria co-inhabiting the surface are shown. Scale bars are 10 μm . Figure taken from Zettler et al. (2013).

Bryant et al. (2016) found similar results concerning microbial compositions in the North Atlantic Ocean. Additionally, numerous putative xenobiotic degradation genes were enriched in their

samples. For example, the ring-cleaving enzyme homogentisate 1,2-dioxygenase, which is thought to be involved in the degradation of PAHs (one of the POPs known to attach to microplastics, as discussed in section 1.1) was identified. Other enzymes primarily included oxygenases and reductases. However, data from this study actually suggests that the microbial community members primarily obtain nutrients from filter-feeding bryozoans, other eukaryotes and autotrophs (Bryant et al., 2016). Hence, the involvement of these genes in plastic degradation is merely speculative.

Metagenomic studies have also been performed in plastic dump sites from terrestrial origin. A solid waste dump site in Gujarat, India contained a rich microbial community primarily dominated by bacteria (R. Kumar et al., 2021). However, one of the authors' goals was to specifically explore the bacterial diversity, and they did not sequence internal transcribed spacers (ITS) to detect fungi. Nevertheless, the dominant phyla were *Proteobacteria* (15.93% to 24.08%), *Firmicutes* (11.36% to 24.05%), *Bacteroidetes* (4.8 to 22.79%), and *Actinobacteria* (1.63% to 16.95%). These results are, according to the authors, in agreement with other studies of bacterial diversity in landfills (R. Kumar et al., 2021).

The researchers also identified numerous genes possibly involved in the degradation of PE, PET and PS. Specifically for PE, laccase (EC 1.10.3.2), alkane 1-monooxygenase (alkM, alkB1_2, EC 1.14.15.3), long-chain alkane monooxygenase (ladA, EC 1.14.14.28) and midchain alkane hydroxylase (MAH1, CYP96A15) were abundant in multiple samples. Other enzymes in EC categories 2.-.-. and 4.-.-. were also present. This finding is in accordance with LMCOs being involved in plastic degradation. Additionally, alkane oxygenases, involved in the terminal oxidation of short- to medium length alkanes, also seem to be relevant.

Similar experiments have been performed at another dump site near Rajasthan, India (A. Kumar et al., 2024). Samples were taken at various depths in the garbage pile. This time 16S rRNA, 18S rRNA and ITS genomic markers were included. Results now revealed a complex microbial community composed of bacteria and fungi. The bacterial diversity was similar to the study mentioned above, and among the fungi, *Ascomycetes* were the dominant phylum.

One sample at 0.9 m depth was chosen for functional gene analysis. Primarily members of EC 2.-.-. and 3.-.-. categories dominated, and no LMCOs were found. However, peroxidases and various alkane oxygenases were also found to be enriched, suggesting that LMCOs are not the only enzymes involved in PE degradation. Figure 1.8 summarizes the author's proposed pathway for the degradation of four plastics.

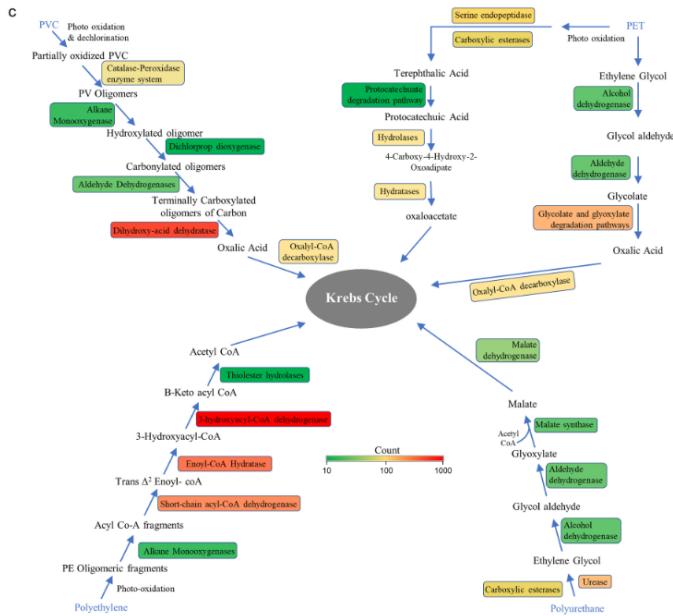


Figure 1.8: Proposed plastic degradation pathways and enriched genes found at 0.9 m depth of a solid waste dumping site in India. Red indicates higher abundance in the whole metagenome analysis. Figure taken from A. Kumar et al. (2024)

Notably, the first step in the proposed PE degradation pathway is attributed to photo-oxidation, producing "PE oligomeric fragments". However, at a depth of 0.9 m, it seems unlikely that UV is a major contributor to chain fragmentation. On the other hand, at some point in the past, the waste at this depth was exposed to the surface. The authors mention technical workers at the dump site saying that the pile grows at about 0.6 m per year, so this would have been approximately one and a half years before the samples were taken. Photo-oxidation might have taken place then, with a minimal amount of chain fragmentation. Nevertheless, an enzyme able to transform PE polymers to oligomers, the most important step, is lacking.

Another major caveat is, as is the case with almost any metagenomic study of this kind, that many different sorts of waste are present in these samples, and it seems premature to attribute the abundance of certain enzymes in the sample to the degradation of specific types of plastic.

The MGNIFY database

The MGNIFY database hosts metagenomic data from all kinds of studies (Richardson et al., 2023). Automated pipelines are in place for assembly, annotation of genes, taxonomic identification etc. This database will be used as a source of information later on. At the time of writing, searching MGNIFY for the word 'plastic' returns 14 studies, most of them from marine biomes. Both studies based on amplicon sequencing and shotgun metagenomics were found in the results. In the first case, only a specific region of interest is amplified (mostly 16S rRNA). In the second case, whole genomes are sequenced, which is most relevant for this thesis. However, in some cases the focus is on a specific type of plastic such as PET. Such studies were also dismissed.

One study, MGYS00005970, is solely confined to PE. Two PE films and one control were incubated in marine conditions for six months, followed by whole genome metagenome sequencing. Apart from the information that the study was performed by the CSIR- National Institute of Oceanography in Panaji, India, no corresponding publication for this study could be found.

MGNIFY also provides pre-computed statistics for each sample in the 'analysis' section such as taxonomic information, GO terms, InterPro annotations etc. Figure 1.9 shows the most abundant GO terms in one sample of this study. Molecular functions such as metabolic process, transport, and small molecule metabolic process are particularly enriched. For biological processes; catalytic,

hydrolase and oxidoreductase activity are notably abundant terms, and membrane terms dominate for cellular component. These statistics suggest that oxidative activity is taking place, and low molecular weight breakdown products are transported across the cell membrane.

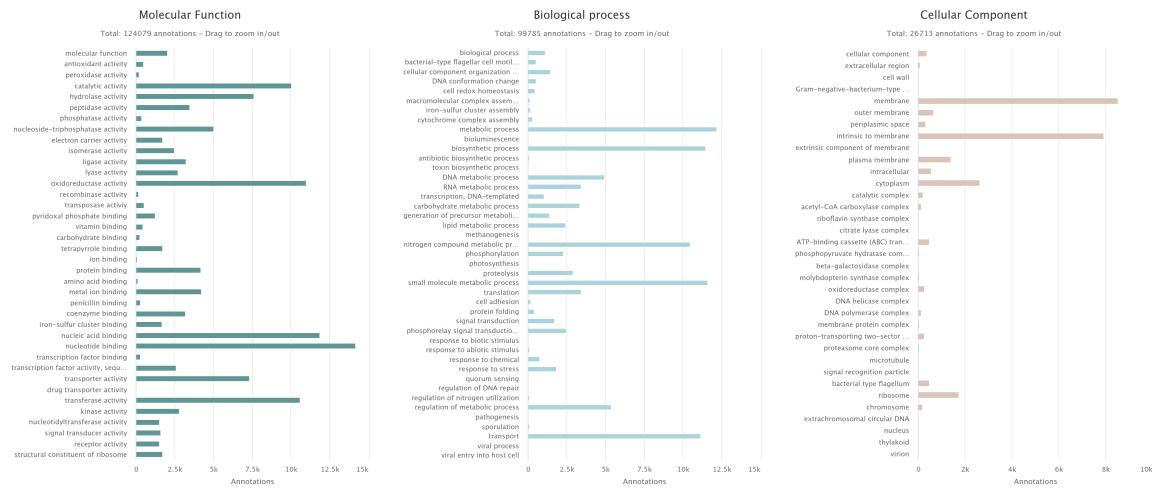


Figure 1.9: GO term annotation for MGNIFY study MGYS00005970. Analysis ID MGYA00594112.

The study performed by Bryant et al. (2016) in the North Atlantic Ocean can also be found in the MGNIFY database under the accession MGYS00005704. This study contains samples from six different locations containing both PE and PP. Each location was sampled twice for particles smaller than 5 mm and larger than 5 mm. The samples in this study show similar GO term annotations, with transport, oxidoreductase activity, and membrane terms being particularly abundant. The taxonomic composition, however, varies significantly among the different samples.

1.4 Laccase-like multicopper oxidases

Based on the previous information, laccase-like multicopper oxidases (LMCOs) seem to be the dominant enzymes involved in PE-degradation. Hence, this section aims to define and describe this family of enzymes.

First, some clarification of terminology is required, since the name 'laccase' has been criticized by some authors. It remains unclear what exactly a laccase is, and why the term 'laccase-like' has been coined. We will then describe the biological distribution and function, as well as structural properties of LMCOs. Lastly, attention is brought to the role of MR regions in these enzymes, and their relevance in the degradation of PE.

1.4.1 Clarification of terminology

Laccases

Historically, the name 'laccase' is derived from the Japanese 'lacquer tree' *Rhus vernicifera*, in which the enzyme was first described by Japanese researchers in 1883 (Lontie, 1984b). Similar to harvesting latex from rubber trees, several cuts are made in the bark of lacquer trees, after which a white phenol-rich liquid (primarily containing urushiol) seeps out. This sap also contains laccase enzymes. Once in the presence of oxygen (the terminal electron acceptor for laccases), the phenols are rapidly oxidized to phenoxy radicals and polymerize to form a hardened wax-like substance which is then harvested and processed to coat various materials (Lontie, 1984b). The general reaction mechanism of a laccase is depicted in Figure 1.10 (Martin et al., 2024).

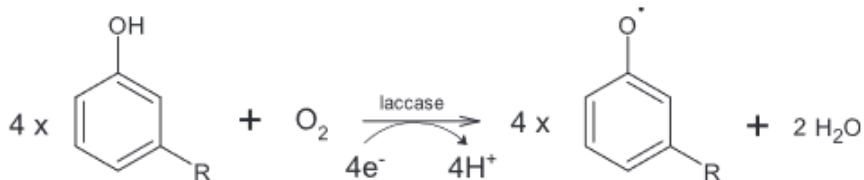


Figure 1.10: General reaction mechanism of an LMCO. Phenols are oxidized to phenoxy radicals, and are either converted into ketones, or polymerize with other phenoxy radicals. Figure taken from Martin et al. (2024)

Laccases (EC 1.10.3.2, oxidoreductases - acting on diphenols and related substances - with oxygen as acceptor - laccase) are defined by the BRENDA database as "A group of multi-copper proteins of low specificity acting on both o- and p-quinols, and often acting also on aminophenols and phenylene-diamine. The semiquinone may react further either enzymically or non-enzymically." (Chang et al., 2021). Laccases are most importantly characterized by (i) the presence of four positively charged copper ions, categorized as type 1, 2 and 3 based on their spectroscopic properties, (ii) the ability to couple the oxidation of a wide variety of substrates with the subsequent reduction of dioxygen to water, and (iii) their potential to be used as biocatalysts in industrial processes (e.g., delignification of pulp in the paper/textile industry, bioremediation of dyes).

The EC classification of laccases has been criticized due to their unusually broad substrate scope (Reiss et al., 2013). Laccases are currently considered as a subgroup of multi-copper oxidases (MCOs) with some activity towards phenolic compounds (as originally found in the lacquer tree). Preferably mono-, di- and polyphenols are oxidized (Lontie, 1984b). In experimental settings, the model substrates ABTS, HBT, DMP or TEMPO are often used as a test to classify laccases (Yang et al., 2017).

However, many other organic and inorganic substrates are also oxidized by the so-called laccases (e.g., diamines, metal ions, benzenethiols, and in our case; polyethylene) (Reiss et al., 2013; Yang et al., 2017). In the past, different authors have classified the same enzyme from the same organism as both laccase and bilirubin oxidase, and in many cases the preferred substrates remain unknown (Reiss et al., 2013). Laccases are also known for indirectly oxidizing certain molecules by using their substrates as mediators (laccase oxidizes molecule X, and molecule X goes on to oxidize molecule Y). Additionally, other enzymes in the MCO family essentially contain the same structural folds and catalytic mechanism, but with slightly different substrate preferences. Hence, the classification of laccases by the reactions they catalyze and the substrates they bind (i.e. by EC number) seems dubious at least.

Reiss et al. (2013) have suggested to use the name 'laccase-like multicopper oxidase' (LMCO) to avoid confusion and prevent premature EC classification. This term will also be used in this thesis. In the following two paragraphs, LMCOs are defined more precisely, primarily based on structural characteristics and mechanisms of action rather than substrate specificity.

To identify key structural characteristics, it is important to understand folding patterns in MCOs. All MCOs contain at least two of the characteristic cupredoxin domains that were first described in the mono-domain copper enzymes azurin and plastocyanin found in higher plants (Hakulinen & Rouvinen, 2015). In the MCO family, primarily two-domain MCOs (2dMCOs), 3dMCOs and 6dMCOs are distinguished (Gräff et al., 2020). Laccase-like MCOs are typically 500 amino acids long, possess three cupredoxin folds, and the four copper ions are arranged in a strongly conserved pattern within these domains. The MCOs relevant to PE degradation are all 3dMCOs.

The nature of these folds and the copper site architectures will be discussed in the following subsection. For now, it suffices to know that the enzymes of interest in the context of PE degradation are MCOs with (i) three cupredoxin domains, (ii) four copper ions, and (iii) the ability to combine substrate oxidation with oxygen reduction. The term LMCO will be used for enzymes with these qualities, disregarding their preferred substrates and EC classification.

Other enzymes in the MCO family

According to the InterPro database, MCOs include laccase (EC 1.10.3.2, mostly 3dMCO), nitrite reductase (EC 1.7.2.1, 2dMCO), ascorbate oxidase (EC 1.10.3.3, 3dMCO), and fission yeast fio1 (3dMCO) (Blum et al., 2025). Additionally, InterPro states that there are a number of similar proteins based on structure and sequence (e.g., the 3dMCO FET3 ferroxidase in baker's yeast), some of which have lost the ability to bind copper. Reiss et al. (2013) also add ferroxidase/ceruloplasmin (EC 1.16.3.1) to this selection. These enzymes are typically larger, contain six copper ions, and six cupredoxin domains (Bielli and Calabrese, 2002). Gräff et al. (2020) define yet another set of MCOs; they discard the nitrite reductases, and include the bilirubin oxidases (EC 1.3.3.5) instead. Some authors also include phenoxazino synthetases (EC 1.10.3.4) (Hakulinen & Rouvinen, 2015). There thus seems to be some confusion as to which members comprise the MCO family. However, laccases are considered as the dominant member.

According to the previous definition, LMCOs might thus also include enzymes that are technically not considered as a laccase by EC number, such as ascorbate oxidases and ferroxidases. However, they possess the same structural patterns (three cupredoxin domains and four copper ions) and catalytic mechanism (substrate oxidation and dioxygen reduction). The only difference is that they have learned to bind different substrates, which is precisely what makes them attractive in the context of PE degradation. My use of the term LMCO is thus not strictly confined to enzymes belonging to EC 1.10.3.2 (laccases).

1.4.2 Biological distribution and function

LMCOs can be found in all kingdoms of life. Depending on the species, different functions prevail. At the time of writing, UniProtKB hosts 64,385 proteins belonging to the multicopper oxidase family. 56,724 from eukaryotes (primarily plants and fungi), 7659 from bacteria, and 2 from archaea. By EC class, the main representatives are: laccase (13,924), nitrite reductase (6401), ferroxidase (3219), and ascorbate oxidase (1443). The biological role of LMCOs in each kingdom will now be briefly discussed. This section primarily aims to highlight the wide variety of functions LMCOs have adopted throughout evolution.

Plants

LMCOs can be isolated out of household peaches, tea leaves, cabbages, beetles, apples, tobacco and various other vegetables (Lontie, 1984b; Martin et al., 2024). Plant LMCOs are probably best understood in the lacquer tree *R. vernicifera* as described above.

More importantly, the main function of plant LMCOs is thought to involve the biosynthesis of lignin. Individual phenol-like lignin monomers are oxidized by LMCOs and spontaneously polymerize to form lignin deposits. To confirm this, experiments have been performed with *Arabidopsis* LMCO mutants, showing a 30% decrease in lignin content of the seeds (Martin et al., 2024).

Ascorbate oxidases can be found in higher plants where they catalyze the oxidation of L-ascorbate (vitamin C), although they show some activity towards other cyclic compounds such as catechols and flavonoids as well (Lontie, 1984b). The enzyme was first extracted from cabbage leaves, but its main sources are cucumber and zucchini green squash (Lontie, 1984b). This enzyme was long thought to have no advantageous function, since it reduces the concentration of one of the key antioxidants. Now it is known to play a role in maintaining oxygen and redox balance in the extracellular matrix, among other functions such as growth and hormonal signaling (Mellidou and Kanellis, 2024).

Fungi

Juxtaposing the role of lignification in plants, fungal LMCOs have adopted the role of lignin degradation. This function is particularly prevalent in white-rot fungi, and the idea is further supported by the absence of LMCOs in brown-rot fungi, which are unable to degrade lignin (Lontie, 1984b). An LMCO knockout from *Sporotrichum* was unable to degrade kraft lignin, and the ability was regained upon reversing the mutation (Youn et al., 1995). To date, several 100 LMCOs have been

isolated from fungi, primarily *Ascomycetes* and *Basidiomycetes* (Le Viet et al., 2025). Some of the first and best understood LMCO crystal structures come from *Trametes versicolor* and *Coprinus cinereus* (Ducros et al., 1998; Piontek et al., 2002).

In *Saccharomyces cerevisiae*, a ferroxidase termed FET3 is involved in iron homeostasis (Taylor et al., 2005). Instead of the usual preference to phenol-like compounds, this LMCO has adapted to bind Fe(II) and oxidize it to Fe(III), allowing it to be used by other iron-binding enzymes. This shift in substrate preference has been attributed to several iron-binding ligands near the catalytic center (Taylor et al., 2005).

Bacteria

The most extensively described LMCO to date comes from the bacterium *Bacillus subtilis*. Multiple crystal structures with bound substrates have been elucidated for this LMCO, as well as several mutational studies (Enguita et al., 2003; Hakulinen & Rouvinen, 2015). When this bacterium forms endospores, its outer layer is comprised of a complex mixture of proteins, one of them being an LMCO (cotA) thought to be responsible for the brown pigmentation and resistance against hydrogen peroxide and UV light (Enguita et al., 2003).

In *Escherichia coli*, an LMCO termed CueO, is expressed in the presence of copper (Roberts et al., 2002). Here, the LMCO is present in the periplasmic space, and is thought to be involved in copper homeostasis. Interestingly, this LMCO has an MR helix completely covering the catalytic site that has been observed to bind three other copper ions (Cortes et al., 2015). This LMCO is also sometimes referred to as a 'cuproxidase', since it has the ability to bind and oxidize copper ions.

Additionally, the ability to degrade lignin through LMCOs has also been observed in soil bacteria such as *Ochrobactrum* sp. (Granja-Travez et al., 2018). Other functions such as the oxidation of pigment and toxins are also prevalent in bacteria (Le Viet et al., 2025).

Animals

Animal LMCOs have been relatively understudied compared to plants, fungi and bacteria. Primarily insect LMCOs have been described, where they are thought to play a role in cuticle sclerotization (production of the exoskeleton) and pigmentation (Le Viet et al., 2025).

Recently, the first structural characterization of a termite LMCO has been reported, and its function is rather peculiar (Škerlová et al., 2024). Aged termite workers will readily sacrifice themselves for the community when confronted with danger. Near the secretory glands, these termites have evolved a sac-like structure containing non-toxic quinones and LMCOs. When the time is ripe, their body ruptures and the LMCOs oxidize the quinones into toxic compounds which are then excreted as a sticky compound to deter predators.

Ceruloplasmins/ferroxidases are also abundant in humans and other mammals. In animals they mostly contain six cupredoxin domains, so they are technically not considered as an LMCO according to our definition. In fact, 95% of copper in healthy human individuals is bound to ceruloplasmin (Lontie, 1984b). Here it is also believed to regulate iron homeostasis, but again, iron is by no means the only substrate (Lontie, 1984b).

The above examples should convince the reader that these enzymes have adopted many different roles throughout evolution. Ranging from defensive to offensive functions, and from binding organic phenols to other metal ions. However, despite the wide variety of function and substrate preference, the structural folds and copper binding motifs are highly conserved, suggesting that copper binding has evolved early on, and all these enzymes likely share a common ancestor.

1.4.3 Structural properties

By EC number, the Protein Data Bank (PDB) currently hosts 451 crystal structures for ferroxidases, 200 for laccases and four for ascorbate oxidases. When filtering on SCOP-e classification (a database that aims to categorize proteins according to structural and evolutionary relationships)(Fox et al., 2014), 218 crystal structures are classified as 'multidomain cupredoxins'.

The cupredoxin fold

LMCOs consist of three cupredoxin domains. These conserved folds were first described in the mono-domain copper enzymes plastocyanin and azurin, both of which bind only a single copper ion according to the type 1 (T1) architecture described below (Hakulinen & Rouvinen, 2015). The folds consist of beta sheets and turns, and are arranged into a typical greek-key topology, where amino acids distant along the primary structure can get in close proximity in the tertiary structure. Figure 1.11 depicts the general architecture of these folds, and a scheme that summarizes the LMCO mechanism of action.

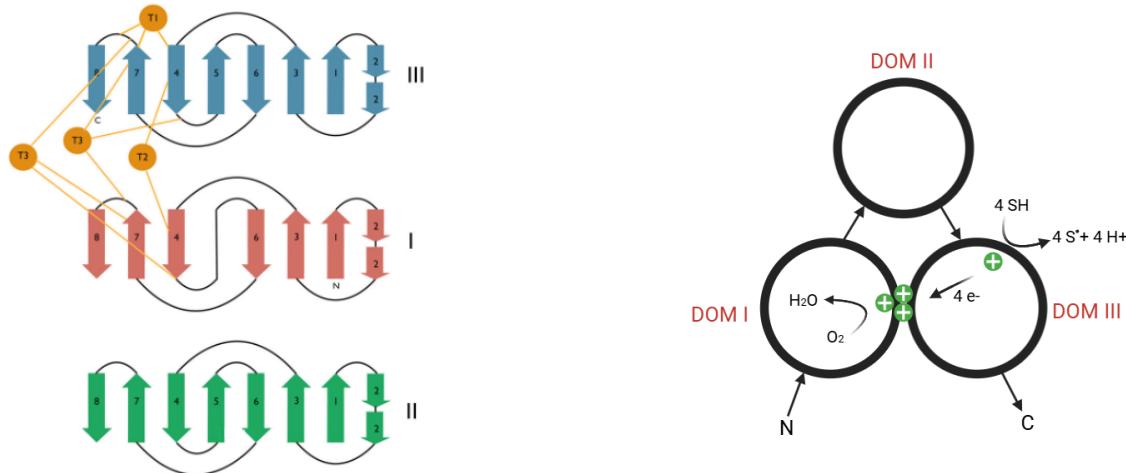


Figure 1.11: Left: Architecture of the three cupredoxin domains in an LMCO from the fungus *Melanocorpus albomyces*, and the position of the four copper ions. Figure adapted from Hakulinen & Rouvinen (2015). **Right:** General scheme of LMCO structure and electron transfer pathway. N and C represent the N- and C-terminus. Green plus-signs represent the four copper ions. This figure was created with BioRender.com.

Copper site architectures

Three types of copper ions exist in LMCOs, distinguished based upon their spectroscopic properties: type 1, 2 and 3 (T1, T2 and T3) (Hakulinen & Rouvinen, 2015; Lontie, 1984a). The T1 copper is positioned in the third domain, where it extracts one electron from the bound substrate. In all known cases, the T1 copper is coordinated by the sulfur of one Cys residue and the nitrogen's of two His residues. A fourth axial ligand is often involved in T1 copper coordination. In bacteria, this is a Met in most cases. In fungi, the Met is often replaced by Phe or Leu, at a larger distance from the T1 copper. The T1 copper site is depicted in Figure 1.12 for a bacterium and a fungus.

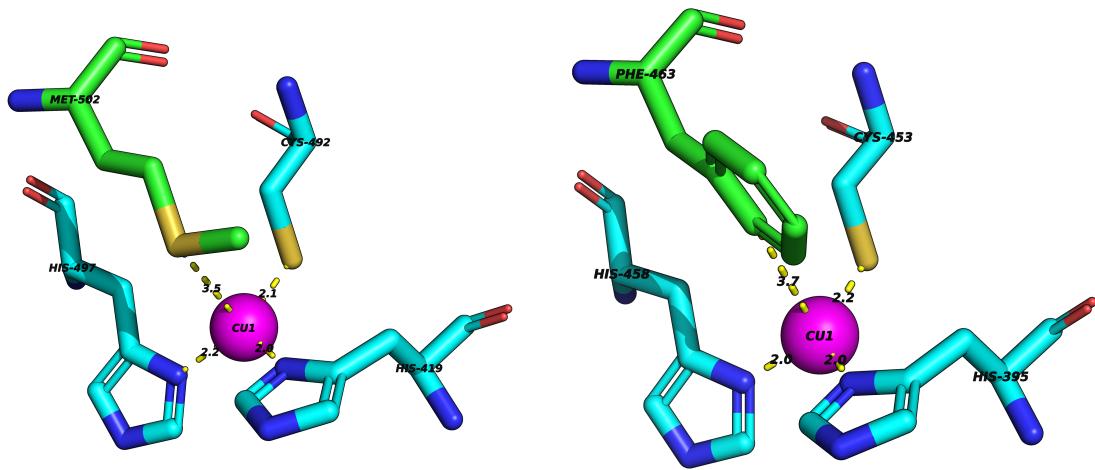


Figure 1.12: **Left:** T1 copper site of *B. subtilis* LMCO (PDB ID: 1GSK) **Right:** T1 copper site of *T. versicolor* LMCO (PDB ID: 1GYC). Notice the difference in axial ligand; Met502 for *B. subtilis*, and Phe463 for *T. versicolor*. The T1 copper ion is depicted in magenta, and distances are to be read in Ångstroms. This figure was generated in PyMol.

Fungal LMCOs exhibit higher reduction potentials than those of bacteria and plants, and this property is tightly linked to the nature of the axial ligand (Sharma, 2025). Generally, LMCO reduction potentials vary between 400-800 mV relative to the normal hydrogen electrode (NHE), and LMCOs are classified as low-, middle- and high-redox potential (Sharma, 2025). A Met as the fourth ligand causes the T1 copper to be richer in electrons, thereby reducing its electrophilic properties. This effect has also been shown experimentally in both ways. The M502L and M502F mutations in *B. subtilis* increased the reduction potential from 455 mV to 548 mV and 515 mV respectively (Durão et al., 2006), and the L499M mutation in *Botrytis aclada* reduced the reduction potential from 720 mV to 580 mV (Osipov et al., 2014). Most likely, fungi have evolved these mutations in order to degrade the recalcitrant lignin polymer.

Additionally, the higher reduction potential in fungi allows them to oxidize a larger scope of substrates. Reiss et al. (2013) have screened LMCOs from several sources against 91 different compounds including aromatic carboxylic acids, aromatic alcohols, benzonitriles, naphthalenes etc. It was found that the high-reduction potential LMCO from *T. versicolor* converted 77% of the 91 substrates, whereas low-reduction potential bacterial and plant LMCOs exhibited a significantly narrower substrate range. Potentially, this knowledge can also be applied to facilitate the oxidation of PE. It should be noted that the nature of the axial ligand is not the only contributor to reduction potential and substrate scope. Other factors such as proper orientation of the substrate, electrostatic environment and hydrophobicity of the T1 site, and size of the binding pocket are also important (Sharma, 2025).

The three other copper ions form what is called a trinuclear T2/T3 cluster (TNC) (Hakulinen & Rouvinen, 2015). These sit in between the first and third domain, approximately 13 Ångstroms away from the T1 site. In all known cases, the copper ions in the TNC are coordinated by eight His residues. When the enzyme is in its fully reduced state, O_2 binds at the TNC, where it is subsequently reduced to H_2O (Sharma, 2025). The T1 site and the TNC are connected through a highly conserved His-Cys-His motif, as depicted in Figure 1.13. It is through this structural motif that the subtracted electron travels intramolecularly from T1 to TNC (Sharma, 2025).

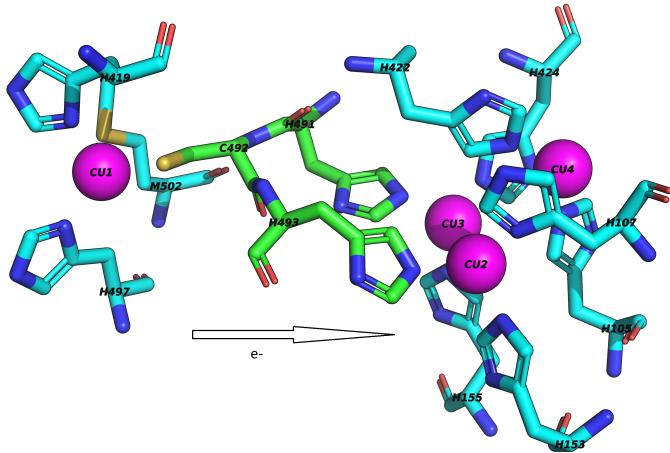


Figure 1.13: T1 and TNC in *B. subtilis* (PDB ID: 1GSK). Notice H491, C492 and H493 forming the bridge between the T1 copper and the TNC depicted in green. CU1 is the T1 copper, CU2-CU3 are the paired T3 coppers and CU4 is the T2 copper. This figure was generated in PyMol.

MR regions in LMCOs

As described in section 1.3.1, the findings by Zampolli et al. (2023) suggest that a hydrophobic MR loop in the second domain allows the enzyme to bind PE, and contains some acidic residues necessary for proton transfer. Additionally, a cold-adapted LMCO described by A. Zhang et al. (2022) also contains a catalytic site surrounded by numerous Met residues. MR regions have also been found in other bacterial LMCOs such as *Escherichia coli* and *Thermus thermophilus* (Miranda-Blancas et al., 2021; Roberts et al., 2002). Both of them are depicted in Figure 1.14.

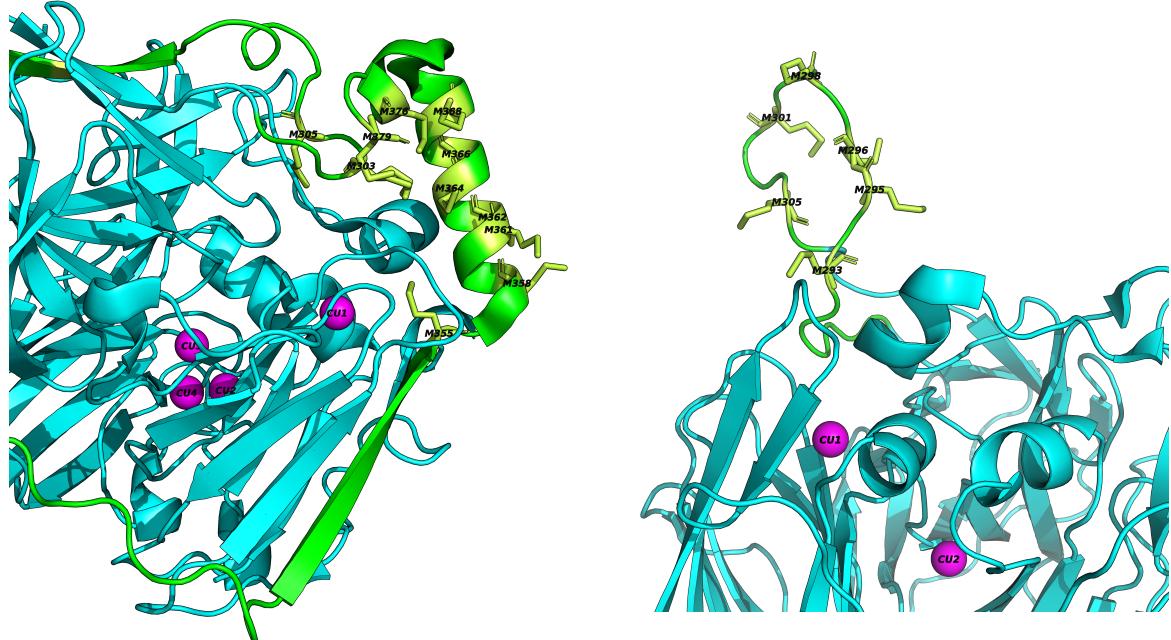


Figure 1.14: **Left:** MR helix covering the T1 site in *E. coli* (PDB ID: 1KV7). **Right:** MR beta-hairpin covering the T1 site in *T. thermophilus* in the open conformation (PDB ID: 6Q29). This crystal structure is missing two copper ions, since these were not present in the crystals.

E. coli's MR region is located in the third domain and begins at position M303. The next 50 amino acids loop around the entire protein, and then return to form an alpha helix covering the T1 site. This helix contains 9 Met residues over a span of approximately 50 residues. Including a disordered region between positions 380-400 that could not be resolved in the crystals, there is a defined region in space where fourteen Met residues cluster together.

The function of this MR region is thought to be in copper accumulation, as it is able to bind an additional three copper ions (Cortes et al., 2015). The enzyme can bind excessive amounts of copper in the cell and render it less toxic by oxidizing it. Notably, the presence of this MR helix decreases the laccase-like activity towards phenols (Kataoka et al., 2007). The active T1 copper site is sterically inaccessible for bulkier substrates due to the MR helix completely covering it. The position of the MR region thus seems to be of critical importance. If it is positioned such that it covers the T1 site, oxidation of bulky substrates cannot take place, since the distance between the two will be too large for electron-proton transfer to occur.

Most relevant is that this MR helix has been shown to rapidly immobilize the enzyme on carbon nanotubes in the context of bioelectrocatalysis (Cui et al., 2021). Several key residues that allow for the binding to occur have been identified. The main driving forces of this interaction are $\pi - \pi$ stacking between the side chains of aromatic amino acids, and hydrophobic interactions between the Met residues and the surface. The authors propose a four-stage mechanism as depicted in Figure 1.15. First, a polar residue "breaks the water layer" (Cui et al., 2021). Then, aromatic residues such as Tyr and Phe lock to the surface due to $\pi - \pi$ interactions. Afterwards, the MR region transitions from helix to loop, and the hydrophobic interactions between several Met residues and the carbon nanotube surface stabilize the complex (Cui et al., 2021).

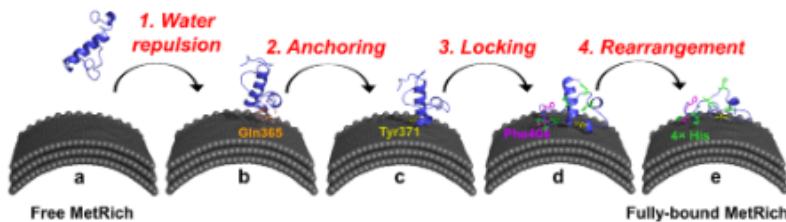


Figure 1.15: Schematic illustration of how the MR helix in *E. coli* binds to carbon nanotubes. Figure taken from Cui et al. (2021).

Furthermore, the researchers mutated the LMCO from *B. subtilis* (*cotA*) by attaching the MR helix from *E. coli* at its C- and N-terminus to confirm that oriented immobilization can occur (Cui et al., 2021). This was indeed the case. It was found that positioning the MR helix at the C-terminus draws the T1 copper site close the substrate, allowing for electron transfer to occur. On the contrary, when positioning the MR helix at the N-terminus, the enzyme is oriented in such a way that the T1 copper site is pointing away from the carbon nanotube. Hence, no current could be measured in their electrode setup.

The MR region in the LMCO from *T. thermophilus* is shorter and less rich in Met. Six Met residues are present between positions 290-310. It is thought to act like a pH-dependent lid that opens and closes, allowing substrates to bind within a specific pH range (Miranda-Blancas et al., 2021). Similarly, this MR region is also capable of orienting the enzyme on carbon nanotubes (Xu et al., 2022). However, in this study, the nanotube surface was positively charged with NH_2 functional groups.

These findings further support the idea that MR regions in LMCOs can facilitate binding to bulky hydrophobic substrates. A similar mechanism might thus be applicable for PE. If the MR region can be positioned such that the T1 copper is as close as possible to the substrate, and the oxidative power is high enough, electron-proton transfer might occur in a favorable manner. However, it remains unclear how long the MR region should be, how many Met residues or hydrophobic residues it should contain, where it has to be positioned etc. Additionally, according to Orlando et

al. (2025), acidic residues in the MR region should also remain close to the T1 copper site and the substrate to mediate proton transfer.

Thus, in light of all the above findings, good candidate PE-degrading enzymes are defined as follows.

- The enzyme is an LMCO. That is, a multicopper oxidase with three cupredoxin domains and four copper ions. One T1 copper ion in the third domain that subtracts the electron from the substrate, and a T2/T3 trinuclear cluster in the first domain where the reduction of dioxygen to water takes place.
- Somewhere along the sequence, a surface-accessible, hydrophobic MR region is present. If this MR region is positioned such that the T1 copper is drawn as close as possible to the substrate, an energetically favorable oxidation might occur. Additionally, the MR region contains acidic residues such as Asp or Glu possibly involved in proton transfer.
- The fourth axial ligand coordinating the T1 copper is not a Met, but preferably a Leu or Phe. This is thought to increase the reduction potential and allows for a broader substrate scope. Most likely, such a candidate will be of fungal origin.

Chapter 2

Materials and Methods

The following figure summarizes the entire workflow of this thesis, and a glance at some of the results.

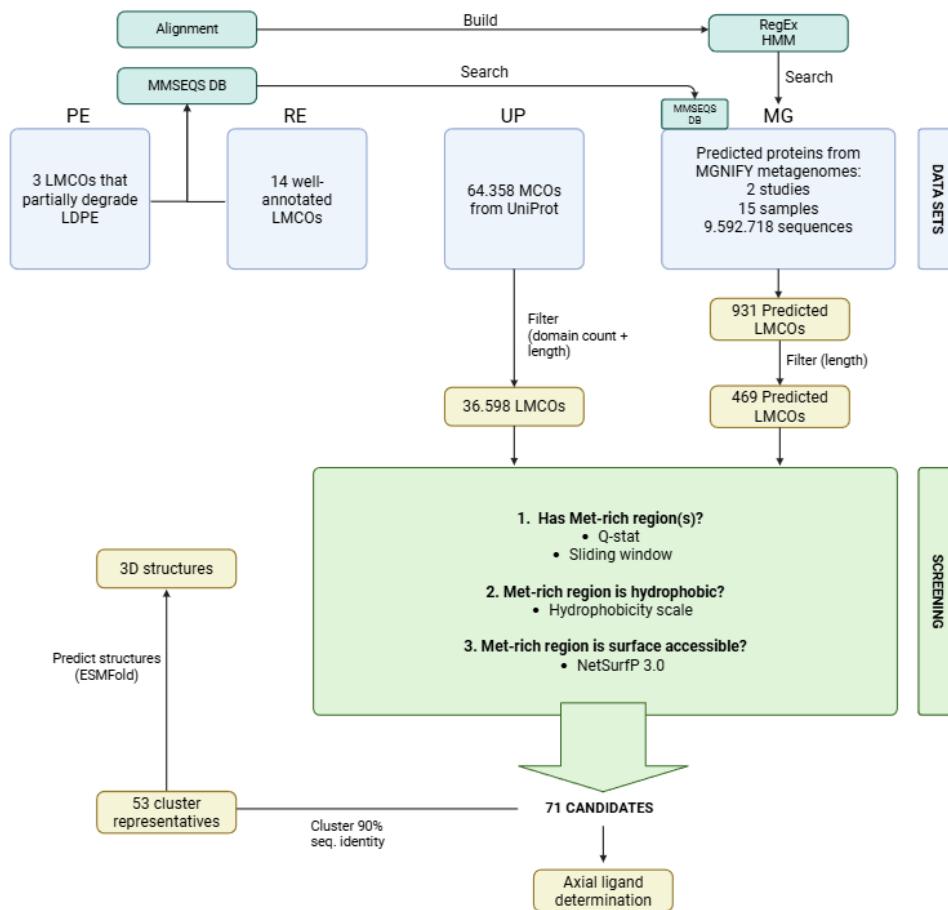


Figure 2.1: Schematic illustration of the entire workflow. Blue rectangles are the raw data sets. This figure was created with BioRender.com.

In summary, LMCOs from the UniProt database and predicted LMCOs from plastic debris metagenomic studies were subjected to a screening process that checks if the sequences possess the desired properties as described above. This resulted in a set of candidates for which three structures were predicted using ESMFold.

All code, written in Python (v3.13.1) and Bash (v5.2.15(1)-release), was kept in a private GitHub repository. Code snippets and all commands that were executed can be found in appendix A. All computational work was performed in a Windows Subsystem for Linux (WSL) environment (v2.3.26.0 and kernel v5.15.167.4-1) with the Debian (v12 bookworm) Unix flavor.

2.1 Literature study

Primary resources included the plasticDB, PAZy, Google Scholar, PubMed, and the Arenberg library in Heverlee, Belgium.

plasticDB and PAZy were used as the initial source for publications related to the enzymatic degradation of PE.

Search strings for Google Scholar and PubMed included ((“polyethylene” OR “polyethene” ”PE” OR ”LDPE” OR ”HDPE”) AND (“degrad*” OR ”biodegrad*” OR ”enzymatic degrad*” OR ”enzyme”)). Once a sufficient amount of publications was found, a tool named *ResearchRabbit* was used to identify shared citations and other relevant papers. The tool provides network graphs to explore similar publications, other authors, as well as earlier and later work.

From the Arenberg library, two books were used as a primary source of information: (i) *Microplastic pollutants* by Crawford & Quinn (2017) for everything related to plastics, and (ii) *Copper enzymes* (volume one and three) by René Lontie (1984) for everything related to LMCOs and other copper enzymes.

2.2 Data sets

2.2.1 Experimentally verified PE-degrading enzymes (PE)

As a starting point, all LMCOs known to be involved in PE degradation, and for which the amino acid sequence has been made public, were fetched. These can also be found in Table 1.1 in section 1.3.1, but they are repeated here for the sake of clarity. Amino acid sequences were manually copied from the NCBI database and saved to a FASTA file.

- LMCO from *Psychrobacter* sp. (abbr.: *bac_zh22*).
GenBank ID: OM891104 (A. Zhang et al., 2022)
- LMCO from *Lysinibacillus fusiformis* (abbr.: *bac_zh23*).
RefSeq ID: WP_193831439.1 (Y. Zhang et al., 2023)
- LMCO from *Rhodococcus opacus* R7 (abbr.: *bac_za23*).
RefSeq ID: WP_128643863.1 (Zampolli et al., 2023)

2.2.2 Representative LMCOs (RE)

Fourteen well-annotated LMCOs with known crystal structures and corresponding publications were gathered from the UniProt database. It was attempted to capture as much variety as possible by including LMCOs from all kingdoms of life, and multiple EC numbers. Four LMCOs come from bacteria, seven from fungi, as well as one plant, one animal, and one archaeal LMCO.

Most of these LMCOs are classified as laccases (EC 1.10.3.2), but some cuproxidases (EC 1.16.3.4) from bacteria and one ferroxidase (EC 1.16.3.1) from *Saccharomyces cerevisiae* were also included. Due to a lack of crystal structures in the PDB, ascorbate oxidases (EC 1.10.3.3) were not included. Amino acid sequences were manually copied from the UniProt database and saved to a FASTA file. Table 2.1 summarizes the members of this data set.

Table 2.1: Well-annotated LMCOs with known crystal structures and corresponding publications

	UniProt ID	PDB ID	EC	Publication	Abbr.
Bacteria					
<i>Bacillus subtilis</i>	P07788	1GSK	1.10.3.2/1.3.3.5	Engnita et al., 2003	bac-Bsu
<i>Escherichia coli</i>	P36649	1KV7	1.16.3.4	Roberts et al., 2002	bac-Eco
<i>Ochrobactrum anthropi</i>	A6XN0	6EVG	1.16.3.4	Graja-Travez et al., 2018	bac-Oan
<i>Thermus thermophilus</i>	Q72HW2	6Q29	1.10.3.2	Miranda-Blancas et al., 2021	bac-Tth
Fungi					
<i>Trametes versicolor</i>	Q12718	1GYC	1.10.3.2	Piontek et al., 2002	fun-Tve
<i>Trametes maxima</i>	D0VWU3	2H5U	1.10.3.2	L'yashenko et al., 2006	fun-Tma
<i>Coprinus cinereus</i>	Q9Y780	1A65	1.10.3.2	Ducros et al., 2001	fun-Cci
<i>Saccharomyces cervisiae</i>	P38993	1ZPU	1.16.3.1	Taylor et al., 2005	fun-Sce
<i>Melanocorpus albomyces</i>	Q70KY3	1GWO	1.10.3.2	Hakulinen et al., 2002	fun-Mal
<i>Thermothelomyces thermophilus</i>	G2QFD0	7ZN6	1.7.7.7	Kosmas et al., 2023	fun-Tth
<i>Steccherinum murashkinskyl</i>	I1VE66	5E9N	1.10.3.2	Glazunova et al., 2018	fun-Smu
Plant					
<i>Zea mays</i>	Q2PAJ1	6KLG	1.10.3.2	Xie et al., 2020	pla-Zma
Animal					
<i>Neocapritermes taracua</i>	A0A0S3AND7	8R71	1.10.3.2	Škerlová et al., 2024	ani-Nta
Archaea					
<i>Pyrobaculum aerophilum</i>	Q8ZWA8	3AW5	1.10.3.2	Sakuraba et al., 2011	arc-Pae

2.2.3 MCOs in UniProt (UP)

All UniProt entries belonging to the MCO family were fetched. Due to reasons discussed in section 1.4.1, the entire MCO family was queried instead of only entries classified as laccases (EC 1.10.3.2). This allowed us to capture possibly important LMCOs for which the EC classification remains unknown.

The data was gathered as follows. On the UniProt web interface, a query for '(family:"multicopper oxidase family")' was submitted, resulting in 64,358 hits. Via the 'Share' option in the interface, a URL for the API endpoint was generated, which was passed as input to a *curl* command and saved to a file (see appendix A.1). For the data format, a tab-separated value-file was chosen. A python script was then used to load the data into a *Polars* data frame, and perform the domain and length filtering as described below, along with some descriptive statistics and plots.

Since MCOs with three cupredoxin domains ('laccase-like' MCOs) are the main focus, this data set was filtered based on the 'Domain [FT]' column. In this column, UniProt labels cupredoxin domains as 'plastocyanin-like'. A regular expression (RegEx) pattern was used to extract entries with three 'plastocyanin-like' domains. Secondly, outliers were removed by filtering sequences shorter or longer than three standard deviations from the mean length. This resulted in a data set of 36,598 3dMCOs that could be subjected to the screening process.

2.2.4 Metagenomic studies related to plastic (MG)

The MGNIFY database was searched for the word 'plastic', resulting in fourteen metagenomic studies. Studies that contained full metagenome assemblies with predicted coding regions, and for which it was known the samples contained PE, were used for further analysis. Two studies with MGNIFY IDs MGYS00005704 and MGYS00005970 were selected. Both studies have been discussed in section 1.3.2. In total, they contain 9,592,718 amino acid sequences. The first study has twelve samples (six locations in the North Atlantic Ocean, particles < 5 mm and > 5 mm), and the second has three samples (two PE strips in marine waters and one control). Each sample comes with its own FASTA file. Data was gathered through a python script that communicates with the MGNIFY web API (see appendix A.2). The code for this script was adopted from the Jupyter notebooks made available by MGNIFY, with a few adaptations for this specific case.

The FASTA headers are formatted by ERZ identifiers:

```
> ERZ1740058.1 - NODE - 1 - length - 241699 - cov - 12.727500.1#290#565#1#...
```

The ERZ number refers to the assembly from which the protein sequence was derived, and the first two numbers after the hashtags are start and end coordinates in that specific node of the assembly.

However, proteins from the MGNIFY protein database and the ESMAtlas are accessed by MGYP numbers, and there was no straightforward way to convert the ERZ to an MGYP accession. This prevented us from fetching predicted structures, which use MGYP accessions. After contacting the MGNIFY team, they proposed the following workaround. Two files can be found in the flat file release of the database; 'mgy_seq_metadata*.tsv.gz' and 'mgy_contig_map*.tsv.gz' that can be used to perform the mapping. The first file maps the ERZ accession to an MGYC (contig) accession, and the second file then allows to go from MGYC to MGYP. This was done by using a *grep* command for the relevant ERZ identifiers.

2.3 Alignment of the PE-RE data set

MUSCLE (v5.3) (Edgar, 2021) was used to generate a multiple sequence alignment (MSA) of the PE-active LMCOs and the representative set (3 + 14 sequences respectively) to inspect whether the sequences were homologous. First, an ensemble alignment was produced and the dispersion was calculated. An ensemble contains multiple replicates of the same alignment with slightly altered parameters. The difference between the replicates gives an idea about the alignment's robustness. A high dispersion means that there is considerable variety among the replicates (i.e., high sequence diversity), indicating that we cannot be as confident in the correctness of the alignment. Afterwards,

the alignment with maximum column confidence was extracted from the ensemble and visualized in SeaView (v5.1) (Gouy et al., 2010).

The *pairwise2* module from Biopython (v1.78) was used to generate pairwise alignments for each pair of sequences. A BLOSUM62 scoring matrix with an open gap penalty of -1 and an extended gap penalty of -0,1 was used (see appendix A.3).

2.4 Searching LMCOs in the MG data set

The MG data set contains 9,592,718 sequences, most of which are probably not LMCOs and thus have to be filtered out. Three methods were employed to search for putative LMCOs; (i) RegEx patterns based on the MSA produced as described above, (ii) a hidden Markov Model (HMM) profile search using HMMER (v3.4), and (iii) a k-mer- and alignment-based search using the MMSEQS2 (v17.b804f) *search* module. A python script was used to match the RegEx patterns and parse the HMMER and MMSEQS2 output. All three methods are sequence-based.

2.4.1 Regular expression patterns

From the alignment produced as described in section 2.3, four RegEx patterns common to all members of the alignment were derived. Square brackets indicate that the pattern matches any of the symbols within. A dot means 'any character' and '{3}' means that the preceding symbol is repeated exactly three times.

- H[WLF]HG
- HP[MIF]H[LIA]H[GLV]
- HCH
- HCH.{3}H

These patterns are all related to the His and Cys residues involved in copper binding. The idea is thus that sequences matching these patterns are likely to possess the copper binding sites characteristic of LMCOs. Sequences that matched all four patterns were considered as hits. The biggest downside of this method is that it cannot capture any variation that might exist within these patterns. For example, an LMCO cannot match the first pattern if it contains HVHG, whereas we might still want to consider this slight alteration.

The filtered UP data set was also subjected to these RegEx patterns as a test. Since all sequences in this data set are technically LMCOs, it was expected that the majority matched all patterns.

2.4.2 Hidden Markov model

One way to circumvent the rigidity of regular expressions is to resort to probabilistic models. Today, HMM profiles can be built from a set of sequences with relative ease, and searches can be performed against large data sets at a similar rate and higher sensitivity to that of the BLAST algorithm.

From the alignment produced as described in section 2.3, an HMM model was built using HMMER (v3.4). The profile was then visualized using the Skylign server (Wheeler et al., 2014) to determine that the copper binding ligands were assigned high emission probabilities. A profile search was then performed against each sample of the MG data set with an E-value cutoff of 0.01 using a bash script (see appendix A.1).

It was also attempted to improve the HMM model by searching the UP data set and extracting the top 50 hits. These hits were then again aligned to the PE-RE data set, and an updated HMM model was built to capture more variety.

2.4.3 MMSEQS2

The MMSEQS2 *search* module requires a query and a target data set (Steinegger and Söding, 2017). It prefilters the target data set based on k-mer matches in the query data set, and then aligns the prefiltered hits to the queries. MMSEQS2 performs many-to-many searches, meaning that for each query in the PE-RE data set, a set of hits will be obtained. This is different from HMMER, where only one profile is used to perform the search.

Both the PE-RE and the MG data set were converted into MMSEQS2 database objects through the command line (see appendix A.1). A search was then performed using the PE-RE data set as a query and the MG data set as the target. Similar to the HMM search, only hits with an E-value below 0.01 were chosen for further analysis.

As a conclusive set of predicted LMCOs, the hits predicted by both the HMM model and MMSEQS2 (that is, the intersection of the two) were used for further analysis. These were then filtered on length based on the distribution observed in the filtered UP data set (243 - 957 amino acids long).

2.5 Candidate screening

Both the filtered UP data set and the predicted LMCOs from the MG data set were subjected to a screening process that first predicts the boundaries of regions enriched in Met for each sequence. Then, for each individual MR region, the hydrophobicity score and relative surface accessibility (RSA) was calculated.

2.5.1 Predicting regions enriched in Met

The first requirement is for the LMCO to contain an MR region. Two methods were used; q-statistics and sliding windows.

Q-statistic

This method was adopted from another paper where the authors were also tasked (for completely different reasons) to identify proteins where Met residues tend to cluster together in relation to some structural or functional property (Aledo, 2022). The idea is as follows. Sequences are first trimmed at the start and end to be cut up in equal segments of 50 amino acids. For each segment, the amount of Met residues is calculated. Using the following formula, a q-statistic is then computed for the entire amino acid sequence.

$$q = \frac{\sqrt{N \sum_{i=1}^N (x_i - \frac{1}{N} \sum_{i=1}^N x_i)^2}}{\sum_{i=1}^N x_i} \Rightarrow q \propto \text{sum(variances)}$$

Where N is the amount of segments and x_i is the amount of Met residues in the i -th segment. The equation boils down to calculating the mean amount of Met in each segment and then summing the variances. q is then proportional to this sum of variances.

If Met residues are distributed in a completely random fashion along the sequence, the variance in each segment will be low, and a lower q value is expected. If, however, the Met residues cluster together in one or more segments, the Met count in each segment will deviate more from the mean, causing the variance and subsequently q to increase (depicted in Figure 2.2 for *bac_za23*).

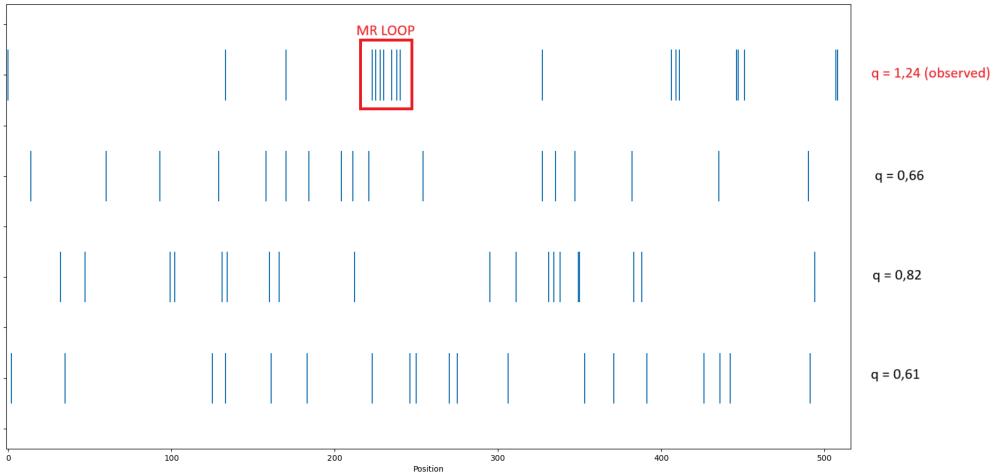


Figure 2.2: Observed and randomized locations of Met in *bac_za23* with corresponding q-stat. Top one is the observed sequence. Bottom three contain the same amount of Met residues, but randomly reshuffled.

One can thus, for each amino acid sequence, reshuffle the Met residues randomly and calculate q for each rearrangement. The result is a distribution of q -statistics as if Met residues were randomly scattered along the sequence (shown in Figure 2.3 for *bac_za23*). If the actual observed q -statistic significantly deviates from this null-hypothesis, it can be said that the Met residues aggregate somewhere along the sequence. This procedure was repeated 10,000 times for each protein, and only those with a p-value smaller than 0.01 were kept for further analysis. The code for this calculation can be found in appendix A.4.

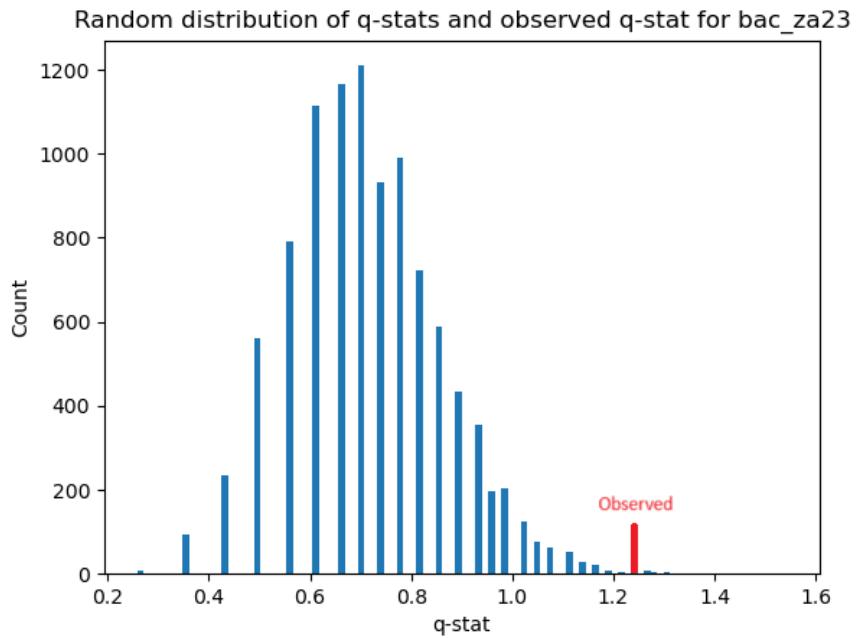


Figure 2.3: Distribution of q -stats after 10,000 random shuffles. The observed q -stat for *bac_za23* is significantly higher because the Met residues tend to cluster together. The height of the red bar has no meaning.

Sliding window

A downside of the q-stat method is that it does not return where the MR region is located. Hence, a sliding window approach was implemented in Python to identify regions in the sequence where the Met proportion (MP) exceeds a certain threshold (see appendix A.5). A window size n (e.g., 40 amino acids) can be defined, and within this window the MP is calculated. If this exceeds a certain threshold t (e.g., 0.15), the region is considered MR.

Some proteins will, however, naturally have a high MP (e.g., *bac_zh22* and *bac_Eco* have an MP of 0.052 and 0.056 respectively), and return regions that pass the sliding window test, without necessarily having an MR window that stands out from the rest of the sequence. Therefore, it was attempted to correct for this bias by adjusting the window size and threshold with respect to the MP of the entire sequence. Sequences that have a high MP will thus have to pass a higher threshold over a smaller window size. I will refer to both methods as static and dynamic sliding windows.

Both methods (q-stat and sliding windows) were tested on the PE-RE data set to optimize the parameters, as this data set is known to contain some LMCOs with experimentally verified MR regions (e.g., *bac_za23* and *bac_Eco*). At the end, sequences that passed both the q-stat and sliding window tests were considered for further analysis.

2.5.2 Scoring hydrophobicity

The ProtScale web tool by Expasy lists 23 different hydrophobicity scales that can be used to score amino acids based on their hydrophobic tendencies, each of them based on different experimental procedures (Gasteiger et al., 2005). One of these scales was chosen to perform this step (Black & Mould, 1991). Table 2.2 depicts the scores assigned to each amino acid.

Table 2.2: Hydrophobicity scores for each amino acid. Based on the scale derived by Black & Mould (1991).

Amino acid	Score	Amino acid	Score
Ala	0,616	Leu	0,943
Arg	0,000	Lys	0,283
Asn	0,236	Met	0,738
Asp	0,028	Phe	1,000
Cys	0,680	Pro	0,711
Gln	0,251	Ser	0,359
Glu	0,043	Thr	0,450
Gly	0,501	Trp	0,878
His	0,165	Tyr	0,880
Ile	0,943	Val	0,835

Most importantly, polar amino acids such as Glu and Arg are assigned low values, whereas non-polar amino acids such as Val or Ala are assigned high values. Each MR region was scored according to this scale, and the final score divided by the length of the MR region in order to normalize the results. The MR region in *bac_za23* was used as a guide to define the optimal hydrophobicity score threshold.

2.5.3 Scoring surface accessibility

Recent advancements in protein language models have allowed for the rapid elucidation of structural characteristics of proteins. NetSurfP 3.0 is a tool that can accurately predict surface accessibility,

secondary structure, structural disorder and backbone dihedral angles of each amino acid in a protein by relying on the embeddings of a pretrained language model ESM-1b (Høie et al., 2022).

All sequences with an MR region were submitted to the NetSurfP web server (<https://services.healthtech.dtu.dk/services/NetSurfP-3.0/>). Results were then downloaded in comma-separated format, and for each sequence the average surface accessibility of the predicted MR window was computed. The MR region in *bac_za23* was used as a guide to define the optimal RSA threshold in the larger data sets.

2.6 Further analysis of the candidates

2.6.1 Predicting the T1 Cu axial ligand

The axial ligand of the T1 Cu site can be identified through the pattern 'HCHxxxHxxxxA', where A is the axial ligand and x is any amino acid. This pattern was converted into a regular expression ('HCH.{3}H.{4}') to identify the axial ligand at the T1 copper site. The last symbol in this pattern contains the axial ligand. Sequences that did not match this pattern were assigned a *None* value. Sequences with a Leu or Phe at this position are, theoretically, expected to exhibit higher reduction potentials and broader substrate scopes.

2.6.2 Clustering and alignment

Candidates were first clustered on 90% sequence identity using the MMSEQS2 *cluster* module to explore sequence diversity (Steinegger and Söding, 2017). Cluster representatives were then aligned using MUSCLE (v5.3) (Edgar, 2021) and visualized in SeaView (v5.1) (Gouy et al., 2010) in order to confirm the shared presence of copper binding residues. This was done through the command line, and the executed commands can be found in appendix A.1.

2.6.3 Structure prediction

To visualize the candidates and their MR regions, three-dimensional structures of three candidates (two from the UP data set and one from the MG data set) were predicted using the ESMfold (v1, num_recycles=3) server provided by Google Colab (Zeming et al., 2023) and visualized in PyMol (v3.1.3.1) (Schrödinger and DeLano, 2020). Appendix B contains pIDDT plots for the predicted structures.

Chapter 3

Results

3.1 Data exploration

3.1.1 Raw UP data set

This data set contains 64,358 UniProt entries belonging to the multicopper oxidase family.

3dMCOs with no EC number dominate

Figure 3.1 shows the top 5 EC numbers in this data set, and the amount of 'plastocyanin-like' domains for each entry. It is expected that laccases (EC 1.10.3.2) are the most prominent members of the MCO family (Gräff et al., 2020). Subsequently, since most laccases are comprised of three cupredoxin domains, most of the domain counts should resort to 3. Additionally, we also expect some 2dMCOs (mostly nitrite reductases and some laccases), as well as some 6dMCOs (ceruloplasmins/ferroxidases) to be present.

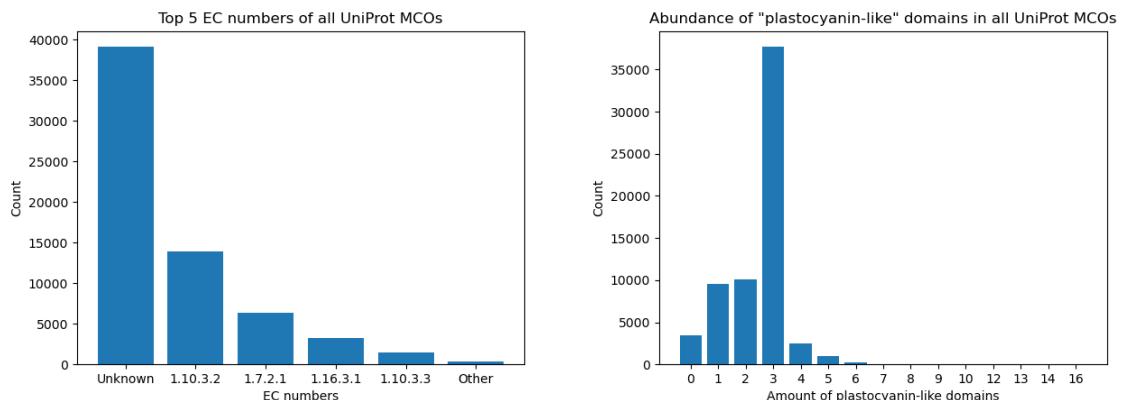


Figure 3.1: **Left:** Top five EC numbers in the UP data set. **Right:** Cupredoxin domain counts

Most entries do not have an EC number assigned to them. The second most abundant EC class is represented by the laccases (EC 1.10.3.2). The other three EC classes include nitrite reductases (EC 1.7.2.1), ferroxidases (EC 1.16.3.1) and ascorbate oxidases (EC 1.10.3.3).

As expected, 3dMCOs are the most abundant, since most laccases are indeed 3dMCOs. Notably, some MCOs other than 2dMCOs, 3dMCOs and 6dMCOs are present as well, which is in disagreement with Gräff et al. (2020). Some MCOs do not have plastocyanin-like domains at all.

Length and mass distribution in accordance with expectation

Figure 3.2 shows the distribution of length and mass in this data set. According to the literature, most LMCOs are around 500 amino acids long with a molecular mass ranging from 65,000-140,00 Da (Lontie, 1984b). Additionally, we would expect some of the 2dMCOs (nitrite reductases and some laccases) to be of shorter length and lower mass, as well as some 6dMCOs (mostly ferroxidases) to be considerably longer and heavier.

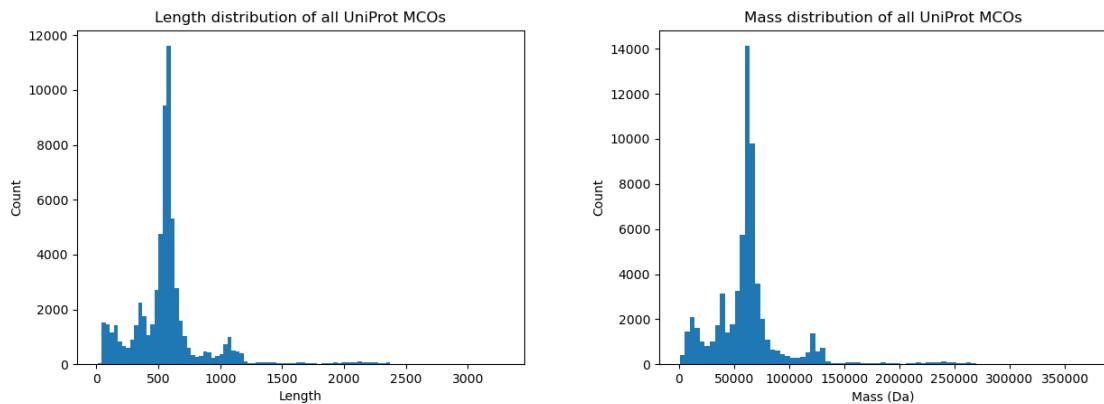


Figure 3.2: **Left:** length distribution. **Right:** mass distribution.

As expected, most sequences are 200-800 amino acids long, with a mean length of 576 amino acids and standard deviation of 324 amino acids. The minimum length is 8 amino acids, and the maximum length is 3219 amino acids. These outliers will have to be filtered out later. Mass distribution also seems to be in accordance with expectation.

Eukaryotes dominate

Figure 3.3 shows the taxonomic distribution of this data set. Eukaryotes are the dominant source of MCOs in the UniProt database. Within the eukaryotes, fungi and plants are most abundant. More specifically, 56,724 entries come from eukaryotes, 7659 from bacteria, and only 2 from archaea.

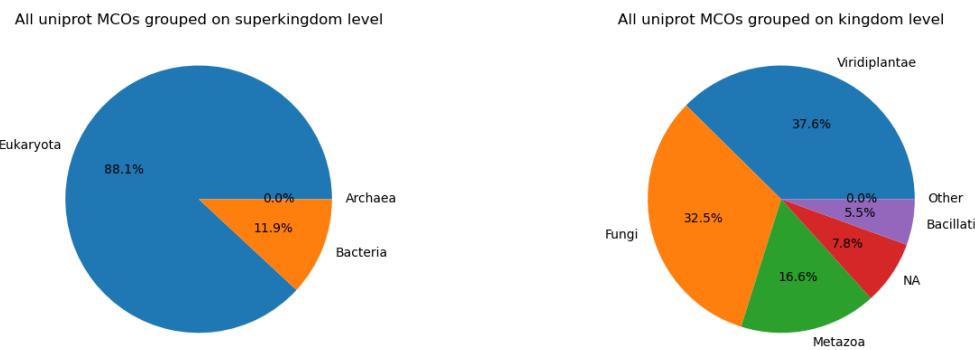


Figure 3.3: Taxonomy of the UniProt data set. **Left:** grouped by superkingdom. **Right:** grouped by kingdom. 'NA' means that the taxonomy information was not available in the entry header.

3.1.2 Filtered UP data set

This data set contains 36,598 LMCOs with three plastocyanin-like domains and an average length of 584 amino acids with standard deviation of 70 amino acids.

Reduction in nitrite reductases and ferroxidases

After filtering on domain counts, it was expected to see a significant reduction in 2dMCOs such as nitrite reductases (EC 1.7.2.1) and 6dMCOs such as ferroxidases (1.16.3.1). However, since most MCOs have an undetermined EC number, this will likely remain the dominant group. Figure 3.4 shows the top 5 EC numbers in the filtered UP data set.

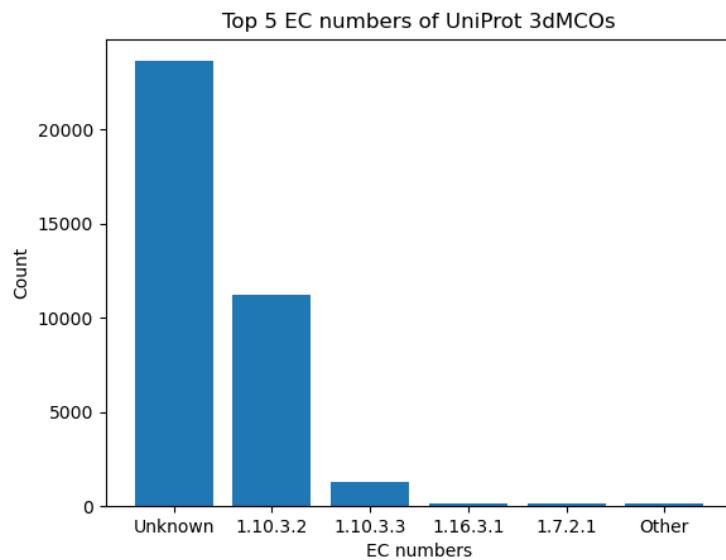


Figure 3.4: Top 5 EC numbers of the filtered UniProt data set.

As expected, a significant reduction in nitrite reductases (EC 1.7.2.1) and ferroxidases (EC 1.16.3.1) took place, and the 'Unknown' category remains the dominant one. This also seems to justify the choice of considering all MCOs. If only entries labeled as laccases (EC 1.10.3.2) were considered, more than 20,000 possible candidates might have been missed.

Bacteria are vastly underrepresented

After filtering, the large majority of bacterial MCOs disappeared, and the eukaryotes now represent almost 99% of the data set. It thus seems that the candidates we will identify downstream will most likely be of fungal or plant origin.

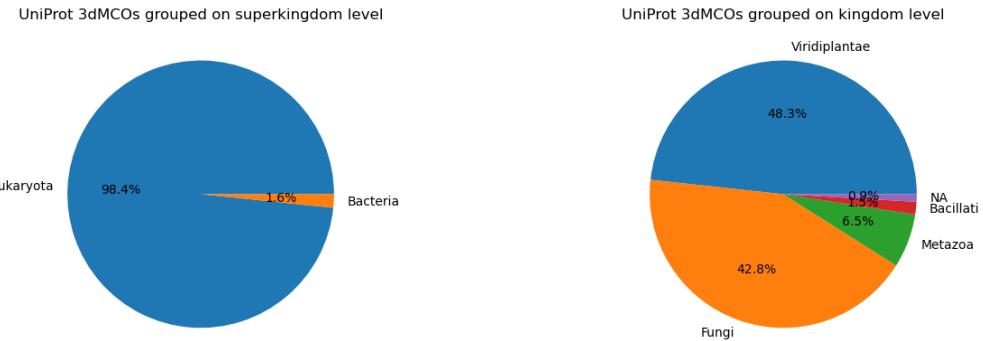


Figure 3.5: Taxonomy of the filtered UniProt data set. **Left:** grouped by superkingdom. **Right:** grouped by kingdom. 'NA' means that the taxonomy information was not present in the entry header.

Met proportion in accordance with expectation

Since we are looking for LMCOs with MR regions, we can briefly look at the distribution of Met proportions for each protein. According to the literature, Met abundance in proteins is about 2% (Aledo, 2022), and this is indeed what was found in this data set as well (Figure 3.6).

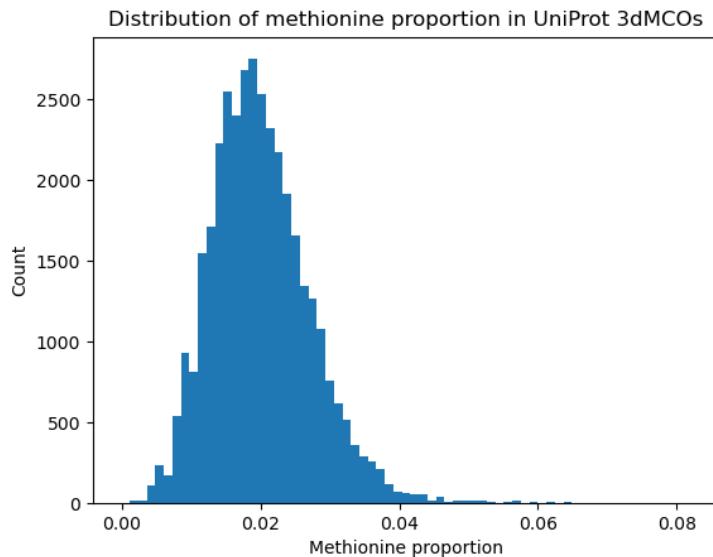


Figure 3.6: Methionine proportion of the filtered UniProt data set.

RegEx test and axial ligand determination

The RegEx patterns described in section 2.4.1 were tested on this data set. The 'HCH' pattern was matched by 28,270 entries (77%). Only 17,497 of the 36,598 (48%) entries matched all four of the regular expressions. Thus, more than half of the proteins annotated by UniProt as 'multicopper oxidase' with three plastocyanin-like domains could not be detected by the set of RegEx patterns derived from the PE-RE data set.

The RegEx pattern described in section 2.6.1 was also used to identify the variety of axial ligands in this data set. It was expected that bacteria, plants and animals mostly have Met, whereas the

fungi should be more enriched in Leu or Phe. Figure 3.7 shows the results in a grouped bar chart.

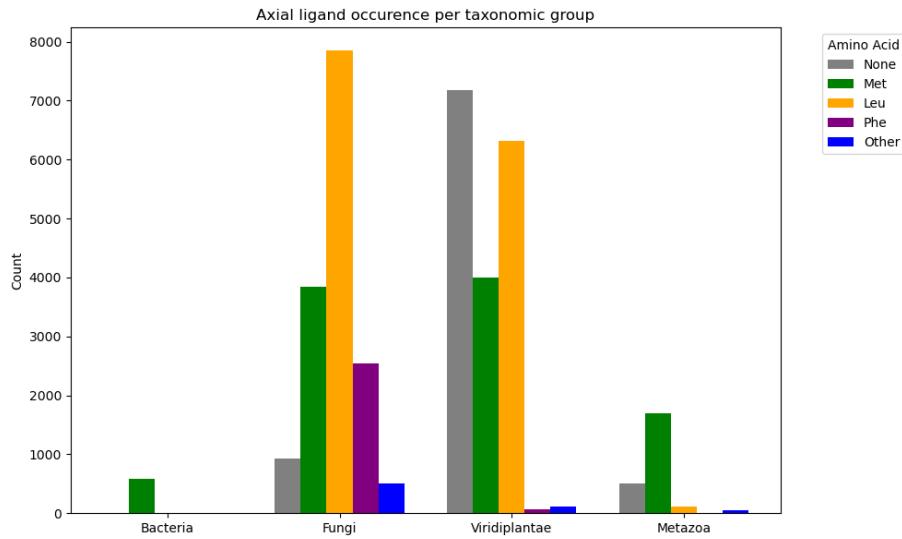


Figure 3.7: Axial ligands of the T1 copper site as identified by a regular expression grouped by taxonomy. 'None' means that the pattern could not be matched.

As expected, the limited amount of bacteria only have Met at the T1 Cu axial ligand position, and all of them were able to be matched with the pattern. For the fungi, it is clear that Leu prevails, and Phe is particularly more abundant than in the other kingdoms. Notably, the large majority of plants are not matched by the pattern, and Leu is more abundant than Met. In the animal kingdom, Met prevails.

3.2 Alignment of the PE-RE data set confirms homology

Figure 3.8 shows the relevant parts of the MSA between the 3 PE-degrading LMCOs from the PE data set, and 14 LMCOs from the RE data set. The alignment indeed confirms that all sequences possess the copper binding sites, and are thus likely to share a common ancestor. However, aside from these conserved blocks, the rest of the alignment is highly variable and indicates strong sequence divergence. Especially the regions in between the known cupredoxin domains are extremely variable and full of gaps. This is further confirmed by the high dispersion of 0.43 calculated from the ensemble, which indicates that the alignment is not robust enough to produce the same result in multiple replicates. We should thus keep the high sequence diversity of these proteins in mind when interpreting the alignment.

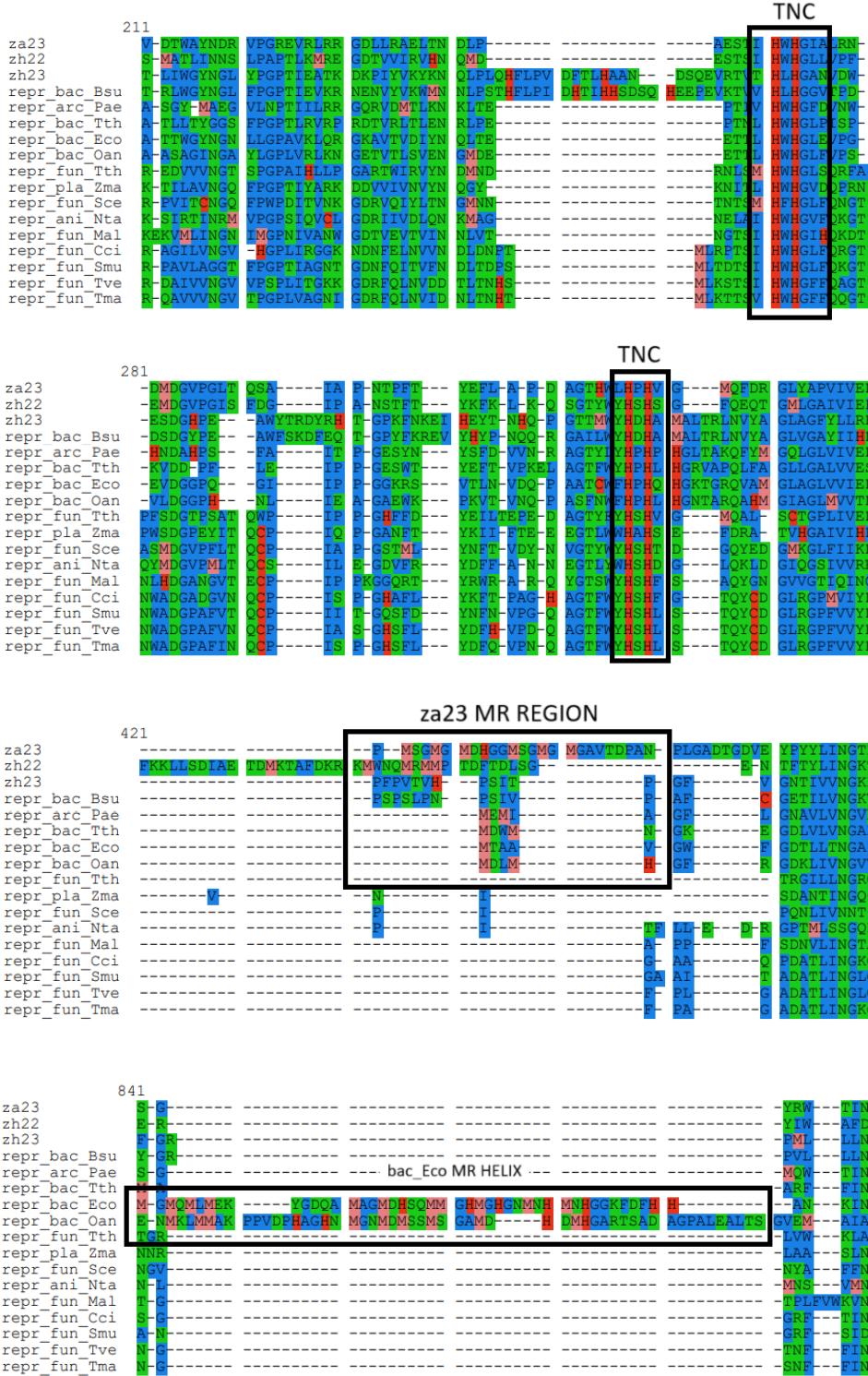


Figure 3.8: Alignment of the PE-RE data set. Only relevant chunks are shown. His and Cys residues are colored in red to highlight the copper binding sites. Met is colored in salmon to highlight known MR regions. Hydrophobic amino acids are colored in blue, and the rest is colored in green. TNC refers to the copper binding ligands at the trinuclear copper center. T1 refers to the copper binding ligands at the T1 copper site. The axial ligand is the fourth copper binding ligand at the T1 copper site.

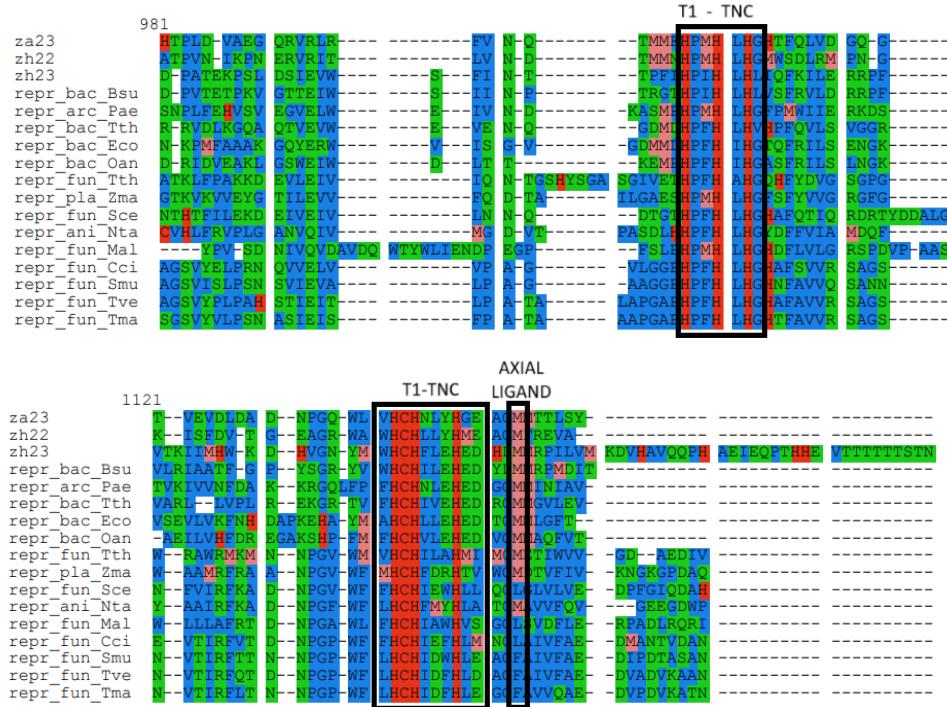


Figure 3.9: Alignment of the PE-RE data set (continued).

Two of the known MR regions of *bac_za23* and *bac_Eco* are also shown. It is suggested by the alignment that both these regions contain insertions not found in other LMCOs (as deduced from the relatively high amount of gaps in the other sequences for these regions). *Bac_zh22* also contains a region mildly rich in Met at the same location as the MR region of *bac_za23*, which was also noted by Zampolli et al. (2025) by looking at its structure. However, as mentioned before, this sequence is unusually rich in Met (total MP of 0.052), so this could just be due to chance. It can also be seen that *bac_Oan* possesses an MR insert at the same location as the one of *bac_Eco*, both of which are considerably longer and richer in Met.

The axial ligand of the T1 copper site can also be seen in the alignment. Fungi rarely have Met at the axial ligand position, except for *fun.Tth*. All the rest have either Phe or Leu. All the non-fungal LMCOs possess Met as the axial ligand. This finding is in agreement with the literature and the results obtained from the UP data set exploration in the previous section.

Table 3.1 shows the calculated sequence identities between each pair. Further confirming the diversity in this family of proteins, it can be seen that LMCOs from the same kingdom barely exceed 30% sequence identity. *bac_zh23* is most identical to *bac_Bsu* with a sequence identity of 42%, likely because both are bacilli. *Bac_za23* has 32% sequence identity with *bac_Tth*, and *bac_zh22* is most identical to *bac_za23* with a sequence identity of 31%. *T. versicolor* and *T. maxima* share the highest sequence identity of 74%, which is logical because they are in the same genus.

Table 3.1: Sequence identity matrix of the PE-RE data set. Bacteria are colored in blue, fungi in light green. Plant, animal and archaea are colorless.

	zh22	za23	zh23	Eco	Bsu	Tth	Oan	Sce	Tth	Tma	Tve	Smu	Cci	Mal	Zma	Nta	Pae
zh22	1																
za23	0,31	1															
zh23	0,24	0,23	1														
Eco	0,29	0,30	0,25	1													
Bsu	0,26	0,25	0,42	0,27	1												
Tth	0,28	0,32	0,24	0,34	0,27	1											
Oan	0,27	0,28	0,25	0,39	0,28	0,33	1										
Sce	0,25	0,24	0,25	0,25	0,23	0,23	0,24	1									
Tth	0,26	0,27	0,24	0,26	0,25	0,26	0,25	0,27	1								
Tma	0,25	0,26	0,24	0,25	0,26	0,27	0,24	0,28	0,29								
Tve	0,27	0,28	0,24	0,26	0,25	0,27	0,26	0,30	0,29	0,74	1						
Smu	0,26	0,26	0,25	0,25	0,25	0,27	0,24	0,31	0,29	0,60	0,62	1					
Cci	0,26	0,28	0,24	0,26	0,26	0,28	0,27	0,30	0,29	0,53	0,53	0,54	1				
Mal	0,25	0,27	0,23	0,26	0,25	0,26	0,25	0,28	0,26	0,31	0,30	0,31	0,31	1			
Zma	0,27	0,26	0,24	0,24	0,25	0,25	0,25	0,28	0,28	0,28	0,27	0,29	0,29	0,26	1		
Nta	0,26	0,28	0,24	0,25	0,24	0,24	0,25	0,27	0,27	0,30	0,29	0,29	0,28	0,28	1		
Pae	0,27	0,30	0,25	0,29	0,28	0,34	0,31	0,25	0,24	0,27	0,26	0,26	0,26	0,26	0,25	1	

3.3 HMM profile

Figure 3.10 shows the HMM logo for the profile derived from the MSA in the previous section. As can be seen, the copper binding ligands possess high information content and emission probabilities. These will thus be able to pick up enzymes with similar patterns in the MG data set.

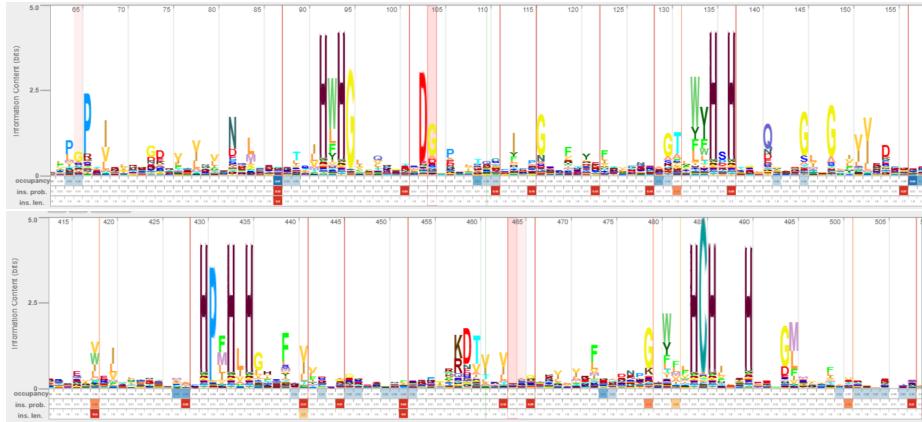


Figure 3.10: Logo of the HMM profile derived from the MSA in section 3.2. Only relevant chunks are shown. This logo was generated by the Skylign server (Wheeler et al., 2014).

As described in the methods section, it was also attempted to extend the HMM model by extracting the top 50 hits of the UP data set identified by this HMM profile, and then perform the alignment and HMM build again. However, it was observed that the top 50 hits were all highly similar to each other, and the profile became flooded with almost identical sequences, thereby actually reducing variability. Indeed, searching the MG data set with this updated HMM profile resulted in less hits. This also pointed out that there exists some redundancy in the UP data set, and a clustering would have been favorable before attempting this step. In the end, it was chosen to stick with the original HMM profile, since we also do not want the contribution of the three PE-degrading enzymes being overshadowed by other non-PE-degrading LMCOs.

3.4 469 predicted LMCos in the MG data set

In total, 238 sequences were identified by the RegEx patterns, 1138 by the HMM profile, and 945 unique hits by the MMSEQS2 *search* module. The HMM profile thus seems to be the most sensitive method. All hits identified by the RegEx patterns, except for one, were also picked up by the HMM profile and MMSEQS2. Figure 3.11 summarizes the results in a Venn diagram.

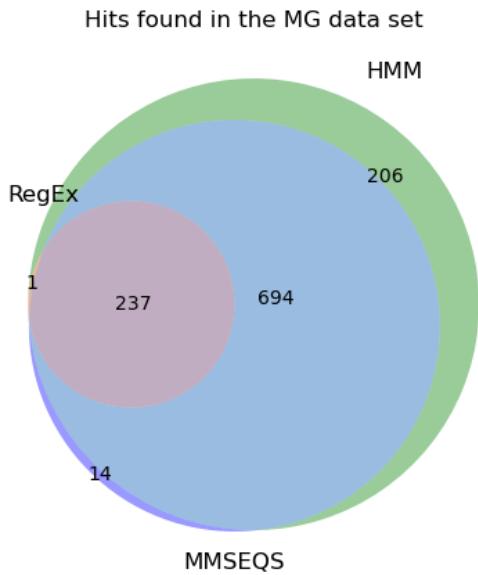


Figure 3.11: Venn diagram of predicted LMCOs found in the MG data set using three different methods. Green circle is HMM, Blue circle is MMSEQS2, and the inner pink circle are the sequences picked up by the RegEx patterns.

The 931 hits identified by both the HMM and MMSEQS2 were filtered based on length, only keeping those longer than 243 and shorter than 957 amino acids. After this filtering step, 469 sequences remained; 252 from MGYS00005704, and 217 from MGYS00005970. These can now be screened for the presence of hydrophobic MR binding regions.

3.5 Parameter optimization

3.5.1 Identifying MR regions

In the PE data set, *bac_za23* is known to possess an MR region as described in section 1.3.1. Additionally, *bac_zh22* also possesses a catalytic site surrounded by numerous Met residues. However, this sequence has an unusually large MP of 0.052 and lacks a well-defined region where the Met residues cluster together as observed in *bac_za23*. Hence, at least the MR region in *bac_za23* should be correctly predicted by the q-stat and sliding windows.

In the RE data set, *bac_Eco* is known to possess an MR helix involved in copper binding, and immobilization on carbon nanotubes (as described in section 1.4.3). *Bac_Tth* also has an MR region, but shorter and less rich in Met, which has not been shown to bind hydrophobic surfaces. Additionally, from the alignment produced in section 3.2, it was observed that *bac_Oan* has an MR region highly similar to the one of *bac_Eco*. Hence, the MR regions of *bac_Eco* and *bac_Oan* should be correctly predicted.

The q-stat method with a p-value threshold of 0.01 and 10,000 random shuffles consistently predicted *bac_za23*, *bac_Eco* and *bac_Oan* to have Met residues that tend to cluster together. *Bac_Tth* and *bac_zh22* did not pass the q-stat test.

Concerning the static sliding window; *bac_za23* has an MR region spanning from position 222-261. Along this window of 39 amino acids, 7 Met residues are present, which results in an MP of 0.18. *bac_Eco*'s MR region spans from position 355-400 and contains 14 Met residues, which is an

MP of 0.31. Lastly, *bac_Oan* has an MR window from position 368-408 with 10 Met residues and a proportion of 0.25. It thus seems that a window size of 40 and a minimum threshold of 0.15 is appropriate. Table 3.2 shows which set of parameters correctly predicted the three known sequences with an MR region.

Table 3.2: Different sets of parameters for the static sliding window. Numbers represent the amount of sequences with at least one region that passes the test. Green indicates that the three sequences with a known MR region were identified. n=window size, t=MP threshold.

$n \backslash t$	0.10	0.15	0.20
20	12	6	5
40	8	3	2
60	5	2	1

The window size and threshold combination of ($n=40; t=0.15$) correctly predicted the three known sequences with an MR region. As expected, lower thresholds and window sizes also did, but with some potential false positives. The other three results in red contained either *bac_Eco* and *bac_Oan*, or only *bac_Eco*. It was also noted that this combination of parameters (40;0.15) predicted the MR region for *bac_za23* to start at position 201 and end at position 262, which is 22 amino acids longer than what the authors report (222-261) (Orlando et al., 2025). For *bac_Eco* it is known that the MR window spans from positions 355-400, and the static sliding window ($n=40; t=0.15$) predicted positions 328-422, which is considerably longer at both ends.

Concerning the dynamic sliding window; *bac_za23* has a total MP of 0.037, which means that in order for its MR region to pass a dynamic sliding window test, a window size of roughly $1.5//MP = 40$ and a threshold of $4.5 \cdot MP = 0.17$ can be chosen. The following table shows how many sequences passed the dynamic sliding window test with different parameters around these values.

Table 3.3: Different sets of parameters for the dynamic sliding window. 'MP' is the methionine proportion of the entire sequence. Color coding is the same as the previous table.

$n \backslash t$	3 · MP	4.5 · MP	6 · MP
1//MP	13	7	5
1.5//MP	8	4	1
2//MP	8	2	1

The dynamic sliding window returns slightly different results. In general, it seems to be less strict, as more sequences pass the defined thresholds. It predicts one extra sequence (*fun_Tth*) in the ($n=1.5//MP; t=4.5 \cdot MP$) combination. Upon inspecting this sequence, it can indeed be seen that positions 565-587 contain 6 Met residues. Additionally, we have confirmed that *bac_Eco* passes the test not simply because it has a high MP (0.056), but because there is actually an MR window that stands out from the rest of the sequence.

The same start and end coordinates were predicted for the MR region of *bac_za23*. However, for *bac_Eco*, the MR region was predicted to range from position 342-400, which is more accurate than what the static sliding window returned. Hence, it was chosen to use the dynamic sliding window to predict the boundaries of the MR regions. The static sliding window will also be used to determine if the MR regions pass a fixed threshold, but not to determine the location of the region.

3.5.2 Scoring hydrophobicity

The next step is to score the identified MR regions based on hydrophobicity. According to the hydrophobicity scale that was chosen, the observed MR region in *bac_za23* has a score of 0.53. The predicted MR region of *bac_za23* (which is 22 amino acids longer) has a score of 0.50. We can thus pick 0.50 as a minimum threshold. However, depending on the distribution of hydrophobicity scores in the larger data sets, a higher threshold could be defined.

3.5.3 Scoring surface accessibility

The final step is to determine if the identified MR regions are accessible to the surface of the protein. Again, *bac_za23* was used as a reference and submitted to the NetSurfP web server. The result is depicted in Figure 3.12.

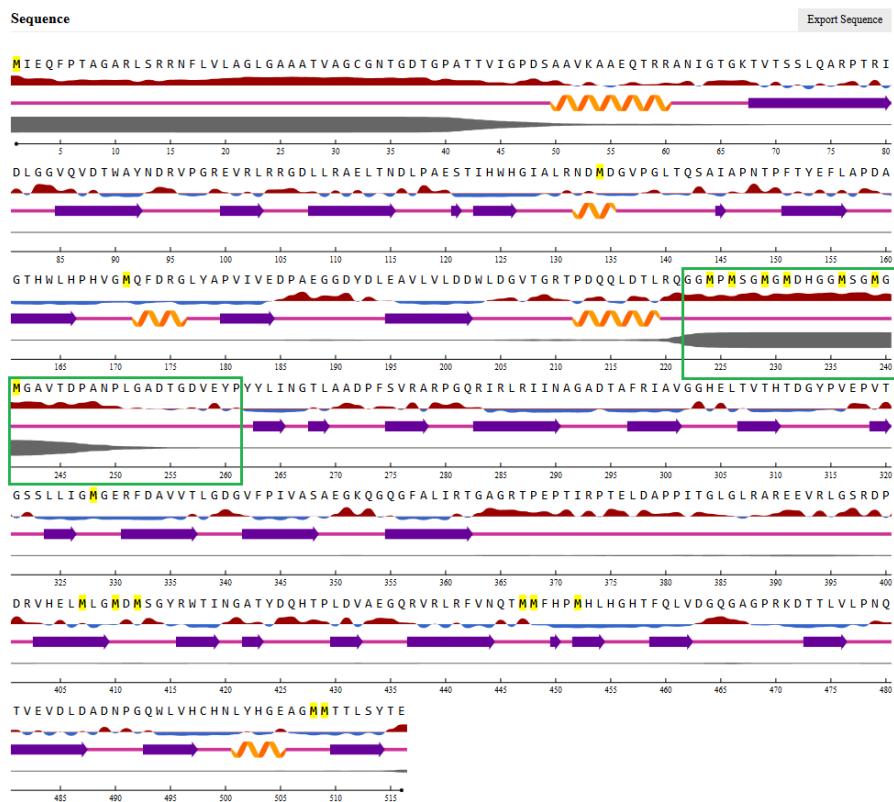


Figure 3.12: NetSurfP output for *bac_za23*. The MR window is shown in a green rectangle. The tool correctly predicts that this region is accessible to the surface (red line under the residues means surface exposed, blue means buried). Additionally, the region is characterized by a high probability of disorder (as seen by the expanding grey box at the bottom).

Calculating the mean RSA of the region in the green rectangle returns a result of 56.17%. Hence, a minimum RSA of 0.50 seems like a reasonable threshold.

To conclude, we will use a q-stat calculation with 10,000 random shuffles and p-value threshold of 0.01, followed by a static sliding window with window size 40 and MP threshold of 0.15, and a dynamic sliding window with window size of 1.5//MP and threshold of 4.5·MP. The dynamic sliding window will be used to predict the start and end coordinates of MR regions. Each MR region needs to pass a hydrophobicity score and RSA larger than 0.50.

3.6 71 Candidates found

Both the filtered UP data set and the predicted LMCOs in the MG data set were screened for the presence of hydrophobic MR binding regions. Figure 3.13 summarizes the results for both data sets.

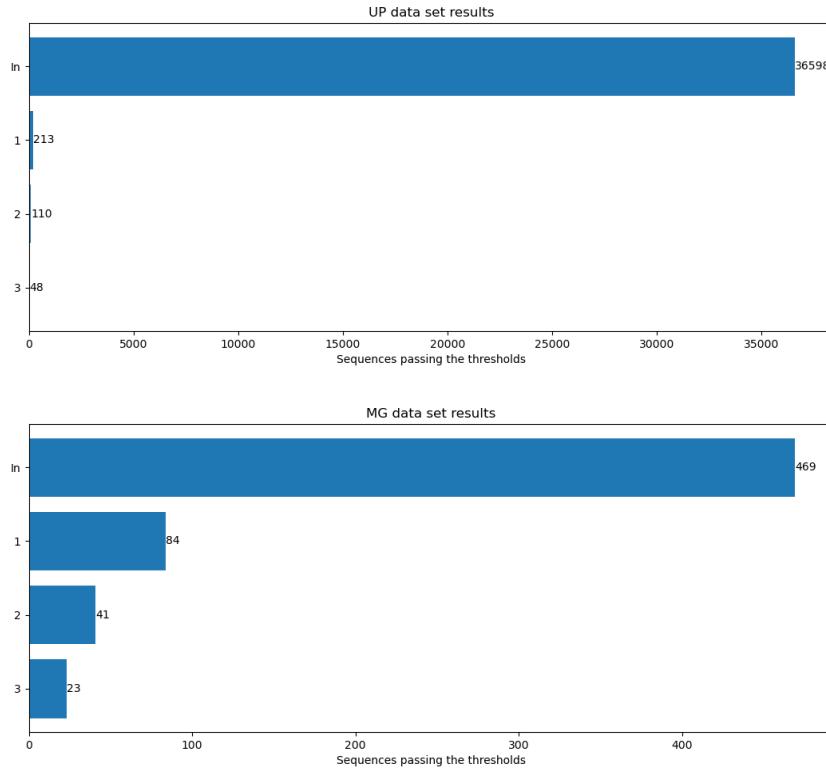


Figure 3.13: Summary of the results after subjecting the filtered UP and MG data set to the pipeline. 1=has an MR region; 2=Hydrophobicity score exceeds 0.50; 3=RSA exceeds 0.50

Only 0.13% of sequences fulfilled the three conditions in the UP data set, whereas for the MG data set, 4.91% of the input remained. Together, they represent 71 LMCOs with a hydrophobic, surface accessible MR region. Notably, no LMCOs from MGYS00005704 were retained, and out of the 23 candidates (all found in MGYS00005970), ten came from the control sample, seven from the PE1 sample, and six from the PE2 sample.

3.6.1 297 LMCOs with MR regions

Figure 3.14 summarizes the results of the first step; identifying MR regions. For the UP data set, 213 entries passed the q-stat and both sliding windows. 233 MR regions were present in these 213 sequences, meaning that some contain more than one non-overlapping MR region. 342 sequences passed the q-stat, but did not pass the sliding window thresholds. These are thus LMCOs where the Met residues tend to aggregate together, but they do not pass the minimum thresholds we have defined. Most of the sequences predicted by the static sliding window were also found by the dynamic sliding window. However, the dynamic sliding window predicted 209 sequences more than the static sliding window.

For the MG data set, 84 sequences containing 93 MR regions passed all three tests. Here, there seems to be more consensus between the three methods. Most sequences identified by the q-stat method also have an MR region passing the minimum thresholds. All sequences picked up by the dynamic sliding window were also predicted by the static sliding window.

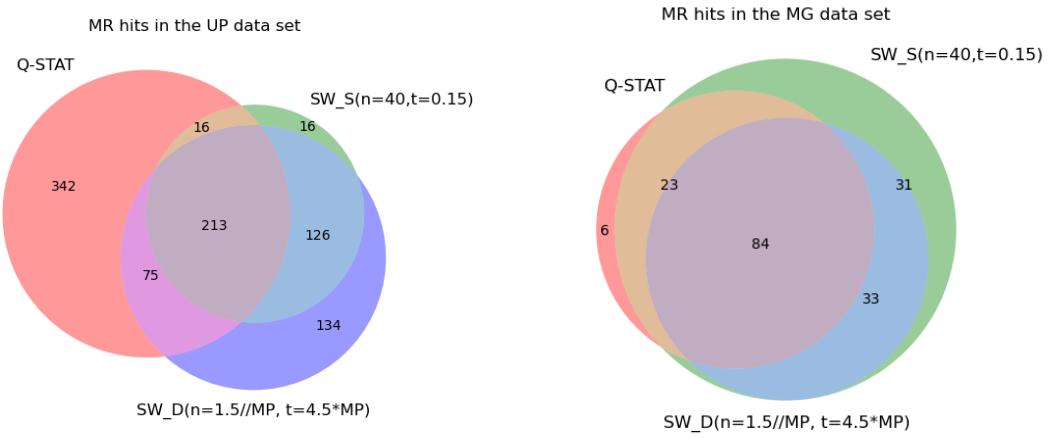


Figure 3.14: Identification of MR regions in the UP (left) and MG (right) data set using the q-statistic, a static sliding window (SW_S) and a dynamic sliding window (SW_D).

Figure 3.15 shows the length and MP distribution of the MR regions for both data sets. It can be seen that MR regions in the MG data set typically tend to be shorter and richer in Met. For the UP data set, the average length and MP of the MR regions was 52 amino acids and 0.20 respectively. In the MG data set, the mean length and MP was found to be 38 amino acids and 0.26 respectively.

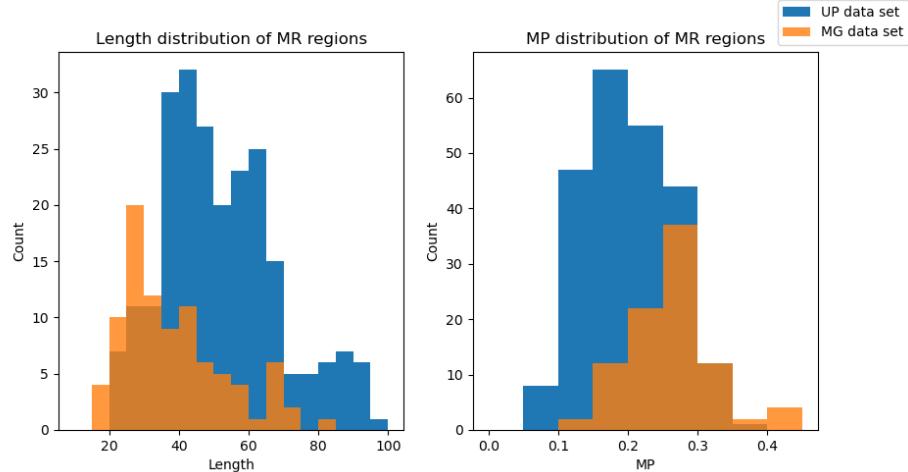


Figure 3.15: **Left:** Length distribution of the MR regions. **Right:** MP distribution of the MR regions. Blue is the UP data set, orange is the MG data set.

3.6.2 71 LMCOS pass hydrophobicity and RSA thresholds

We can now score the MR regions on hydrophobicity and RSA. Figure 3.16 summarizes the results for both data sets.

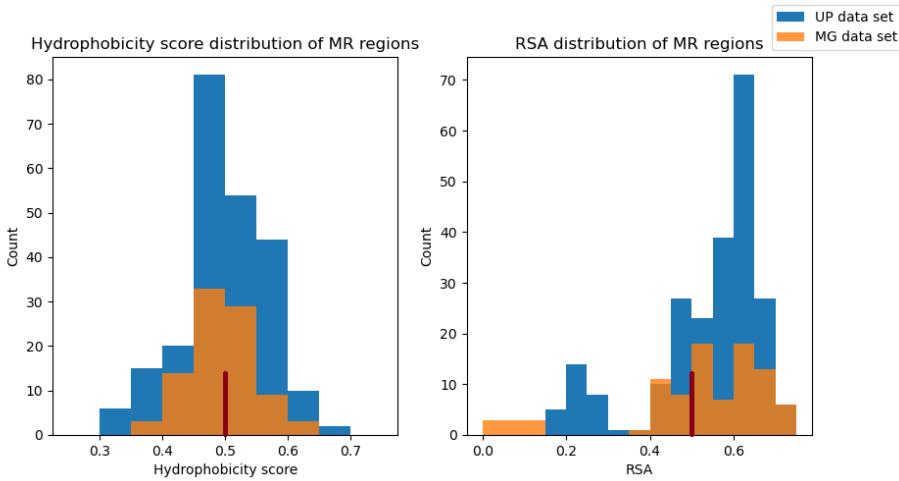


Figure 3.16: **Left:** Hydrophobicity score distribution of the MR regions. **Right:** RSA distribution of the MR regions. Blue is the UP data set, orange is the MG data set. Red bars indicate the thresholds.

The distribution of hydrophobicity scores and RSA is rather similar for both data sets. Mean hydrophobicity score was 0.49 for the both the UP and MG candidates. Mean RSA was 0.54 in the UP data set, and 0.52 in the MG data set. The distribution in RSA also clearly shows the distinction between buried and surface accessible MR regions.

In the end, 51 hydrophobic surface accessible MR regions coming from 48 LMCOs were found in the UP data set, and 23 hydrophobic surface accessible MR regions coming from 23 non-identical LMCOs were found in the MG data set. It was also noted that 67 out of 73 MR regions contain acidic residues (Glu or Asp), possibly involved in proton transfer (Orlando et al., 2025).

3.7 Further analysis of the candidates

3.7.1 Taxonomy and EC number of the UP candidates

31 of the 48 UP candidates were from fungal origin, and within the fungi, 30 belonged to the phylum of Ascomycota. The other one was a Basidiomycete. Eleven candidate LMCOs came from plants and only one from the Metazoan kingdom. Thus, in total, 44 of 48 candidates were eukaryotes, as was expected by the strong bias towards fungi and plants in this data set. The other four are indeed of bacterial origin, and no archaeal LMCOs were identified.

Concerning the EC classifications; 38 of the UP candidates did not have an EC number assigned to them. This could have been expected, since it was noticed during the data exploration that the vast majority of entries in this data set had an unknown EC number. Notably, all four bacterial LMCOs belonged to the nitrite reductases (EC 1.7.2.1). According to the literature, nitrite reductases are mostly 2dMCOs (Gräff et al., 2020), but these entries were, according to UniProt annotations, in fact 3dMCOs. Only two candidates were classified as laccases (EC 1.10.3.2), and the remaining four as ascorbate oxidases (EC 1.10.3.3).

3.7.2 One LMCO with Leu as axial ligand

In the UP data set, 4 out of 48 candidates could not matched with the regular expression for axial ligand determination. Two candidates, both fungi, did not have a Met as an axial ligand; one with Leu, and one with Trp. The remainder had Met. It is thus likely that the candidate with a Leu exhibits a higher reduction potential, making this the most valuable candidate identified. To the author's knowledge, no LMCO crystal structures with Trp at the axial ligand position are known, and it is unclear what its effect on the reduction potential will be.

In the MG data set, 6 out of 23 candidates could not be matched with the pattern, and the other 17 all had a Met as the axial ligand. Hence, candidates likely to exhibit a higher reduction potential were not found in this data set.

3.7.3 Clustering and alignment

MMSEQS2 *cluster* module was used to cluster the 71 candidates on 90% sequence identity. This resulted in 53 cluster representatives, indicating that some are highly similar, but most of them share less than 90% sequence identity. Additionally, none of the LMCOs from the MG data clustered together with a candidate from the UP data set, demonstrating that the MG candidates represent LMCOs that would not have been found in the UP data set.

The MSA between all candidates confirmed the shared presence of the copper binding sites, except for two sequences. It was also observed that the candidate with a Trp as its axial ligand might be erroneously predicted. The MSA pointed out that the RegEx pattern ('HCHxxxHxxxxA') was present, but there was an abnormal insertion in between the 'HCHxxxH' and the axial ligand position. This is shown in figure 3.17. Hence, according to the MSA, this LMCO does contain a Met at the axial ligand position, instead of the predicted Trp by the RegEx pattern. This will have to be confirmed through structural analysis.

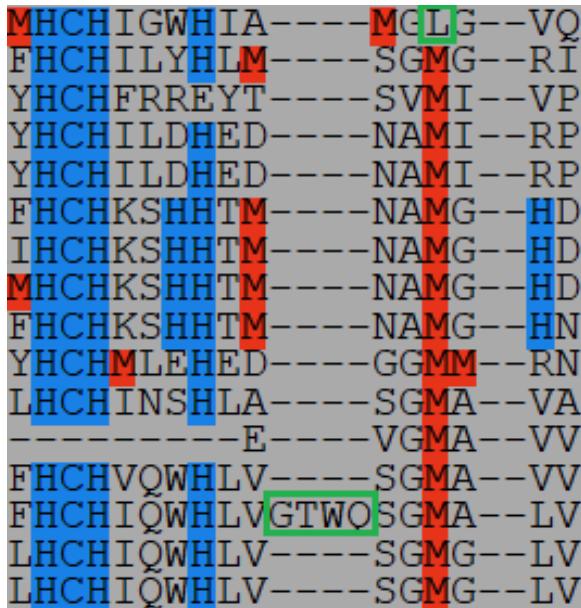


Figure 3.17: Part of the MSA between the clustered representatives of all candidates. His and Cys are colored blue. Met is colored red. Top green rectangle highlights the Leu found at the axial ligand position for one candidate. Bottom green rectangle highlights the insertion at the T1 binding site unique to this entry, which caused the Trp (W) to be incorrectly predicted as the axial ligand.

3.7.4 Predicted structures

Two candidates from the UP data set were selected for further visual investigation, which I will now term *up_can1* and *up_can2*. One candidate from the MG data set, *mg_can1*, was also visualized. Figure 3.18 shows the location of Met residues in each candidate compared to *bac_za23*.

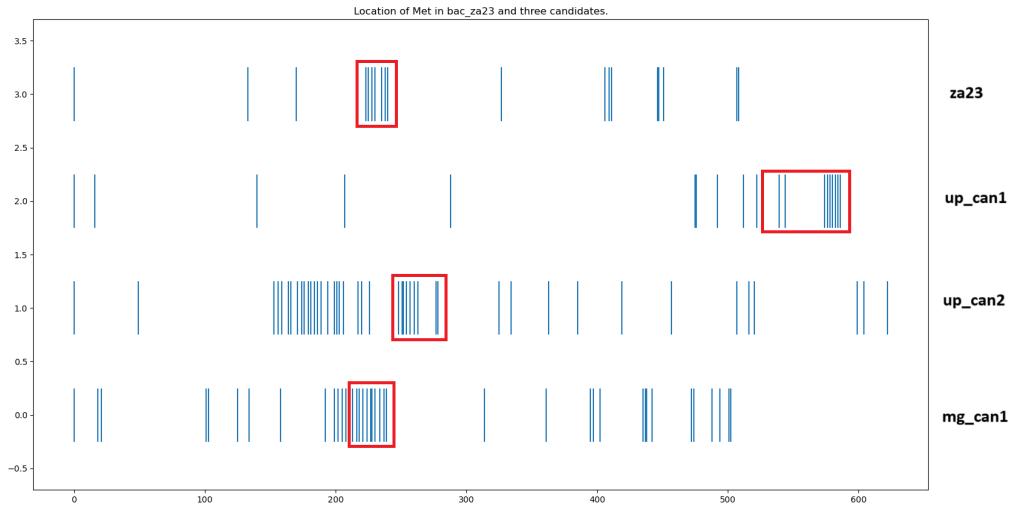


Figure 3.18: Location of Met in the three candidates compared to *bac_za23*. Predicted MR regions are surrounded by red rectangles.

Up_can1 is the LMCO with Leu as its axial ligand. It has a predicted MR region at the C-terminus of 65 amino acids long (positions 536-600), an MP of 0.14, a hydrophobicity score of 0.52, and an average RSA of 0.51. *Up_can2* has an MR region of 28 amino acids long (positions 242-269), an MP of 0.25, hydrophobicity score of 0.54, and an average RSA of 0.65. This is the LMCO for which Trp was predicted as the T1 axial ligand by the RegEx pattern. Lastly, *mg_can1* has an MR region of 33 amino acids long (positions 210-242) with an MP of 0.33, hydrophobicity score of 0.55, and an average RSA of 0.66. Figure 3.19 shows the predicted structures of the two UP candidates, and Figure 3.20 shows the predicted structure for *mg_can1*.

It can be seen that the MR regions are predicted to be in a highly disordered state. This was to be expected, since these are supposed to be flexible binding regions. In *up_can1*, the MR region seems to be in close sideways proximity T1 Cu site, which is also not hindered or covered by other regions (i.e., it seems to have an open binding pocket). This could also be seen when displaying the enzyme in surface mode in PyMol. In *up_can2*, however, the catalytic site seems to be more buried, and the MR region, almost looping around the entire protein, is much longer than what was actually predicted by the dynamic sliding window (as can be seen more clearly in figure 3.18). Hypothesizing how the enzyme would bind to a PE molecule is thus rather tricky for this enzyme. Additionally, the incorrectly predicted Trp can be seen at a distance from the expected T1 site, further confirming that Met is indeed the axial ligand for this LMCO.

Mg_can1 is most similar to *bac_za23* concerning the location and length of its MR region, but it is considerably richer in Met. The MR region nicely extends out of the catalytic pocket, seemingly encircling it. It can thus be hypothesized that once this MR region binds to PE, the T1 Cu site will follow and be drawn close to the substrate. However, the MR region is expected to be highly motile and flexible, and there is no way to robustly predict the distance between PE and T1 Cu just by looking at a static structure alone. For this, more advanced molecular dynamics simulations are required. Additionally, the MR region might even hamper the electron transfer by positioning itself in between PE and the catalytic pocket.

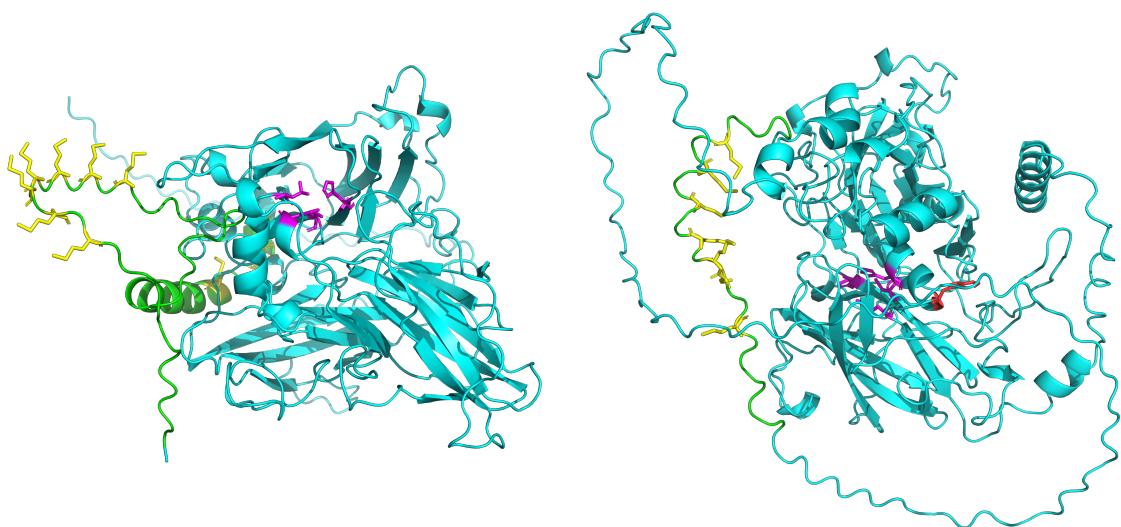


Figure 3.19: **Left:** *up_can1*. **Right:** *up_can2*. MR regions are depicted in green, with Met residues as yellow sticks. T1 copper site is depicted in magenta. Additionally, the incorrectly predicted axial ligand Trp in *up_can2* is depicted in red.

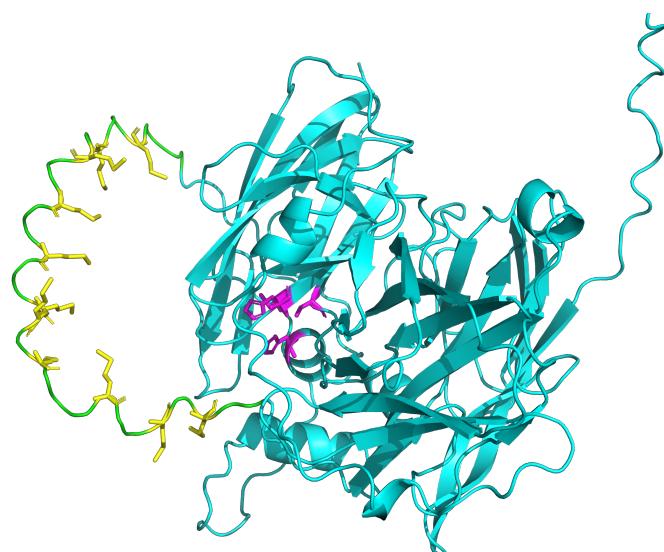


Figure 3.20: Predicted structure of *mg_can1*. MR regions are depicted in green, with Met residues as yellow sticks. T1 copper site is depicted in magenta.

Chapter 4

Discussion

4.1 Interpretation of results

In this work, the LMCO family of enzymes was identified to be involved in PE degradation. Then, a key set of qualities to which the enzyme must comply were defined: (i) the presence of an MR region that is both (ii) hydrophobic, and (iii) surface accessible. This would allow the catalytic T1 copper center to be drawn near the PE substrate and subtract an electron, with oxidation and chain scission to follow. 37,065 LMCOs coming from two large data sets were screened for these qualities, and 71 candidates were found. After clustering on 90% sequence identity, 53 cluster representatives remained, indicating that the candidates exhibit significant sequence diversity. Aligning the candidates indeed confirmed that all but two possessed the characteristic copper binding sites.

Furthermore, some variety could be observed concerning the location, length and composition of the predicted MR regions. Future experiments should be able to point out the optimal configuration. Should the MR binding region be located at the C- or N- terminus, somewhere in the middle? What is the optimal length and composition? If the MR region is located at the opposite end of the catalytic T1 copper site, the enzyme would be in no position to subtract an electron of the PE molecule. Ideally, the MR region should be positioned such that the T1 copper is drawn as close as possible to the substrate.

4.1.1 UP LMCOs not matching RegEx patterns

In the UP data set exploration, it was observed that 52% of sequences could not be matched with the RegEx patterns derived from the MSA of the PE-RE data set, and 23% did not contain the pattern 'HCH'. These results can mean several things. Most likely is that there still exists some variability in the copper binding sites that the regular expressions simply cannot capture. However, it was unexpected to see that almost a fourth of the sequences did not match the highly conserved 'HCH' pattern involved in electron transfer. This implies that these proteins have either adapted other mechanisms of electron transfer that do not rely on the His-Cys-His pathway, have lost the ability to bind copper, or are misclassified.

Some of these sequences were manually inspected and it was found that the two His residues are present most of the time, but the Cys residue was sometimes replaced with another amino acid. F.e. the *Eucalyptus* LMCO with UP identifier A0A059BT38 contains 'HLH' instead of 'HCH', and the same was found for the *Arabidopsis* LMCO with UP identifier A0A1P8BCG5. 'HVH' also occurred for some entries. However, other entries such as the *Fusarium* (fungi) and *Morus* (plant) LMCOs with UP identifier X0K4Z4 and W9S2S2 respectively did not even posses 'H.H' at all, which suggests that they are either misclassified or have lost the ability to bind copper.

Additionally, the large majority of plant LMCOs did not match the pattern for axial ligand determination, which also relies on the presence of an 'HCH' pattern. As explained in the previous paragraph, some variety in this pattern was observed for the plants, which is probably why they return a *None* value. Perhaps if the expression allows for more variety at the Cys position, a

better result can be obtained. Alternatively, an MSA can also point out what is happening at these locations in the sequence. Certainly, for *up_can2*, the erroneous axial ligand prediction was corrected by looking at the MSA of all candidates.

These results also pointed out that the RegEx patterns in this work are too strict and do not suffice for the identification of LMCOs and their axial ligands. Hence, other methods such as HMMs and MMSEQS2 had to be employed. Indeed, when searching the MG data set for LMCOs, RegEx patterns only identified one sequence that was not picked up by the HMM or MMSEQS2 methods.

4.1.2 Enrichment of MR regions in metagenomic sequences

Proportionally speaking, the MG data set contained significantly more candidate LMCOs than the UP data set, suggesting that hydrophobic MR regions might indeed be correlated with the presence of plastic debris. Although it is a remarkable finding, this cannot be said for certain.

Firstly, no candidates remained from the metagenomic study in the North Atlantic Ocean (MGYS00005704). If plastic debris indeed selected for LMCOs with hydrophobic MR binding regions, why were there no candidates found in this study? Secondly, the control sample from the second study (MGYS00005970) contained the most candidates. Unfortunately, it remains unclear what this control sample exactly was, as this cannot be found in the project information. However, it is certainly not a PE film, and given the assumptions of this work, this makes little sense. If the MR regions were indeed involved in PE degradation, the two PE samples (incubated for six months in marine conditions) should have contained more candidate LMCOs than the control.

Perhaps the enrichment of MR regions is somehow related to the marine environment, and not to the presence of PE per se. We should also keep in mind that the natural function of these MR regions will most likely be unrelated to PE degradation. To further investigate this, more metagenomic data can be screened, including samples from terrestrial environments. For this thesis, it was chosen to use studies from the MGNIFY database which already had predicted coding regions available. However, other studies such as the ones mentioned in section 1.3.2 can be found in the NCBI database (A. Kumar et al., 2024; R. Kumar et al., 2021). In this case, searches should be performed on the DNA level, or predicting open reading frames can be included in the analysis. Recently, metagenomic data from contaminated ponds in Belgium has also been made public, with PE present in the samples (Krzynowek et al., 2025). This study can also be screened for the presence of LMCOs and/or other enzymes involved in PE degradation.

4.1.3 Limitations of MR region prediction

Figure 3.18 showed the predicted MR regions for three of the identified candidates. A clear downside of the methodology can be seen, as *up_can2* contains an MR region much longer than what was predicted. In the predicted structure (Figure 3.19), the full MR region can also be seen as a long loop covering almost the entire protein, with the predicted MR region only comprising a small portion.

It was decided in section 3.5 that the dynamic sliding window would be used to predict the MR regions on the basis of a more accurate prediction for the known MR region in *E.coli*. This means that an MR region must exceed an MP proportion 4.5 times its entire MP over a window size of 1.5 divided by the MP. This LMCO thus needs to have a region of $1.5 // 0.067 = 22$ amino acids long, surpassing an MP of $4.5 \cdot 0.067 = 0.30$, which is very strict. Hence, the MR region preceding the predicted one probably did not exceed the threshold, and was thus not detected. A static sliding window would probably have predicted this entire region, showing that the choice of sliding window significantly affects the results. This could also be seen in the Venn diagram (Figure 3.14); for the UP data set, dynamic sliding windows picked up 209 sequences that the static one could not detect.

A second downside is that the methods are purely sequence based. Met residues need to cluster together along the primary structure to be detected. However, it might very well be possible that when the protein folds, distant Met residues come together in 3D space to form an MR cluster. Such cases cannot be detected by the methods provided in this work. An example of this might be *bac_zh22* which, despite having a catalytic pocket surrounded by numerous Met residues, could

not be detected by the q-stat and sliding window method because the Met residues do not cluster together along the sequence. Additionally, this sequence has an abnormally high MP of 0.052, which means that the dynamic thresholds will be rather strict.

To further improve the detection of MR regions, one could also resort to motif searches. For example, a hidden Markov Model with two transition states can be defined; inside MR region, and outside MR region. The model can be trained on the MR regions identified in this work, where it essentially learns to tell the difference between both states. A classic example of such a model is the occasionally dishonest casino switching between fair and loaded dice, but in this case the die is biased towards rolling Met residues (Durbin et al., 1998). This would likely allow for much faster results as well, since reshuffling every sequence in the UP data set 10,000 times for the q-stat calculation took approximately six hours.

4.2 Comparison to the ACTPAC approach

ACTPAC is EU-funded project also aiming to deconstruct PE into end products that can be repurposed. However, their approach differs from the one envisioned in this work. Instead of using enzymes isolated from living cells, ACTPAC aims to develop inorganic catalysts (e.g., zeolites) to selectively rearrange C-C bonds in PE, with C6-C18 alkanes as the end product. Although not specifically mentioned in their project description, my guess is that a fraction of these end products will be repurposed into fuels, oils and waxes of different kinds, which is rather attractive from an economic point of view. Furthermore, ACTPAC aims to engineer cytochrome oxidases to selectively oxidize C6-C12 alkanes to diols and diacids, which are then supposed to be used for the production of new polyester plastics (ACTPAC, 2025).

Thus, both the methods and end products differ. The approach proposed in this work is purely enzymatic and would likely result in a variety of oxygenated end products through the direct and unspecific oxidation of the PE polymers. The exact nature and length of these end products remain difficult to predict for now, but alcohols, ketones, and carboxylic acids of differing length can be expected, as suggested by Yao et al. (2022) and Zampolli et al. (2023). An enzymatic cascade following oxidation might result in a more narrowly defined range of end products. ACTPAC, on the other hand, first selectively cleaves the PE polymers into alkanes between 6 to 18 carbons long, without the incorporation of functional groups. Only afterwards do they turn towards enzymes to oxidize the terminal ends of alkanes, resulting in diacids and diols. Figure 4.1 highlights the difference between both pathways.

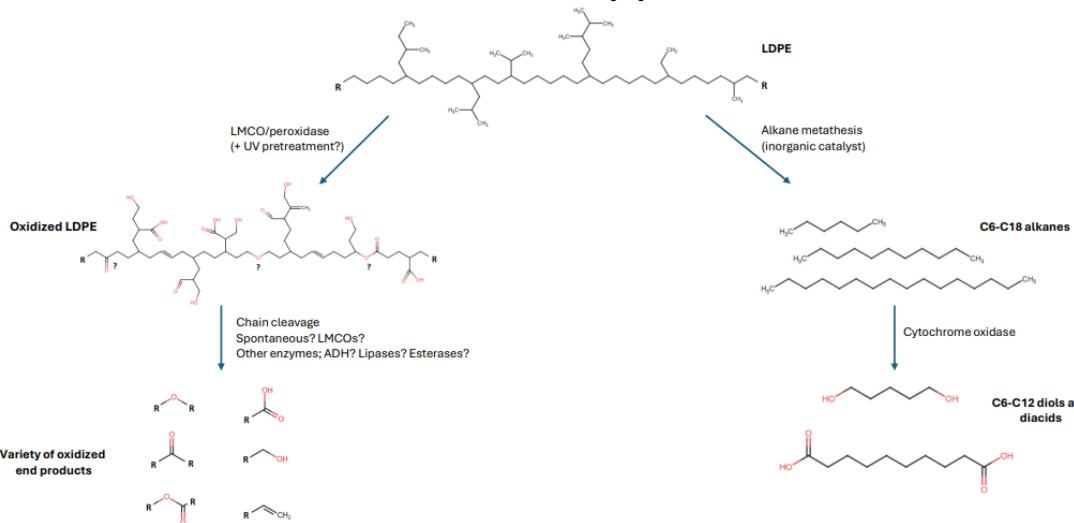


Figure 4.1: Simplified scheme of the pathway for PE depolymerization in this work (left), compared to the approach proposed by the ACTPAC project (right). Question marks on the left indicate that it still remains unclear if oxidation can take place along the main chains, as opposed to branching points. The exact mechanisms of chain cleavage after the oxidation also remains unclear. Additionally, the ACTPAC approach will probably not be limited to LDPE.

In the exploratory phases of this work, one of the ideas was to look for enzymes able to cleave C-C bonds without the incorporation of oxygenated functional groups, which would mimic the ACTPAC approach but with enzymes instead of inorganic catalysts. However, no studies reporting on such an enzymatic conversion could be found, and the energy required to cleave C-C bonds was thought to be unsurpassable. Additionally, to the author's knowledge, no enzyme catalyzing the cleavage of long C-C chains has ever been found in nature. Perhaps it is possible to engineer such an enzyme, but the focus was kept on enzymes already known to act on PE, which happened to be through an oxidative mechanism.

4.3 Future perspectives

4.3.1 Pocket analysis can provide more informed decisions

Y. Zhang et al. (2023) also developed a computer-aided strategy to look for potential PE-degrading enzymes. Here, the criteria were less strict; the enzyme must possess (i) a metal catalytic center, and (ii) an open substrate binding pocket. In contrast to this work, the researchers do not screen for the presence of a hydrophobic binding region. However, the plea for an open substrate binding pocket is equally relevant. The further the T1 catalytic center is buried in the enzyme, the more difficult it will be to approach the bulky PE substrate, which is many times larger than the enzyme itself. Indeed, the MR helix described in *E. coli* is positioned such that it completely covers the catalytic copper center, thereby hindering the access of larger bulky substrates (Kataoka et al., 2007; Roberts et al., 2002). Hence, despite the presence of an MR region experimentally shown to bind carbon nanotubes by Cui et al. (2021), this LMCO would not make a good PE-degrading candidate due to its catalytic center being inaccessible to the surface.

To resolve this issue, further computational work could be done through pocket descriptor tools as designed by a previous thesis student (van Meegen, 2024). This tool can locate putative binding pockets, describe their shape, and tell how narrow or deep they are. Additionally, amino acid composition of the binding pocket can also be determined. In this case, open substrate binding pockets with hydrophobic residues would make good candidates. This sort of analysis could provide a more informed decision as to how likely it really is for the LMCO to oxidize PE.

4.3.2 Experiment proposal

The central claim of this thesis is that the identified candidates can be tested in an experimental setting in order to determine if this has been a viable screening strategy. How should one actually proceed to do this? What kind of experiments can be thought of? Currently, four open questions remain; (i) How and where does one obtain PE? (ii) How does one express and purify the identified LMCO candidates? (iii) Under which conditions should the two be combined? (iv) What are the end products and how can they be repurposed? I shall now proceed to comment on these questions and propose further avenues of research.

Source and identification of PE

The first requirement is a well-defined source of PE. There is an important difference between commercial PE as it is found on the market, and lab-grade PE that one would buy or synthesize with full control over its composition. Commercial PE will, depending on the product, contain varying degrees of branching, crystallinity, molecular weight, along with the presence of numerous additives, all of which are not known beforehand. Additionally, PE found discarded in the environment has already been partially degraded, and exists in many different forms (macro-, micro-, nanoplastics, films, fibers, beads...). In the ACTPAC framework, experiments are first performed on lab-grade model PE compounds as a proof of concept (ACTPAC, 2025). Only afterwards are the methods extrapolated to real-life PE. Hence, I propose a similar approach.

Pilot experiments could be performed on small amounts of lab-grade LDPE with a relatively low molecular weight and no additives. I propose LDPE, since there exists no evidence of LMCOs acting on HDPE. Gradually, however, the amount of branching can be decreased, and the molecular weights and crystallinity increased. This should point out what types of PE can be oxidized by the candidate enzymes, and what the limit is concerning the degree of branching, crystallinity, and molecular weight, all important factors contributing to the degradability of PE. Additionally, I propose experiments to be performed on homogenized PE powder or granules instead of films or strips, since this provides the largest surface availability. The optimal size and diameter of the particles can be also be determined experimentally.

Eventually, however, the end goal is to degrade real-life PE as it is found on the market and as waste in the environment. This requires a proper sampling of commercially available and waste PE to determine in what form and state it is typically found. Ideally, various locations and product types are sampled and fully characterized. That is; FTIR or ATR-FTIR to confirm it is indeed PE, and to determine the abundance of additives, as well as the current degree of degradation. Secondly, (HT-)GPC to determine distributions of molecular weight. Thirdly, DSC, XPS or XRD to determine the degree of crystallinity. A combination of FTIR and DSC has already been shown to distinguish LDPE, LLDPE and HDPE in marine plastic debris (Lynch et al., 2024). The amount and nature of additives will likely be of importance when going to larger scales. Considering the low specificity of LMCOs, additives could be equally good substrates, possibly hindering the enzyme's activity. Hence, an experiment with and without common additives should point out its effects.

Expression and purification of LMCOs

Once a properly characterized source of PE is available, the second step is to express and purify the LMCOs put forth in this work. This would require culturing the host organism and inducing LMCO production, f.e. by adding $CuSO_4$ to the medium (Sodhi et al., 2024; Zampolli et al., 2023). Alternatively, since the source organisms of sequences derived from metagenomic studies can often times not be cultured, the genes of interest are chemically synthesized and transformed into an expression system. Afterwards, the enzyme can be isolated and purified. It should then be tested against model compounds such as ABTS or HBT, to confirm laccase-like activity. Following this, optimal temperature and pH can also be determined.

However, LMCOs are notoriously difficult to express heterologously, suffering from low yields and high purification costs (Sodhi et al., 2024). Currently, the yeast *Pichia pastoris* or the bacterium *E. coli* are the most commonly used expression systems. However, efforts are still being made to

optimize the process by searching more suitable hosts, improving culture media and other cultivation parameters (e.g., stronger promoters, co-culturing techniques) (Sodhi et al., 2024; Yang et al., 2017).

Degradation assays

Once PE and purified LMCOs are present, the two can be combined in a test tube or another recipient to evaluate degradative capabilities. Main parameters to consider are temperature, pH, duration of the reaction, and amount of enzyme. Temperature and pH should be tailored to the enzyme's optimal values as determined in previous assays. Considering the highly recalcitrant nature of PE at room temperature, I hypothesize that the process would greatly benefit from elevated temperatures, as the system would be in a higher energy state, making it easier for the LMCO to subtract an electron. This is also suggested by Zampolli et al. (2023), who have been the only ones to perform the reaction at 60°C. However, this emphasizes the need for a thermostable LMCO.

Furthermore, I would expect alkaline conditions to be preferred over acidic ones. Currently, the mechanism of action is thought to involve an electron-proton transfer (Orlando et al., 2025; Zampolli et al., 2023). The electron is of course absorbed by the LMCO and passed on to dioxygen. The proton is presumably transferred to a negatively charged acidic residue (Asp or Glu) in its conjugate base form. Hence, an acidic milieu would cause the equilibrium to shift towards protonated residues in the enzyme, hampering the ability to absorb a proton from the substrate. In alkaline conditions, all the acidic residues will be in their negatively charged form, thereby facilitating proton transfer.

Additionally, it would be worthwhile to confirm that the MR regions indeed bind to the PE surface. Cui et al. (2021), who showed the rapid immobilization of *E. coli*'s LMCO on carbon nanotubes through the MR helix, have devised an experiment to visualize this using green fluorescent proteins (GFP). GFP was first expressed on its own, and the carbon nanotubes treated with the protein. Since it does not contain a hydrophobic MR region, binding on the nanotube surface did not occur, and no fluorescence was observed. However, when the nanotubes were treated with a GFP-LMCO fusion protein, a fluorescent signal could be seen on the surface. A similar experiment might be devised for PE films to further strengthen the argument that the MR region indeed immobilizes the enzyme on the surface.

Finally, to confirm that oxidation has taken place, (ATR-)FTIR should be used to calculate changes in carbonyl indices, and GPC can be used to track changes in molecular weight. When working with powders or granules, AFM (atomic force microscopy) can detect changes in the size of the particles, as was also done by Oiffer et al. (2024). When working with films or strips, WCA (water contact angles) can be used to detect changes in surface hydrophobicity and SEM (scanning electron microscopy) should point out morphological changes of the material. Another approach would be to use radioactively or isotopically labeled PE and track the movement of the labeled carbon atoms (which should then be detected in the end products). This approach was also applied by Albertsson et al. (1987), who performed one of the first long-term experiments on PE biodegradation.

End products

The end products of the enzymatic treatments can be determined by GC-MS and/or LC-MS. Zampolli et al. (2023) obtained C8 and C16 carboxylic acids by treating low molecular weight LDPE powder at 60°C with their LMCO isolated from *R. opacus* R7. This suggests that one LMCO, under the right conditions, can already result in end products of low molecular weight. However, alkanes, alcohols and ketones of varying length were also found, indicating that a variety of oxidized end products remain. In such a case, these should be further purified to obtain a narrowly defined range of end products.

On the contrary, Oiffer et al. (2024) argue that no single enzyme will cause the complete depolymerization of PE. Instead, as mentioned in the introduction, they propose an enzymatic cascade with a peroxidase as the first step. Most strikingly is the fact that their experiment required both chemical and physical pretreatments, followed by at least four enzymes to obtain carboxylic acids as end products, whereas Zampolli et al. (2023) only needed one LMCO, without pretreatment, to essentially arrive at the same result. My hypothesis is that the elevated temperature of 60°C in

Zampolli et al. (2023), as opposed to 20°C for the oxidative step in Oiffer et al. (2024), explains these discrepancies.

Despite Zampolli et al. (2023) showing that one enzyme suffices for carboxylic acid end products, a case can still be made for enzymatic cascades, as this would allow for much greater control over the end products. On the other hand, one needs to characterize, describe and isolate more enzymes, demanding more time, money and energy.

Assuming that carboxylic acids of varying length will be the result, the question still remains on what to do with them. As mentioned before, ACTPAC also eventually arrives at carboxylic acids through a different pathway. They intend to use them for the production of new polyester plastics (ACTPAC, 2025). However, other avenues can be explored (e.g., manufacturing of pharmaceuticals or soaps).

4.3.3 Towards industrial applications

Given that the results of small scale experiments are positive, efforts should to be made to scale up the process. This requires input from multiple fields of research such as waste management and processing, engineering of industrial bioreactors, and the marketing of end products. PE waste must first be collected and separated from other plastics at a massive scale, which is already a complicated engineering challenge on its own. Today, a variety of technologies are being employed for efficient waste processing and sorting, e.g., magnetic separation, laser/optical sorting, digital watermarking and machine learning (Plastics Europe, 2025). Afterwards, PE will likely need to be processed into a homogeneous powder and incubated with the enzyme in large bioreactors, similar to how *Carbios* tackles the deconstruction of PET. These bioreactors would require a constant flow of oxygen for the LMCOs to work.

Three types of industrial LMCO applications are currently distinguished: (i) Free LMCO systems, (ii) immobilized LMCO systems, and (iii) LMCO mediator systems (Bonnet et al., 2025). Free LMCO systems refer to free-living enzymes in solution with the substrate. This approach benefits from greater substrate availability, but does not allow for the recovery of enzymes offered by immobilized systems. On the other hand, immobilizing an enzyme on a surface often leads to loss of activity. Free LMCO systems also allow for the use of crude extracts instead of purified enzymes, which would reduce purification costs. Mediator systems rely on the presence of other natural or synthetic molecules acting as electron shuttles between the LMCO and the substrate, which allows for a larger substrate scope. However, this approach suffers from additional costs of the mediators, which sometimes need to be supplied in high ratios (Bonnet et al., 2025). All three approaches have their pros and cons. Given that we're expecting the enzyme to directly bind to PE, a mediator system would make less sense for the candidates identified in this work. It remains to be seen what will work best.

Then there is still the issue of a cost-effective large-scale production and purification of LMCOs. Heterologous expression of LMCOs currently lacks the desired yields and stability required for industrial applications. Research in this field is focused on finding naturally hyper-expressive LMCO strains, improving fermentation techniques, and increasing expression levels though stronger promoters, optimized codon usage, and better culturing conditions (Sodhi et al., 2024; Yang et al., 2017). Nevertheless, a wide range of companies such as *Novozymes* (now *Novenesis*), *Creative Enzymes* and *MycoTechnology* already have LMCO products on the market for an equally wide range of applications such as denim bleaching, removal of bitterness in foods, and delignification (Sodhi et al., 2024).

Lastly, another bottleneck that I foresee is the tendency of LMCOs to both polymerize and depolymerize. LMCOs operate in some kind of equilibrium between both actions, as suggested by their role in the biosynthesis of lignin in plants, and the degradation of lignin in fungi. Considering the low specificity of LMCOs, breakdown products might become equally good substrates as PE itself. Hence, once sufficient oligomeric products are in the reaction mixture, the predominant reaction might be to polymerize instead of depolymerize. This could then result in polymers linked by esters or ether bonds, which could precipitate in the reaction mixture. Although this has not been observed on small scales, the issue might present itself on larger scales.

Conclusion

Seventy one candidate PE-degrading LMCOs have been discovered through a computational screening method based on three requirements that appeared relevant for PE binding and oxidation in the literature: all candidates posses (i) a Met-rich region that is both (ii) hydrophobic and (iii) surface-accessible. The candidates exhibit considerable sequence diversity, and the location, composition, and length of the binding regions were found to vary substantially. Future computational work includes optimizing the MR region prediction, screening more (metagenomic) data, and studying the binding pocket of candidate enzymes. Future wet-lab experiments should point out the optimal configuration of the MR regions, and if this screening strategy contains any real-life value at all.

Bibliography

- A. Zhang, A., Hou, Y., Wang, Q., et al. (2022). Characteristics and polyethylene biodegradation function of a novel cold-adapted bacterial laccase from antarctic sea ice psychrophile psychrobacter sp. NJ228. *Journal of Hazardous Materials*, 439, 129656. <https://doi.org/10.1016/j.jhazmat.2022.129656>
- ACTPAC. (2025). Actpac. <https://www.actpac.eu/>
- Albertsson, A.-C., Andersson, S., & Karlsson, S. (1987). The mechanism of biodegradation of polyethylene. *Polymer Degradation and Stability*, 18(1), 73–87. [https://doi.org/10.1016/0141-3910\(87\)90084-X](https://doi.org/10.1016/0141-3910(87)90084-X)
- Albertsson & Karlsson. (1990). The influence of biotic and abiotic environments on the degradation of polyethylene. *Progress in Polymer Science*, 15(2), 177–192. [https://doi.org/10.1016/0079-6700\(90\)90027-X](https://doi.org/10.1016/0079-6700(90)90027-X)
- Aledo, J. C. (2022). A Census of Human Methionine-Rich Prion-like Domain-Containing Proteins. *Antioxidants*, 11(7), 1289. <https://doi.org/10.3390/antiox11071289>
- Amaral-Zettler, L. A., Zettler, E. R., & Mincer, T. J. (2020). Ecology of the plastisphere [Publisher: Nature Publishing Group]. *Nature Reviews Microbiology*, 18(3), 139–151. <https://doi.org/10.1038/s41579-019-0308-0>
- American Chemical Society. *Leo hendrick baekeland and the invention of bakelite*. en. <https://www.acs.org/education/whatischemistry/landmarks/bakelite.html>. 1993, November 9.
- Andler, R., Tiso, T., Blank, L., et al. (2022). Current progress on the biodegradation of synthetic plastics: From fundamentals to biotechnological applications. *Reviews in Environmental Science and Bio/Technology*, 21(4), 829–850. <https://doi.org/10.1007/s11157-022-09631-2>
- Bielli, P., & Calabrese, L. (2002). Structure to function relationships in ceruloplasmin: A moon-lighting protein. *Cellular and Molecular Life Sciences CMLS*, 59(9), 1413–1427. <https://doi.org/10.1007/s00018-002-8519-2>
- Black & Mould. (1991). Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Analytical Biochemistry*, 193(1), 72–82. [https://doi.org/10.1016/0003-2697\(91\)90045-U](https://doi.org/10.1016/0003-2697(91)90045-U)
- Blum, M., Andreeva, A., Florentino, L. C., et al. (2025). InterPro: The protein sequence classification resource in 2025. *Nucleic Acids Research*, 53(D1), D444–D456. <https://doi.org/10.1093/nar/gkae1082>
- Bonnet, O., Fa'aui, T., Leung, I. K., et al. (2025). Challenges and applications of laccase in bioremediation. In *Laccase and Polyphenol Oxidase* (pp. 153–185). Elsevier. <https://doi.org/10.1016/B978-0-443-13301-5.00007-X>
- Bryant, J. A., Clemente, T. M., Viviani, D. A., et al. (2016). Diversity and Activity of Communities Inhabiting Plastic Debris in the North Pacific Gyre [Publisher: American Society for Microbiology]. *mSystems*, 1(3), 10.1128/msystems.00024–16. <https://doi.org/10.1128/msystems.00024-16>
- Buchholz, P. C. F., Feuerriegel, G., Zhang, H., et al. (2022). Plastics degradation by hydrolytic enzymes: The plastics-active enzymes database—PAZy. *Proteins: Structure, Function, and Bioinformatics*, 90(7), 1443–1456. <https://doi.org/10.1002/prot.26325>
- Chamas, A., Moon, H., Zheng, J., et al. (2020). Degradation Rates of Plastics in the Environment [Number: 9 Publisher: American Chemical Society]. *ACS Sustainable Chemistry & Engineering*, 8(9), 3494–3511. <https://doi.org/10.1021/acssuschemeng.9b06635>

- Chang, A., Jeske, L., Ulbrich, S., et al. (2021). BRENDA, the ELIXIR core data resource in 2021: New developments and updates. *Nucleic Acids Research*, 49(D1), D498–D508. <https://doi.org/10.1093/nar/gkaa1025>
- Cortes, L., Wedd, A. G., & Xiao, Z. (2015). The functional roles of the three copper sites associated with the methionine-rich insert in the multicopper oxidase CueO from *E. coli*. *Metalomics*, 7(5), 776–785. <https://doi.org/10.1039/c5mt00001g>
- Crawford & Quinn. (2017, February 1). *Microplastic pollutants*. Elsevier Inc.
- Cui, H., Zhang, L., Söder, D., et al. (2021). Rapid and Oriented Immobilization of Laccases on Electrodes via a Methionine-Rich Peptide [Publisher: American Chemical Society]. *ACS Catalysis*, 11(4), 2445–2453. <https://doi.org/10.1021/acscatal.0c05490>
- Cuthbertson, A., Lincoln, C., Miscall, J., et al. (2024). Characterization of polymer properties and identification of additives in commercially available research plastics [Publisher: Royal Society of Chemistry]. *Green Chemistry*, 26(12), 7067–7090. <https://doi.org/10.1039/D4GC00659C>
- Ducros, V., Brzozowski, A. M., Wilson, K. S., et al. (2001). Structure of the laccase from *Coprinus cinereus* at 1.68 Å resolution: evidence for different type 2 Cu- depleted isoforms. *Acta Crystallographica Section D*, 57(2), 333–336. <https://doi.org/10.1107/S0907444900013779>
- Ducros, V., Brzozowski, A., Wilson, K. S., et al. (1998). Crystal structure of the type-2 Cu depleted laccase from *Coprinus cinereus* at 2.2 Å resolution [Publisher: Nature Publishing Group]. *Nature Structural Biology*, 5(4), 310–316. <https://doi.org/10.1038/nsb0498-310>
- Durão, P., Bento, I., Fernandes, A. T., et al. (2006). Perturbations of the T1 copper site in the CotA laccase from *Bacillus subtilis*: Structural, biochemical, enzymatic and stability studies. *JBIC Journal of Biological Inorganic Chemistry*, 11(4), 514–526. <https://doi.org/10.1007/s00775-006-0102-0>
- Durbin, R., Eddy, S., Krogh, A., et al. (1998, April). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (1st ed.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511790492>
- Edgar, R. C. (2021). Muscle v5 enables improved estimates of phylogenetic tree confidence by ensemble bootstrapping. *bioRxiv*.
- Engbersen & de Groot. (2017). *Bio-organische chemie voor levenswetenschappen*. Wageningen Academic Publishers.
- Enguita, F. J., Martins, L. O., Henriques, A. O., et al. (2003). Crystal Structure of a Bacterial Endospore Coat Component: A LACCASE WITH ENHANCED THERMOSTABILITY PROPERTIES [Publisher: Elsevier]. *Journal of Biological Chemistry*, 278(21), 19416–19425. <https://doi.org/10.1074/jbc.M301251200>
- European Commission. (2021). *Turning the tide on single-use plastics*. Publications Office of the European Union. <https://doi.org/doi/10.2779/800074>
- Fox, N. K., Brenner, S. E., & Chandonia, J.-M. (2014). SCOPe: Structural Classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Research*, 42(D1), D304–D309. <https://doi.org/10.1093/nar/gkt1240>
- Fujisawa, M., Hirai, H., & Nishida, T. (2001). Degradation of polyethylene and nylon-66 by the laccase-mediator system. *Journal of Polymers and the Environment*, 9(3), 103–108. <https://doi.org/10.1023/A:1020472426516>
- Gambarini, V., Pantos, O., Kingsbury, J. M., et al. (2022). PlasticDB: A database of microorganisms and proteins linked to plastic biodegradation. *Database*, 2022, baac008. <https://doi.org/10.1093/database/baac008>
- Gasteiger, E., Hoogland, C., Gattiker, A., et al. (2005). Protein Identification and Analysis Tools on the ExPASy Server. In J. M. Walker (Ed.), *The Proteomics Protocols Handbook* (pp. 571–607). Humana Press. <https://doi.org/10.1385/1-59259-890-0:571>
- Geyer, R. (2020). A brief history of plastics. In M. Streit-Bianchi, M. Cimadevila, & W. Trettnak (Eds.), *Mare plasticum - the plastic sea: Combating plastic pollution through science and art* (pp. 31–47). Springer International Publishing. https://doi.org/10.1007/978-3-030-38945-1_2

- Ghatge, S., Yang, Y., Ahn, J.-H., et al. (2020). Biodegradation of polyethylene: A brief review [Number: 1]. *Applied Biological Chemistry*, 63(1), 27. <https://doi.org/10.1186/s13765-020-00511-3>
- Glazunova, O. A., Polyakov, K. M., Moiseenko, K. V., et al. (2018). Structure-function study of two new middle-redox potential laccases from basidiomycetes *Antrodiaella faginea* and *Steccherinum murashkinskyi*. *International Journal of Biological Macromolecules*, 118, 406–418. <https://doi.org/10.1016/j.ijbiomac.2018.06.038>
- Gouy, M., Guindon, S., & Gascuel, O. (2010). SeaView Version 4: A Multiplatform Graphical User Interface for Sequence Alignment and Phylogenetic Tree Building. *Molecular Biology and Evolution*, 27(2), 221–224. <https://doi.org/10.1093/molbev/msp259>
- Gräff, M., Buchholz, P., Le Roes-Hill, M., et al. (2020). Multicopper oxidases: Modular structure, sequence space, and evolutionary relationships. *Proteins: Structure, Function, and Bioinformatics*, 88(10), 1329–1339. <https://doi.org/10.1002/prot.25952>
- Granja-Travez, R. S., Wilkinson, R. C., Persinoti, G. F., et al. (2018). Structural and functional characterisation of multi-copper oxidase CueO from lignin-degrading bacterium *Ochrobaculum* sp. reveal its activity towards lignin model compounds and lignosulfonate. *The FEBS Journal*, 285(9), 1684–1700. <https://doi.org/10.1111/febs.14437>
- Guo, Z. (2025, April). Proposing the ACTPAC Framework to Enhance Industrial Recycling and Upcycling of Plastic Waste. Retrieved May 18, 2025, from <https://www.actpac.eu/knowledge/proposing-the-actpac-framework-to-enhance-industrial-recycling-and-upcycling-of-plastic-waste/>
- Gyung Yoon, M., Jeong Jeon, H., & Nam Kim, M. (2012). Biodegradation of Polyethylene by a Soil Bacterium and AlkB Cloned Recombinant Cell. *Journal of Bioremediation & Biodegradation*, 03(04). <https://doi.org/10.4172/2155-6199.1000145>
- Hakulinen, N., Kiiskinen, L.-L., Kruus, K., et al. (2002). Crystal structure of a laccase from Melanocarpus albomyces with an intact trinuclear copper site [Publisher: Nature Publishing Group]. *Nature Structural Biology*, 9(8), 601–605. <https://doi.org/10.1038/nsb823>
- Hakulinen & Rouvinen. (2015). Three-dimensional structures of laccases. *Cellular and Molecular Life Sciences*, 72(5), 857–868. <https://doi.org/10.1007/s0018-014-1827-5>
- Hawkins, L. W. (1984). *Polymer degradation and stabilization*. Springer-Verlag Berlin Heidelberg.
- Hiemstra, A.-F., Gravendeel, B., & Schilthuizen, M. (2025). Birds documenting the Anthropocene: Stratigraphy of plastic in urban bird nests. *Ecology*, 106(2), e70010. <https://doi.org/10.1002/ecy.70010>
- Høie, M. H., Kiehl, E. N., Petersen, B., et al. (2022). NetSurfP-3.0: Accurate and fast prediction of protein structural features by protein language models and deep learning. *Nucleic Acids Research*, 50(W1), W510–W515. <https://doi.org/10.1093/nar/gkac439>
- Iiyoshi, Y., Tsutsumi, Y., & Nishida, T. (1998). Polyethylene degradation by lignin-degrading fungi and manganese peroxidase. *Journal of Wood Science*, 44(3), 222–229. <https://doi.org/10.1007/BF00521967>
- Kataoka, K., Komori, H., Ueki, Y., et al. (2007). Structure and Function of the Engineered Multi-copper Oxidase CueO from *Escherichia coli*—Deletion of the Methionine-Rich Helical Region Covering the Substrate-Binding Site. *Journal of Molecular Biology*, 373(1), 141–152. <https://doi.org/10.1016/j.jmb.2007.07.041>
- Kosinas, C., Zerva, A., Topakas, E., et al. (2023). Structure–function studies of a novel laccase-like multicopper oxidase from *Thermothelomyces thermophila* provide insights into its biological role. *Acta Crystallographica. Section D, Structural Biology*, 79(Pt 7), 641–654. <https://doi.org/10.1107/S2059798323004175>
- Krzynowek, A., Van de Moortel, B., Pichler, N., et al. (2025). Effects of microplastics on Daphnia-associated microbiomes in situ and in vitro. *The ISME Journal*, 19(1), wrae234. <https://doi.org/10.1093/ismejo/wrae234>
- Kumar, A., Lakhawat, S. S., Singh, K., et al. (2024). Metagenomic analysis of soil from landfill site reveals a diverse microbial community involved in plastic degradation. *Journal of Hazardous Materials*, 480, 135804. <https://doi.org/10.1016/j.jhazmat.2024.135804>

- Kumar, R., Pandit, P., Kumar, D., et al. (2021). Landfill microbiome harbour plastic degrading genes: A metagenomic study of solid waste dumping site of Gujarat, India. *Science of The Total Environment*, 779, 146184. <https://doi.org/10.1016/j.scitotenv.2021.146184>
- Le Viet, L. H., Matsuda, T., & Leung, I. (2025). Laccases: Biological functions and potential applications. In *Laccase and Polyphenol Oxidase* (pp. 133–152). Elsevier. <https://doi.org/10.1016/B978-0-443-13301-5.00006-8>
- Lear, G., Maday, S. D. M., Gambarini, V., et al. (2022). Microbial abilities to degrade global environmental plastic polymer waste are overstated [Publisher: IOP Publishing]. *Environmental Research Letters*, 17(4), 043002. <https://doi.org/10.1088/1748-9326/ac59a7>
- Lontie, R. (1984a). *Copper proteins and copper enzymes* (Vol. 1). CRC Press, Inc.
- Lontie, R. (1984b). *Copper proteins and copper enzymes* (Vol. 3). CRC Press, Inc.
- Lyashenko, A. V., Bento, I., Zaitsev, V. N., et al. (2006). X-ray structural studies of the fungal laccase from Cerrena maxima. *JBIC Journal of Biological Inorganic Chemistry*, 11(8), 963–973. <https://doi.org/10.1007/s00775-006-0158-x>
- Lynch, J. M., Corniuk, R. N., Brignac, K. C., et al. (2024). Differential scanning calorimetry (DSC): An important tool for polymer identification and characterization of plastic marine debris. *Environmental Pollution*, 346, 123607. <https://doi.org/10.1016/j.envpol.2024.123607>
- Martin, E., Dubessay, P., Record, E., et al. (2024). Recent advances in laccase activity assays: A crucial challenge for applications on complex substrates. *Enzyme and Microbial Technology*, 173, 110373. <https://doi.org/10.1016/j.enzmictec.2023.110373>
- Mellidou, I., & Kanellis, A. K. (2024). Revisiting the role of ascorbate oxidase in plant systems. *Journal of Experimental Botany*, 75(9), 2740–2753. <https://doi.org/10.1093/jxb/erae058>
- Miranda-Blancas, R., Avelar, M., Rodriguez-Arteaga, A., et al. (2021). The β -hairpin from the *Thermus thermophilus* HB27 laccase works as a pH-dependent switch to regulate laccase activity. *Journal of Structural Biology*, 213(2), 107740. <https://doi.org/10.1016/j.jsb.2021.107740>
- Oiffer, T., Leipold, F., Süss, P., et al. (2024). Chemo-Enzymatic Depolymerization of Functionalized Low-Molecular-Weight Polyethylene. *Angewandte Chemie International Edition*, 63(50), e202415012. <https://doi.org/10.1002/anie.202415012>
- Orlando, C., Bellei, M., Zampolli, J., et al. (2025). Comparative analysis of polyethylene-degrading laccases: Redox properties and enzyme-polyethylene interaction mechanism. *ChemSusChem*, n/a(n/a), e202402253. <https://doi.org/https://doi.org/10.1002/cssc.202402253>
- Osipov, E., Polyakov, K., Kittl, R., et al. (2014). Effect of the L499M mutation of the ascomycetous *Botrytis aclada* laccase on redox potential and catalytic properties [Publisher: International Union of Crystallography]. *Acta Crystallographica Section D: Biological Crystallography*, 70(11), 2913–2923. <https://doi.org/10.1107/S1399004714020380>
- Piontek, K., Antorini, M., & Choinowski, T. (2002). Crystal Structure of a Laccase from the Fungus *Trametes versicolor* at 1.90-a Resolution Containing a Full Complement of Coppers * [Publisher: Elsevier]. *Journal of Biological Chemistry*, 277(40), 37663–37669. <https://doi.org/10.1074/jbc.M204571200>
- Plastics Europe. (2024). Plastics - the fast facts 2024. <https://plasticseurope.org/knowledge-hub/plastics-the-fast-facts-2024/>
- Plastics Europe. (2025). Collection & sorting. <https://plasticseurope.org/sustainability/circularity/waste-management-prevention/collection-sorting/>
- Reiss, R., Ihssen, J., Richter, M., et al. (2013). Laccase versus laccase-like multi-copper oxidase: A comparative study of similar enzymes with diverse Substrate spectra (C. M. Soares, Ed.). *PLoS ONE*, 8(6), e65633. <https://doi.org/10.1371/journal.pone.0065633>
- Richardson, L., Allen, B., Baldi, G., et al. (2023). MGnify: The microbiome sequence data analysis resource in 2023. *Nucleic Acids Research*, 51(D1), D753–D759. <https://doi.org/10.1093/nar/gkac1080>
- Roberts, S. A., Weichsel, A., Grass, G., et al. (2002). Crystal structure and electron transfer kinetics of CueO, a multicopper oxidase required for copper homeostasis in *Escherichia coli*. *Proceedings of the National Academy of Sciences of the United States of America*, 99(5), 2766–2771. <https://doi.org/10.1073/pnas.052710499>

- Ronca, S. (2017, January 1). Chapter 10 - polyethylene. In M. Gilbert (Ed.), *Brydson's plastics materials (eighth edition)* (pp. 247–278). Butterworth-Heinemann. <https://doi.org/10.1016/B978-0-323-35824-8.00010-4>
- Sakuraba, H., Koga, K., Yoneda, K., et al. (2011). Structure of a multicopper oxidase from the hyper-thermophilic archaeon Pyrobaculum aerophilum. *Acta Crystallographica Section F Structural Biology and Crystallization Communications*, 67(7), 753–757. <https://doi.org/10.1107/S1744309111018173>
- Santo, M., Weitsman, R., & Sivan, A. (2013). The role of the copper-binding enzyme – laccase – in the biodegradation of polyethylene by the actinomycete *rhodococcus ruber*. *International Biodeterioration & Biodegradation*, 84, 204–210. <https://doi.org/10.1016/j.ibiod.2012.03.001>
- Schrödinger, L., & DeLano, W. (2020, May 20). *Pymol* (Version 2.4.0). <http://www.pymol.org/pymol>
- Sharma, N. (2025). Reaction mechanism and redox potential of laccase. In *Laccase and Polyphenol Oxidase* (pp. 111–132). Elsevier. <https://doi.org/10.1016/B978-0-443-13301-5.00005-6>
- Škerlová, J., Brynda, J., Šobotník, J., et al. (2024). Crystal structure of blue laccase BP76, a unique termite suicidal defense weapon [Publisher: Elsevier]. *Structure*, 32(10), 1581–1585.e5. <https://doi.org/10.1016/j.str.2024.07.015>
- Sodhi, A. S., Bhatia, S., & Batra, N. (2024). Laccase: Sustainable production strategies, heterologous expression and potential biotechnological applications. *International Journal of Biological Macromolecules*, 280, 135745. <https://doi.org/10.1016/j.ijbiomac.2024.135745>
- Steinegger, M., & Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets [Publisher: Nature Publishing Group]. *Nature Biotechnology*, 35(11), 1026–1028. <https://doi.org/10.1038/nbt.3988>
- Sun & Wang. (2023). Human Exposure to Microplastics and Its Associated Health Risks [Publisher: American Chemical Society]. *Environment & Health*, 1(3), 139–149. [https://doi.org/10.1021/envhealth.3c00053](https://doi.org/10.1021/ envhealth.3c00053)
- Syranidou, E., Karkanorachaki, K., Barouta, D., et al. (2023). Relationship between the Carbonyl Index (CI) and Fragmentation of Polyolefin Plastics during Aging [Publisher: American Chemical Society]. *Environmental Science & Technology*, 57(21), 8130–8138. <https://doi.org/10.1021/acs.est.3c01430>
- Taylor, A. B., Stoj, C. S., Ziegler, L., et al. (2005). The copper-iron connection in biology: Structure of the metallo-oxidase Fet3p. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43), 15459–15464. <https://doi.org/10.1073/pnas.0506227102>
- Thompson, R. C., Swan, S. H., Moore, C. J., et al. (2009). Our plastic age [Publisher: Royal Society]. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1526), 1973–1976. <https://doi.org/10.1098/rstb.2009.0054>
- Tournier, V., Duquesne, S., Guillamot, F., et al. (2023). Enzymes' power for plastics degradation [Number: 9]. *Chemical Reviews*, 123(9), 5612–5701. <https://doi.org/10.1021/acs.chemrev.2c00644>
- van Meegen, C. (2024, June). *Computational analysis of protein binding sites with binding pocket descriptors* [Master's thesis]. KULeuven.
- Wei & Zimmerman. (2017). Microbial enzymes for the recycling of recalcitrant petroleum-based plastics: How far are we? *Microbial Biotechnology*, 10(6), 1308. <https://doi.org/10.1111/1751-7915.12710>
- Wheeler, T. J., Clements, J., & Finn, R. D. (2014). Skyalign: A tool for creating informative, interactive logos representing sequence alignments and profile hidden Markov models. *BMC Bioinformatics*, 15(1), 7. <https://doi.org/10.1186/1471-2105-15-7>
- Whisnant, D. (2025). *Molecular weight averages*. Retrieved April 28, 2025, from [https://eng.libretexts.org/Bookshelves/Materials%5C%5FScience/Polymer%5C%5FChemistry%5C%5F\(Whisnant\)/05%3A%5C%5FMolecular%5C%5FWeight%5C%5FAverages](https://eng.libretexts.org/Bookshelves/Materials%5C%5FScience/Polymer%5C%5FChemistry%5C%5F(Whisnant)/05%3A%5C%5FMolecular%5C%5FWeight%5C%5FAverages)
- Xie, T., Liu, Z., & Wang, G. (2020). Structural basis for monolignol oxidation by a maize laccase [Publisher: Nature Publishing Group]. *Nature Plants*, 6(3), 231–237. <https://doi.org/10.1038/s41477-020-0595-5>

- Xu, Z., Yang, S., Xie, Y., et al. (2022). Modulating the adsorption orientation of methionine-rich laccase by tailoring the surface chemistry of single-walled carbon nanotubes. *Colloids and Surfaces B: Biointerfaces*, 217, 112660. <https://doi.org/10.1016/j.colsurfb.2022.112660>
- Yang, J., Li, W., Ng, T. B., et al. (2017). Laccases: Production, Expression Regulation, and Applications in Pharmaceutical Biodegradation. *Frontiers in Microbiology*, 8, 832. <https://doi.org/10.3389/fmicb.2017.00832>
- Yao, C., Xia, W., Dou, M., et al. (2022). Oxidative degradation of UV-irradiated polyethylene by laccase-mediator system. *Journal of Hazardous Materials*, 440, 129709. <https://doi.org/10.1016/j.jhazmat.2022.129709>
- Yoshida, S., Hiraga, K., Takehana, T., et al. (2016). A bacterium that degrades and assimilates poly(ethylene terephthalate). *Science*, 351(6278), 1196–1199. <https://doi.org/10.1126/science.aad6359>
- Youn, H.-D., Hah, Y. C., & Kang, S.-O. (1995). Role of laccase in lignin degradation by white-rot fungi. *FEMS Microbiology Letters*, 132(3), 183–188. <https://doi.org/10.1111/j.1574-6968.1995.tb07831.x>
- Zampolli, J., Orro, A., Manconi, A., et al. (2021). Transcriptomic analysis of Rhodococcus opacus R7 grown on polyethylene by RNA-seq [Publisher: Nature Publishing Group]. *Scientific Reports*, 11(1), 21311. <https://doi.org/10.1038/s41598-021-00525-x>
- Zampolli, J., Mangiagalli, M., Vezzini, D., et al. (2023). Oxidative degradation of polyethylene by two novel laccase-like multicopper oxidases from rhodococcus opacus r7. *Environmental Technology & Innovation*, 32, 103273. <https://doi.org/10.1016/j.eti.2023.103273>
- Zeming, L., Halil, A., Roshan, R., et al. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637), 1123–1130. <https://doi.org/10.1126/science.adc2574>
- Zettler, E. R., Mincer, T. J., & Amaral-Zettler, L. (2013). Life in the “Plastisphere”: Microbial Communities on Plastic Marine Debris [Publisher: American Chemical Society]. *Environmental Science & Technology*, 47(13), 7137–7146. <https://doi.org/10.1021/es401288x>
- Zhang, Y., Plesner, T. J., Ouyang, Y., et al. (2023). Computer-aided discovery of a novel thermophilic laccase for low-density polyethylene degradation. *Journal of Hazardous Materials*, 458, 131986. <https://doi.org/10.1016/j.jhazmat.2023.131986>
- Zhang, Y., Lin, Y., Gou, H., et al. (2022). Screening of Polyethylene-Degrading Bacteria from Rhizopertha Dominica and Evaluation of Its Key Enzymes Degrading Polyethylene. *Polymers*, 14(23), 5127. <https://doi.org/10.3390/polym14235127>

Appendix A

Code snippets

A.1 Commands executed

```
1 # Gathering all UniProt entries in the 'multicopper oxidase' family:  
2 curl --output data/up/up_raw.tsv.gz "https://rest.uniprot.org/uniprotkb/stream?  
    compressed=true&fields=accession%2Cid%2Clength%2Cec%2Cmass%2Clineage_ids%2  
    Cft_domain%2Clineage%2Cprotein_families%2Csequence&format=tsv&query=%28%28family  
    %3A%22multicopper+oxidase+family%22%29%29"  
3  
4 # Unzip the file:  
5 gzip -d data/up/up_raw.tsv.gz  
6  
7 # Take a quick look at the distribution of EC numbers in the fourth column:  
8 cut -f4 data/up/up_raw.tsv | sort | uniq -c  
9  
10 # Alignment of the PE-RE data set. These sequences were manually copied and saved to  
    files 'pe.fasta' and 're.fasta'  
11 # The two combined are in a file 'pe_re.fasta'  
12 # This command returns a diversified ensemble:  
13 muscle -align data/pe-re/pe_re.fasta -output out/muscle/pe_re_aln_diversified.fasta  
14 # Returned dispersion value of 0.4238.  
15  
16 # Extracting the MSA with maximum column confidence:  
17 muscle -maxcc out/muscle/pe_re_aln_diversified.fasta -output out/muscle/  
    pe_re_aln_maxcc.fasta  
18  
19 # Building HMM profile from the previous alignment:  
20 # fragthresh=0 because we are dealing with full length sequences.  
21 # First positional argument is the output file, second argument is the seed alignment  
.  
22 hmmbuild --amino --fragthresh 0 out/hmmer/pe_re.hmm out/muscle/pe_re_aln_maxcc.fasta  
23  
24 # Searching the mg files with the HMM profile:  
25 # This was done using a script that loops over each fasta file from the mg dataset  
    and executes hmmsearch:  
26 # For one file:  
27 hmmsearch --noali --tblout out/hmmer/mg1_hits/"$file_out".out -E 0.01 out/hmmer/pe_re  
    .hmm $file_in  
28 # For all files:  
29 bash search_mg.sh > out/programs/search_mg.out  
30  
31 # Creating mmseqs database object:  
32 mmseqs createdb data/pe-re/pe_re.fasta data/mmseqsDB/pe_reDB # PE-RE data set  
33 mmseqs createdb data/up/up_filtered.fasta/ data/mmseqsDB/upDB # UP data set  
34 mmseqs createdb data/mg/mg1/* data/mg/mg2/* data/mmseqsDB/mgDB # MG data set  
35  
36 # Searching the mgDB with the pe_reDB as query:  
37 mmseqs search data/mmseqsDB/pe_reDB data/mmseqsDB/mgDB out/mmseqs/mg_searchDB tmp -e  
    0.01
```

```

38 # Converting to parsable format:
39 mmseqs convertalis data/mmseqsDB/pe_reDB data/mmseqsDB/mgDB out/mmseqs/mg_searchDB
40         out/mmseqs/mg_search.out
41
42 # Clustering all candidate LMCOs on 90% seq. identity:
43 # After running the pipeline, candidates were stored in a file 'all_candis.fasta'
44 mmseqs createdb data/all_candis.fasta data/mmseqsDB/all_candisDB
45 mmseqs cluster all_candisDB all_candis_cluDB tmp --min-seq-id 0.9
46
47 # Generate readable tsv file:
48 mmseqs createtsv all_candisDB all_candisDB all_candis_cluDB all_candis_clu.tsv
49
50 # Extract fasta file with only the clustered representatives:
51 mmseqs createsubdb all_candis_cluDB all_candisDB all_candis_clu_repDB
52 mmseqs convert2fasta all_candis_clu_repDB all_candis_clu_rep.fasta
53
54 # Check how many sequences are in there:
55 cat all_candis_clu_rep.fasta | grep '>' | wc -l
56 # Returns 53
57
58 # Aligning all cluster representatives:
59 muscle -align all_candis_clu_rep.fasta -output out/muscle/all_candis_clu_rep_aln.
60         fasta
61
62 # Connecting to the MGNIFY FTP server:
63 # This is to obtain the ERZ - MGYP mappings.
64 ftp ftp.ebi.ac.uk # Make connection. Log in with anonymous
65 cd pub/databases/metagenomics/peptide_database/current_release # cd to relevant
66         directory
67 prompt off # Prevent having to confirm each download
68 mget mgy_contig_map_* mgy_seq_metadata_* # Get relevant files

```

A.2 MG data retrieval

```

1 """
2 This file is designed to interact with the MGNify database.
3 More specifically, it functions to retrieve data from the MGYS00005704 and MGYS0000
4 study.
5 """
6
7 # Data download
8 from urllib.request import urlretrieve
9
10 # Connection to MGnify API
11 from jsonapi_client import Session as APISession
12 from jsonapi_client import Modifier # allows us to query specific values in given
13         fields (e.g.: 'taxon-lineage').
14 import requests
15 from itertools import islice
16
17 # For storing and manipulating data:
18 import polars as pl
19
20 # Here we're specifying the endpoint of interest:
21 ENDPOINT = 'https://www.ebi.ac.uk/metagenomics/api/v1'
22
23 def fetchMG1():
24     # Opening a handle on the endpoint using jsonapi_client lib:
25     with APISession(ENDPOINT) as mgnify:
26         analyses = map(lambda r: r.json, mgnify.iterate('studies/MGYS00005704/
27         analyses'))
28         analyses = pl.json_normalize(analyses)
29
30     # We only want the analyses from NextSeq500, as these have deeper sequencing
31     depth:

```

```

29     analyses = analyses.filter(pl.col('attributes.instrument-model') == 'NextSeq
500')
30
31     # Now the download part, for every analysisID we get the predicted CDS:
32     for analysisId in analyses['attributes.accession']:
33         print(f'--- Processing {analysisId} ---')
34         # Now changing the endpoint to the downloads sections
35         for download in mgnify.iterate(f'analyses/{analysisId}/downloads'):
36             file_name = f'{analysisId}_predicted_CDS.fasta.gz'
37             # Some conditions concerning file format and description label:
38             if (download.description.label == 'Predicted CDS (aa)'
39                 and
40                 download.file_format.name == 'FASTA'):
41                 print(f'Downloading file for {analysisId}')
42                 urlretrieve(download.links.self.url, file_name)
43                 break
44         print(f'Finished for {analysisId}')
45
46     return True
47
48
49 def fetchMG2():
50     # Opening a handle on the endpoint using jsonapi_client lib:
51     with APISession(ENDPOINT) as mgnify:
52         analyses = map(lambda r: r.json, mgnify.iterate('studies/MGYS00005970/
analyses'))
53         analyses = pl.json_normalize(analyses)
54         print(analyses.head())
55
56         # Now the download part, for every analysisID we get the predicted CDS:
57         for analysisId in analyses['attributes.accession']:
58             print(f'--- Processing {analysisId} ---')
59             # Now changing the endpoint to the downloads sections
60             for download in mgnify.iterate(f'analyses/{analysisId}/downloads'):
61                 file_name = f'{analysisId}_predicted_CDS.fasta.gz'
62                 # Some conditions concerning file format and description label:
63                 if (download.description.label == 'Predicted CDS (aa)'
64                     and
65                     download.file_format.name == 'FASTA'):
66                     print(f'Downloading file for {analysisId}')
67                     urlretrieve(download.links.self.url, file_name)
68                     break
69         print(f'Finished for {analysisId}')
70
71     return True
72
73
74 def main():
75     """
76     Execute both functions
77     """
78     fetchMG1()
79     fetchMG2()
80
81 if __name__ == "__main__":
82     main()

```

A.3 PE-RE pairwise alignment

```

1 from itertools import combinations
2 import sys
3 from Bio import pairwise2
4 from Bio.Align import substitution_matrices
5
6 def readFasta(path_to_fasta:str) -> dict:
7     """
7     Read Fasta file and return as a dictionary of header:sequence

```

```

9
10    Input:
11        :param: path_to_fasta: path to fasta file.
12
13    Returns:
14        :rtype: dict:
15            keys: FASTA headers as a string.
16            values: sequences as a string.
17        """
18    # Check if the file extension is a valid one:
19    assert path_to_fasta.endswith(".fasta"), "This is not a FASTA file"
20
21    # Initiate empty dictionary to store FASTA headers and corresponding sequences +
22    # length:
23    result = {}
24
25    # Open the file:
26    with open(path_to_fasta, "r") as f:
27        # Iterate over each line:
28        for line in f:
29            # Remove leading and trailing white spaces:
30            line = line.strip()
31
32            # Store a new sequence id if the line starts with ">":
33            if line.startswith(">"):
34                sequence_id = line[1:].strip()
35                result[sequence_id] = "" # The value is an empty string.
36                continue
37
38            # If line does not start with ">", simply append the line to the value as
39            # a string,
40            result[sequence_id]+= line
41
42    return result
43
44 def getSeqIdentity(seq1:str, seq2:str) -> float:
45     """
46     This function takes two sequences as input, aligns them in a pairwise manner, and
47     calculates the sequence identity.
48
49     Input:
50         :param: seq1: Amino acid sequence 1
51         :param: seq2: Amino acid sequence 2
52
53     Returns:
54         :rtype: float: Sequence identity of both sequences
55     """
56
57     # Define substitution matrix:
58     matrix = substitution_matrices.load("BLOSUM62")
59
60     # Perform the alignment using biopython:
61     alignment = pairwise2.align.globalds(seq1, seq2, matrix, open=-1, extend=-0.1)
62
63     # Extract alignment length:
64     # alignment[0] is simply the first alignment produced (this can vary because of
65     # different paths in the loopback algorithm)
66     # alignment[0][4] is the fourth field of the object, containing the alignment
67     # length.
68     align_length = alignment[0][4]
69
70     # Compute identical positions in the aligned strings:
71     identical_positions = sum(a == b for a, b in zip(alignment[0][0], alignment
72     [0][1]))
73
74     # Divide amount of identical position by alignment length:
75     identity = (identical_positions / align_length)

```

```

71     return identity
72
73
74 def main():
75     # Load the alignment file:
76     path_to_fasta = sys.argv[1]
77
78     # Read as FASTA dictionary:
79     seqs = readFasta(path_to_fasta)
80
81     # Get all pairwise combinations:
82     combis = list(combinations(seqs.keys(),2))
83
84     # Loop over each combination and calculate the sequence identity:
85     for combi in combis:
86         print(f'\n{combi[0]} AND {combi[1]}:\n')
87         print(getSeqIdentity(seqs[combi[0]], seqs[combi[1]]))
88
89
90 if __name__ == "__main__":
91     main()

```

A.4 Q-stat implementation

```

1 import numpy as np
2 from collections import Counter
3 import math
4
5 def trimSequence(aa_seq:str) -> str:
6     """
7         This function takes an amino acid sequence and trims the fronts and ends equally
8         such that the sequence length is divisible by 50.
9
10    Input:
11        :param: aa_seq: amino acid sequence
12
13    Returns:
14        :rtype: str: trimmed amino acid sequence
15    """
16    if len(aa_seq) % 50 == 0:
17        return [aa_seq[i:i+50] for i in range(0, len(aa_seq),50)]
18
19    elif (len(aa_seq) % 50) % 2 == 0: # Even
20        trim_length = int((len(aa_seq) % 50) / 2)
21        trimmed_aa_seq = aa_seq[trim_length:len(aa_seq)-trim_length]
22        assert len(trimmed_aa_seq) % 50 == 0
23        return trimmed_aa_seq
24
25    else: # Uneven
26        trim_length = int(((len(aa_seq) % 50) + 1) / 2)
27        trimmed_aa_seq = aa_seq[trim_length:len(aa_seq)-trim_length+1]
28        assert len(trimmed_aa_seq) % 50 == 0
29        return trimmed_aa_seq
30
31 def convertToVector(aa_seq:str, aa:str) -> np.ndarray:
32     """
33         This function takes an aa sequence and returns a vector of zeros and ones.
34         The ones represent the location of the given amino acid in the sequence.
35
36    Input:
37        :param: aa_seq: amino acid sequence
38        :param: aa: amino acid symbol
39
40    Returns:
41        :rtype: np.ndarray: array of ones and zeros where every one represent the
42        position of the given amino acid

```

```

42 """
43
44 # Make sure only one amino acid symbol is given
45 assert len(aa) == 1, 'Please provide one amino acid symbol.'
46
47 # Convert to vector of all zeros:
48 v = np.zeros(len(aa_seq), dtype=int)
49
50 # Loop over aa sequence
51 for i in range(len(aa_seq)):
52     # If aa at index i is the given aa, convert the position in the vector of
53     # zeros to a one.
54     if aa_seq[i] == aa:
55         v[i] = int(1)
56
57 return v
58
59 def segmentVector(v:np.ndarray) -> list:
60 """
61 Chop a numpy array into segments of length 50
62
63 Input:
64     :param: v: Numpy array
65
66 Returns:
67     :rtype: list: List of numpy arrays. Each array is of length 50
68 """
69 assert len(v) >= 50, 'Vector must be at least length 50'
70 return [v[i:i+50] for i in range(0, len(v), 50)]
71
72
73 def getRandomizedVector(v:np.ndarray) -> np.ndarray:
74 """
75 Return a randomly shuffled numpy array
76
77 Input:
78     :param: v: Numpy array
79
80 Returns:
81     :rtype: np.ndarray: Randomly shuffled version of input vector
82 """
83 np.random.shuffle(v)
84 return v
85
86
87 def getQStat(aa_seq:np.ndarray) -> float:
88 """
89 This function calculates the q statistic for a given array of ones and zeros.
90 The sequences are chopped up in segments of 50, and the amount of methionines in
91 each segment is counted.
92
93 Input:
94     :param: aa_seq: Amino acid sequences represented by a numpy array of ones and
95     zeros
96
97 Returns:
98     :rtype: float: the q-statistic for the given amino acid sequence
99 """
100 # Chop the array into segments of 50:
101 segments = segmentVector(aa_seq)
102
103 # List of ones in each segment:
104 counts = [Counter(segment)[1] for segment in segments]
105
106 assert len(counts) == len(segments)
107
108 N = len(segments) # Amount of segments

```

```

107     A = sum(counts)/N # Average amount of Mets in a segment
108
109     # List of variance for each segment:
110     variances = [(count-A)**2 for count in counts]
111
112     # Return the q value:
113     return (math.sqrt(N * sum(variances)))/sum(counts)
114
115
116 def getNullDistri(aa_seq:np.ndarray, n:int=1000) -> np.ndarray:
117     """
118     Calculate q stats for n random shuffles of the input vector
119
120     Input:
121         :param: v: numpy array of ones and zeros
122         :param: n: amount of times to reshuffle the vector
123
124     Returns:
125         :rtype: np.ndarray: numpy array of q-stats for every reshuffle.
126     """
127     q_stats = [getQStat(getRandomizedVector(aa_seq)) for i in range(0,n)]
128
129     return np.array(q_stats)
130
131
132 def getPValue(aa_seq:np.ndarray, n_shuffles:int) -> float:
133     """
134     This function calculates q-statistics for the given sequence and for the same
135     sequence with randomly shuffled ones.
136     It then calculates a p-value to see if the observed q-stat is significantly
137     higher than what is expected under a random distribution.
138
139     Input:
140         :param: aa_seq: Vector of ones and zeros representing an amino acid sequence
141         :param: n_shuffles: The amount of times to randomly shuffle the methionines
142         and calculate a new q-stat
143
144     Returns:
145         :rtype: float: The proportion of q-stats from random distribution higher than
146         the observed q-stat in aa_seq
147
148     # Calculate the q stat for the observed sequence:
149     q_observed = getQStat(aa_seq)
150
151     # Get the distribution of q stats as if the ones were randomly distributed
152     q_shuffled = getNullDistri(aa_seq, n_shuffles)
153
154     # The p value is the proportion of q_shuffled values higher than the observed q
155     # stat
156     return np.mean(q_shuffled >= q_observed)
157
158 def main():
159     """
160     Example q-stat calculation for bac_za23
161
162     za23 = 'MIEQFPTAGARLSRRNFLVLAGLGAAATVAGCGNTGDTGPATTVIGPDSAAVK \
163             AAEQTRRANIGTGKTVTSSLQARPTRIDLGGVQVDTWAYNDRVPGREVRLRRG \
164             DLLRAELTNLPAESTIHWHGIALRNNDMDGVPLTQSIAIPNTPFTYEFLAPD \
165             AGTHWLHPHVGMQFDRGLYAPVIVEDPAEGGDDYDLEAVLVLDWLDGVGTGRTP \
166             DQQQLDLRQGGMPMSGMGMHDGGMSGMGMGAVTDPANPLGADTGDVEYPYYLI \
167             NGTLAADPFSVRARPGQRIRLRIINAGADTAFRIAVGGHELTVTHTDGYPVEP \
168             VTGSSLLIGMGERFDAVVTLDGVFPIVASAECKQGQGFALIRTGAGRTPEPT \
169             IRPTELDAPPITGLGLRAREEVRLGSRDPDRVHELMLGMDMSGYRWTINGATY \
170             DQHPTPLDVAEGQRVRLRFVNQTMMFHPMHLHGHTFQLVDGQGAGPRKDTLVL \
171             PNQTVEVDLDADNPQGQWLVHCHNLYHGEAGMMTTLSYTE'
172
173     # First we trim the sequence such that it can be divided into chunks of 50:

```

```

170 za23_trimmed = trimSequence(za23)
171 print(za23_trimmed)
172
173 # Then we convert it to a vector of ones and zeros, with ones representing Met,
174 # and zeros all the rest:
175 za23_vector = convertToVector(za23_trimmed, 'M')
176 print(za23_vector)
177
178 # Calculate the q-stat for the observed sequence:
179 q_stat = getQStat(za23_vector)
180 print(q_stat) # Returns 1.31
181
182 # Randomly reshuffle the ones and zeros 10,000 times and return a list of q-stats
183 # for each shuffle:
184 null_distr = getNullDistri(za23_vector, 10000)
185
186 # We can also directly calculate the p-value with 10,000 random shuffles:
187 p_value = getPValue(za23_vector, 10000)
188 print(p_value) # Returns 0.0005, but will vary every time since this a random
189 process
190
191 if __name__ == "__main__":
192     main()

```

A.5 Sliding window implementation

```

1 from collections import Counter
2
3 def getProportion(aa_seq:str, aa:str) -> float:
4     """
5         Get the proportion of a given aa in aa_seq.
6
7         Input:
8             :param: aa_seq: Amino acid sequence as a string.
9             :param: aa: Amino acid symbol.
10
11        Returns:
12            :rtype: float: Proportion of aa in aa_seq.
13        """
14    return Counter(aa_seq)[aa]/len(aa_seq)
15
16 def getEnrichedRegions(aa_seq:str, aa:str, n:float, t:float, dynamic:bool=False) ->
17     tuple:
18     """
19         This function takes an amino acid sequence as input and returns a tuple of two
20         lists.
21         The first list contains tuples of (start,end) coordinates for each enriched
22         region in the sequence.
23         The second list contains a list of strings. Each string is an enriched region in
24         the sequence
25         A region is enriched if the proportion of aa in a given window n exceeds a
26         certain threshold t.
27
28         Input:
29             :param: aa_seq: Amino acid sequence.
30             :param: aa: Amino acid symbol for which to find enriched regions.
31             :param: n: Size of the sliding window OR if dynamic is true divide this value
32                 with the aa proportion.
33             :param: t: Threshold for the regions OR if dynamic is true multiply this value
34                 with the aa proportion.
35             :param: dynamic: If this is true, threshold and window size are calculated
36                 with respect to the aa proportion.
37
38         Returns:
39             :rtype: tuple: ([list of start:end indices as tuples], [sequences as a list
40                 of strings]).
41        """
42

```

```

33
34     # Assert validity of input:
35     assert len(aa) == 1, 'Please provide one amino acid symbol.'
36
37     if dynamic == False:
38         assert (t > 0) and (t <= 1), 'Threshold must be between 0 and 1.'
39         assert (n > 1), 'Window size must be greater than 1.'
40         n = int(n)
41
42     else:
43         # If dynamic is True, set the parameters according to the proportion of the
44         # given amino acid:
45         n = int(n//getProportion(aa_seq, aa))
46         t = t*getProportion(aa_seq, aa)
47
48         # Assert validity:
49         assert (t > 0) and (t <= 1), 'Threshold must be between 0 and 1.'
50         assert (n > 1), 'Window size must be greater than 1.'
51
52     # Set window size to the length of the amino acid sequence if it exceeds it
53     if n >= len(aa_seq):
54         n = len(aa_seq)
55
56     # Initiate empty list to store indices of enriched regions:
57     indices = []
58
59     # Set current start and end indices:
60     start = None
61     end = None
62
63     # Loop over the aa_seq in a sliding window fashion:
64     for i in range(len(aa_seq)-n+1):
65         # Define the current window
66         window = aa_seq[i:i+n]
67
68         # If proportion of aa exceeds threshold:
69         if getProportion(window, aa) > t:
70             # If we are not already in an enriched region, start a new one. Otherwise
71             # extend the previous one
72             if start is None:
73                 start = i
74                 end = i+n-1
75             else:
76                 # Extend the current region:
77                 end = max(end, i+n-1)
78
79             # If window is not enriched, finalize the current region:
80             else:
81                 if start is not None:
82                     indices.append((start,end)) # Store the start and end positions
83                     # Reset start and end positions:
84                     start = None
85                     end = None
86
87             # Add the last region if it is there
88             if start is not None:
89                 indices.append((start,end))
90
91             #### At this point we have a list of tuples with each tuple containing the (start,
92             # end) coordinates of an MR region. ####
93
94             # Return None if no MetRich regions were found:
95             if len(indices) == 0:
96                 return None
97
98             # If only one was found, return it
99             elif len(indices) == 1:
100                 return (indices, [aa_seq[start:end+1] for start,end in indices])

```

```

98
99     # If more were found, we have to check if they do not overlap
100    else:
101        indices_merged = [] # Empty list to store merged start and end indices
102        indices.sort() # Sort for easier downstream analysis
103        current_start, current_end = indices[0] # Set current start and end position
104        as the first indices that were found
105        # Loop over the indices
106        for i in range(1, len(indices)):
107            # If start position of i is between start and end position of i-1, merge
108            them
109            next_start, next_end = indices[i]
110            if next_start <= current_end:
111                current_end = max(current_end, next_end)
112            else:
113                # If no overlap, append current start and end to the merged list
114                indices_merged.append((current_start, current_end))
115                # Update the current start and end:
116                current_start, current_end = next_start, next_end
117
118                # Add the last region:
119                indices_merged.append((current_start, current_end))
120
121
122
123 def main():
124     """
125     Example calculation for bac_za23:
126     """
127     za23 = 'MIEQFPTAGARLSRRNFLVLAGLGAAATVAGCGNTGDTGPATTVIGPDSAAVK \
128             AAEQTRRANIGTGKTVTSSLQARPTRIDLGGVQVDTWAYNDRVPGREVRLLRG \
129             DLLRAELTNLPAESTIHWHGIALRNNDMDGVPGLTQSIAAPNTPFTYEFLAPD \
130             AGTHWLHPHVGMQFDRGLYAPVIVEDPAEGGDYDLEAVLVLDWDLGVTGRTP \
131             DQQQLDTLRQGGMPMSGMGMHDGGMSGMGAVTDPANPLGADTDVGDVEYPYYLI \
132             NGTLAADPFSVRARPGQRIRLRIINAGADTAFRIA VGGHELTVTHTDGYPVEP \
133             VTGSSLLIGMGERFDAVVTLGDGVFPIVASAEKGKQQGFALIRTGAGRTPEPT \
134             IRPTELDAPPITGLGLRAREEVRLGSRDPDRVHEMLMLGMDMSGYRWTINGATY \
135             DQHTPLDVAEGQRVRLRFVNQTMMFHPMHLHGHTFQLVDGQGAGPRKD TTLVL \
136             PNQTVEVLDADNPQWLVHCHNLYHGEAGMMTTLSYTE'
137
138     # Static sliding window:
139     met_rich_regions_stat = getEnrichedRegions(aa_seq=za23, aa='M', n=40, t=0.15,
140     dynamic=False)
141     print(met_rich_regions_stat)
142     # Returns [(201, 262)], [
143     #DWLDGVTGRTPDQLDTRLQGGMPMSGMGMHDGGMSGMGAVTDPANPLGADTDVGDVEYPYY']
144
145     # Dynamic sliding window:
146     # Notice that n and t have different meaning here.
147     # The actual window size will be n//MP and the actual threshold will be t*MP
148     met_rich_regions_dyn = getEnrichedRegions(aa_seq=za23, aa='M', n=1.5, t=4.5,
149     dynamic=True)
150     print(met_rich_regions_dyn)
151     # Returns [(201, 262)], [
152     #DWLDGVTGRTPDQLDTRLQGGMPMSGMGMHDGGMSGMGAVTDPANPLGADTDVGDVEYPYY']
153     # Which is the same as before since the parameters have been chosen that way.
154
155
156 if __name__ == "__main__":
157     main()

```

Appendix B

ESMFold pIDDT plots

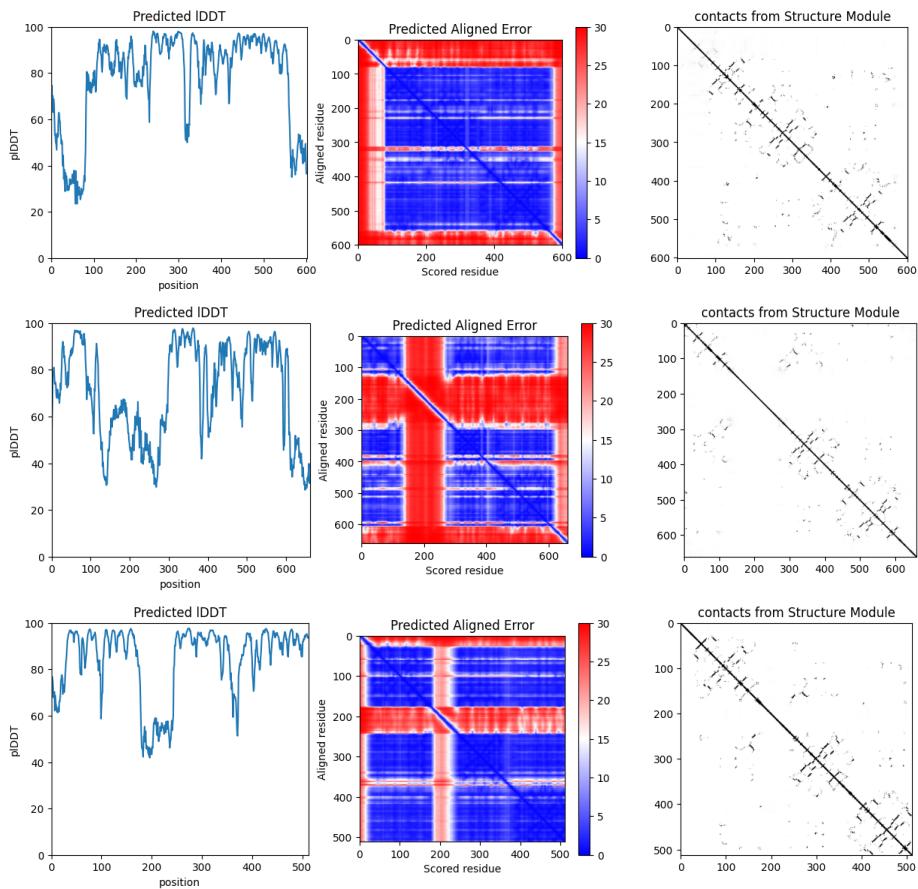


Figure B.1: pIDDT plots for the predicted structures of the three candidates. From top to bottom: *up_candi1*, *up_candi2*, *mg_candi1*

Appendix C

Additional background information

C.1 A brief history of plastics

We study the past in order to understand the present. One does not have to look too far to see that modern man is surrounded by what we now call *plastic* materials. In our kitchens, bathrooms, bedrooms and offices, plastics appear in all shapes and forms. The packaging of our foods and drinks, the clothes we wear, the casing of our computers, phones, cars and other electronic equipment, hip replacements, eye lenses, the filters in our cigarettes... are only a few examples. Eighty years ago, scientists already foresaw the "plastic age" in which we now find ourselves (Thompson et al., 2009):

"This [imaginary] plastic man will come into a world of colour and bright shining surfaces where childish hands find nothing to break, no sharp edges, or corners to cut or graze, no crevices to harbour dirt or germs... The walls of his nursery, his bath... all his toys, his cot, the moulded light perambulator in which he takes the air, the teething ring he bites, the unbreakable bottle he feeds from [all plastic]. As he grows he cleans his teeth and brushes his hair with plastic brushes, clothes himself with in plastic clothes, writes his first lesson with a plastic pen and does his lessons in a book bound with plastic. The windows of his school curtained with plastic cloth entirely grease- and dirt-proof... and the frames, like those of his house are of moulded plastic, light and easy to open never requiring any paint. [...] until at last he sinks into his grave in a hygienically enclosed plastic coffin."

(Excerpt taken from Thompson et al. (2009). The quote originates from a book called *Plastics* by Yarsley & Couzens (1945), for which I could not find the original publication. Hence, I am not quoting it directly since I could not read it.)

Obviously, the invention of contemporary plastic materials has revolutionized modern society and provides great comfort and welfare. But why, when and how did this revolution come about? And first of all, what even is a plastic?

The word 'plastic' is derived from the ancient Greek term *plastikos*, which refers to a material that can be molded or formed into a desired shape (Crawford & Quinn, 2017). Under this rather broad definition, many materials such as glass, rubber, metals... could be categorized as a plastic, since they are suitable for molding into desired shapes when using the appropriate techniques. However, when using the word 'plastic' in the 21st century, we often refer to a specific category of plastics, namely; fossil-based synthetic (man-made) polymers. The one common characteristic is that they all consist, at the molecular level, of long chains of repeated individual building blocks. One such long chain molecule (a polymer) can comprise of anywhere between 10,000 to 100,000,000 building blocks (termed monomers), which are primarily retrieved from natural gas or oil reserves buried deep beneath the Earth's crust (Crawford & Quinn, 2017). During production, many polymer molecules are synthesized, melted, and extruded, out of which the plastic emerges in its desired shape.

Nonetheless, molding and shaping different materials is an ancient phenomenon, and not solely confined to humans. We have been weaving clothes and baskets from wool and plant fiber, and

forging metals since the earliest imaginable times. Termites and ants use their surroundings to build intricate and highly complex housing structures, as do birds, bees and wasps when building their nests. Ironically enough, a recent publication in *Ecology* identified a bird's nest almost entirely made of plastic waste (Hiemstra et al., 2025). For some species, the ability to manipulate materials present in the environment has been evolutionary selected upon for millions of years. What then, precisely, is so revolutionary about the modern plastics encountered today?

The first fully synthetic polymer is often traced back to 1907, when Belgian chemist Leo Hendrik Baekeland mixed phenol and formaldehyde in a steam pressure vessel to create what he conveniently named *Bakelite*. At the time, *Bakelite*, "The Material of a Thousand Uses", was primarily used in the electrical and automotive industries for its high resistance to electricity, heat and chemical modification (American Chemical Society, 1993). Quickly, it also permeated the lives of ordinary citizens through jewelry, salt-pepper shakers, radio dials etc. Before the advent of *Bakelite*, the materials at our disposal were limited to what nature had to offer us, and their physio-chemical properties were not to be chosen by man. After *Bakelite*, mankind began to imagine and produce its own materials from scratch. Starting from a set of monomers, followed by the appropriate catalytic mechanism (mostly high pressures and heat), and finalized by the right amount of additives to improve flexibility and durability. The playground was set to start designing polymers with man-chosen properties and a near infinite amount of applications and shapes. Afterwards, a wide range of new synthetic polymers, including polyethylene (PE) in 1933, emerged.

It was, however, not until 1940-1950 that plastics began to be mass produced (Crawford & Quinn, 2017). In the context of resource shortages during the Second World War, a need for new, cheap, and more durable materials presented itself. Combined with the post-war economic growth and the upcoming of consumerism, this slowly led to the mass production and consumption of plastics that persists to this day. Additionally, equipped with this new-found ability to manufacture synthetic polymers, completely new types of products such as single-use packaging appeared on the market. Around 1960, plastic debris was observed in the guts of seabirds, and in the 1970s reports expressing the concerns of plastic in the marine environment and their uptake by animals first appeared (Thompson et al., 2009). Since then, the production of plastic has only risen, and the accumulation of these materials in the environment has been of growing concern.

C.2 Techniques to measure degradation

Mechanical alterations

Some mechanical properties such as elongation or tensile strength will have been altered after enzymatically treating a PE sample. These relate to the material's ability to resist pulling or stretching forces. Therefore, some authors will report on changes in relative elongation (RE) and relative tensile strength (RTS). These properties also correlate with the degree of crystallinity in the material (f.e. higher crystallinity will increase the tensile strength) (Ronca, 2017).

Infrared spectroscopy (FTIR or ATR-FTIR)

One of the most common techniques today relies on infrared waves to detect vibrational modes of certain functional groups such as alcohols, carbonyls, double bonds etc. As we will see later, the enzymatic degradation of PE happens through an oxidative mechanism, which makes this technique particularly suitable. More specifically, the carbonyl index (CI) is calculated to detect changes in the carbonyl band ($\text{C}=\text{O}$) relative to an internal standard, thereby indicating the abundance of this functional group. For PP films, the increase in CI after prolonged UV exposure has been shown to correlate with reductions in \bar{M}_w and \bar{M}_n , as well as increases in crystallinity (Syranidou et al., 2023). This might seem counter-intuitive at first, but since amorphous regions are more susceptible to photo-oxidation, an increase in crystallinity is the logical consequence.

In the case of unweathered PE, the IR spectrum should show a strong intensity double-peaked signal around 2800 cm^{-1} , which represents a C-H stretch (Crawford & Quinn, 2017). Additionally,

a medium-intensity peak at around 1500 cm^{-1} can also be seen. If PE is oxidized, a peak at around 1750 cm^{-1} , representing C=O bonds, should be present.

Electron microscopy

Scanning electron microscopy (SEM) is used to detect cracks and other forms of abrasion on the surface of a PE film or membrane. Since enzymes will act on the surface, it is expected to see morphological changes after treatment.

Weight loss (WL)

Researchers will often times report on reductions in weight. The simplest is weighing the sample before and after treatment. A major issue with this is that weight loss statements are rendered meaningless when the plastics contain additives (Lear et al., 2022). Especially in the case of biodegradation, micro-organisms will readily degrade additives before the polymer itself. Claiming that weight loss is proof of biodegradation when the plastic contains additives is not considered as valid evidence.

More informative, however, are techniques such as high-temperature gel permeation chromatography (HT-GPC or GPC) or size exclusion chromatography (SEC) that can separate polymers in the material based on molecular weight. These techniques allow for a determination of \bar{M}_w and \bar{M}_n of the sample, and can thus detect reductions in average molecular weight (i.e., if the chains have shortened).

Water contact angles (WCA)

In order to detect decreases in hydrophobicity, water droplets can be placed on the surface of a sample. The angle between the droplet and the surface correlates with hydrophobicity of the surface. If the angle is lower after treatment, the surface has become more polarized.

Gas chromatography - Mass spectrometry (GC-MS)

This technique is used to identify the nature of possible degradation products. After treatment, GC-MS apparatus can separate molecules based on length and mass (given that they can be ionized and gasified). The peaks in the spectra that are obtained can be compared to known retention times to determine which molecules have passed through. If depolymerization has occurred, the end products should be detectable.

Radioactively labeled PE

One of the earlier studies on PE biodegradation prepared radioactively labeled PE films and used scintillation techniques to measure the amount of radioactive CO_2 emitted from the system (PE incubated with compost or sterile soil) (Albertsson et al., 1987; Albertsson & Karlsson, 1990). After some math, this measure can be converted to a reduction in weight of the initial polymer.

This certainly seems like the most rigorous technique to detect depolymerization, since the amount of carbon transitioning from polymer to, in this case gaseous CO_2 , can be directly monitored. However, for reasons unknown, modern studies have completely deterred from using this technique. Perhaps radioactively labeled materials are unsafe to work with, or too expensive.

Bibliography

- Albertsson, A.-C., Andersson, S., & Karlsson, S. (1987). The mechanism of biodegradation of polyethylene. *Polymer Degradation and Stability*, 18(1), 73–87. [https://doi.org/10.1016/0141-3910\(87\)90084-X](https://doi.org/10.1016/0141-3910(87)90084-X)
- Albertsson & Karlsson. (1990). The influence of biotic and abiotic environments on the degradation of polyethylene. *Progress in Polymer Science*, 15(2), 177–192. [https://doi.org/10.1016/0079-6700\(90\)90027-X](https://doi.org/10.1016/0079-6700(90)90027-X)
- American Chemical Society. *Leo hendrick baekeland and the invention of bakelite*. en. <https://www.acs.org/education/whatischemistry/landmarks/bakelite.html>. 1993, November 9.
- Crawford & Quinn. (2017, February 1). *Microplastic pollutants*. Elsevier Inc.
- Hiemstra, A.-F., Gravendeel, B., & Schilthuizen, M. (2025). Birds documenting the Anthropocene: Stratigraphy of plastic in urban bird nests. *Ecology*, 106(2), e70010. <https://doi.org/10.1002/ecy.70010>
- Lear, G., Maday, S. D. M., Gambarini, V., et al. (2022). Microbial abilities to degrade global environmental plastic polymer waste are overstated [Publisher: IOP Publishing]. *Environmental Research Letters*, 17(4), 043002. <https://doi.org/10.1088/1748-9326/ac59a7>
- Ronca, S. (2017, January 1). Chapter 10 - polyethylene. In M. Gilbert (Ed.), *Brydson's plastics materials (eighth edition)* (pp. 247–278). Butterworth-Heinemann. <https://doi.org/10.1016/B978-0-323-35824-8.00010-4>
- Syranidou, E., Karkanorachaki, K., Barouta, D., et al. (2023). Relationship between the Carbonyl Index (CI) and Fragmentation of Polyolefin Plastics during Aging [Publisher: American Chemical Society]. *Environmental Science & Technology*, 57(21), 8130–8138. <https://doi.org/10.1021/acs.est.3c01430>
- Thompson, R. C., Swan, S. H., Moore, C. J., et al. (2009). Our plastic age [Publisher: Royal Society]. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1526), 1973–1976. <https://doi.org/10.1098/rstb.2009.0054>

Appendix D

GenAI statement

As expected by the KULeuven guidelines, I here aim to be as transparent as possible concerning the use of generative large language models. GenAI usage was rather limited, primarily employed when stuck on a coding problem. Before resorting to GenAI, I made sure to try and implement the code by myself first, and consult the internet (documentation, Stackoverflow, GeeksForGeeks...). If this did not solve the problem, I turned towards GenAI to see if it could figure out what's wrong in the code. On one occasion I also prompted if it could come up with alternative methods concerning the prediction of Met-rich regions, and it is from this response that I was inspired to implement a dynamic sliding window. The text was completely written by me without the use of GenAI at any point. I here include the GenAI form, and a text document that contains every conversation I had with GPT-4o (free version) in the context of this thesis. These are relatively short conversations spread out over four individual days.

Use of Generative Artificial Intelligence (GenAI) – Form to be completed**Student name:** Biltjes Miguel**Student number:** r0759769**Please indicate with "X" whether it relates to a course assignment, to the BIG-project or to the master's thesis:** This form is related to my master's thesis.Title master's thesis: *In silico* discovery of copper oxidases for the degradation of polyethylene

Promoter: prof. Vera van Noort

 This form is related to a BIG-project.

Title BIG-project: ...

Promoter: ...

 This form is related to a course assignment.

Course name: ...

Course code: ...

Please indicate with "X": I did not use GenAI tools. I did use GenAI tools. In this case specify which one (e.g. ChatGPT/GPT4/...): GPT 4o**Please indicate with "X" (possibly multiple times) in which way you were using it:**

- As a language assistant for reviewing or improving texts you wrote yourself, provided that the model does not add new content. In this case, the use of GenAI is similar to the spelling and grammar check tools we already have today, so you do not need to explicitly mention using GenAI for this.)
- As a search engine to get initial information on a topic or to make an initial search for existing research on the topic. (This way of gathering information is similar to using an ordinary search engine when working on an assignment. As a student, you are responsible for checking and verifying the absence and correctness of references. Therefore, after this initial search, look for scientific sources and conduct your own analysis of the source documents. Interpret, analyse and process the information you obtained; don't just copy-paste it. If you then write your own text based on this information, you do not have to mention you used GenAI.)
- To generate text blocks. (If you do copy-paste text blocks of GenAI output, you have to cite your GenAI sources and quote them, i.e. you clearly state that the item was created via GenAI by citation/reference.)
- To generate graphs or figures. (If you do copy-paste graphs/figures of GenAI output, you have to cite the GenAI sources and quote them, i.e. you clearly state that the item was created via GenAI by citation/reference.)
- To generate some code as part of a larger assignment. (Watch out, this can only be done if the teacher/promotor explicitly allows it.)
- Other (Contact the teacher of the course or the supervisor of the thesis or BIG project. Explain how you comply with article 84 of the examination regulations. Explain the usefulness or added value of using GenAI.)

Further important guidelines and remarks:

The faculty follows the KU Leuven policy regarding responsible use of GenAI. This form is an aid towards transparency about the use of GenAI by the student which is essential. Irresponsible and non-transparent use of GenAI can be considered an irregularity and can be sanctioned. Students who consider to use GenAI should inform themselves through the university website concerning the additional guidelines (How to correctly quote and refer to GenAI? What is (not) allowed? Tips and points of attention for responsible use):
<https://www.kuleuven.be/english/education/student/educational-tools/generative-artificial-intelligence>

09/03/2025

Asked chatGPT to help me debug a function:

"""

Je zei:

Hello, can you help figure out what is going wrong in a python function?

ChatGPT zei:

Of course! Just share the Python function you're working on, and let me know what issue you're facing. I'll help you troubleshoot and fix it!

Je zei:

I'm trying to recursively find the distances between methionine residues in an amino acid string. Currently, my function looks like this:

```
def getMetDistances(aa_seq: str, init:list):
    14
    13 # If no more methionines can be found, return the list that stores the methionine
        indices
    12 if aa_seq.find('M') == -1:
    11     print(init)
    10     return init
    9
    8 # Recursive case
    7 else:
    6     # Find the index of the first occurrence of methionine:
    5     met_index = aa_seq.find('M')
    4     print(met_index)
    3     # Append it to the list
    2     init.append(met_index)
    1     print(init)
105     # Call the function again with an updated list of methionine indices:
    1     getMetDistances(aa_seq[met_index+1:], init)
```

I tested this function with the following call:

```
print(getMetDistances('MAAMAAAAAMAAA', []))
```

It should return [0,2,5]. The print statements in the function all work and indeed '[0,2,5]' is printed to std out at the end, but the return statement results in 'None', can you figure out why?

ChatGPT zei:

I see what's happening here! The issue is that your function is using recursion, but the return value from the recursive call isn't being passed back up the call stack properly. This is why you're seeing None as the final return value.

When you call `getMetDistances(aa_seq[met_index+1:], init)` recursively, the function returns None because there's no return statement in the recursive call. You need to ensure that the recursive call returns its result and passes it up the chain.

To fix this, modify the recursive call so that it properly returns the result from the next level of recursion. Here's the updated function:

python

Kopiëren

```
def getMetDistances(aa_seq: str, init: list):
    # If no more methionines can be found, return the list that stores the methionine
    # indices
    if aa_seq.find('M') == -1:
        return init # This is the base case

    # Recursive case
    else:
        # Find the index of the first occurrence of methionine:
        met_index = aa_seq.find('M')
        # Append it to the list
        init.append(met_index)
```

```
# Call the function again with an updated list of methionine indices and return its
result

    return getMetDistances(aa_seq[met_index+1:], init)
```

```
# Test the function

print(getMetDistances('MAAMAAAAAMAAA', []))
```

Explanation:

The key change here is that we added return before the recursive call. This ensures that the value returned by the recursive call is propagated back up the call stack.

Now, when the base case (if aa_seq.find('M') == -1) is hit, the list init will be returned, and the recursive calls will keep returning until the final result is returned to the top.

Expected output:

When you run this with the input 'MAAMAAAAAMAAA', it should now correctly return:

[0, 2, 5]

Let me know if this works for you!

.....

19/03/2025

Guidance on implementing the q-stat approach. Also enquired for alternative methods to identify MetRich regions.

.....

Je zei:

Hello! Can you help me implement a statistical test in python?

ChatGPT zei:

Absolutely! I'd be happy to help you implement a statistical test in Python. What kind of statistical test are you trying to implement? For example, are you looking to perform a t-test, chi-square test, ANOVA, correlation analysis, or something else? Let me know the details, and I can guide you through it.

Je zei:

I'm looking at the distribution of methionines in amino acid sequences. The task is to find sequences where the methionines are spatially clustered together along the primary sequence. After converting the amino acid sequence to a vector of ones and zeros, where a one represents methionine, and a zero represents any other amino acid, I can calculate the q statistic using the following code:

```
def getQStat(v:np.ndarray) -> float:  
    4    """  
    3    This function calculates the q statistic for a given array of ones and zeros.  
    2    Ones represent methionine, zeros represent any other amino acid.  
    1    The sequences are chopped up in segments of 50, and the amount of mehtionines  
    in each segment is counted.
```

56

```
1    """  
2    segments = segmentVector(v)  
3  
4    counts = [] # Initialize list to store Met counts in each segment  
5  
6    for segment in segments:  
7        counts.append(Counter(segment)[1])  
8
```

```

9 assert len(counts) == len(segments)
10
11 N = len(segments) # Amount of segments
12 A = sum(counts)/N # Average amount of Mets in a segment
13
14 variances = [] # Initialize list to store variances.
15
16 # Calculate variances:
17 for i in range(len(segments)):
18     variances.append((counts[i] - A)**2)
19
20 # Return the q value:
21 return math.sqrt(N * sum(variances))/sum(counts)

```

I now want to calculate the q statistic for that same sequence, but after randomly shuffling the ones and zeros. By doing this random shuffling many times and calculating the q statistic every time, I will get a distribution of q statistics as if the methionines were randomly distributed. The idea is that, if the methionines are not randomly distributed, but rather cluster together at some segment in the sequence, the q statistic will be significantly larger than what is expected if they were randomly distributed along the sequence length. I have been able to implement the random shuffling, and am planning to repeat this procedure 1000 or 10.000 times. However, I am uncertain on how to implement the statistical test. How can I calculate a p-value for my observed sequence compared to the randomly shuffled ones? Thank you in advance for your help.

ChatGPT zei:

It sounds like you're aiming to assess the clustering of methionines in your amino acid sequences by comparing the observed q statistic to a distribution of q statistics from randomized sequences. The key here is to calculate a p-value that tells you whether the observed clustering is significantly different from what you'd expect under the assumption of random distribution.

Here's how to compute the p-value:

Calculate the observed q statistic for the original sequence.

Generate randomized sequences (shuffling the ones and zeros) and compute the q statistic for each randomized sequence.

Compare the observed q statistic to the distribution of q statistics obtained from the shuffled sequences.

Compute the p-value: The p-value is the proportion of randomized q statistics that are greater than or equal to the observed q statistic.

The steps to implement the p-value calculation could look like this:

1. Define the function to shuffle the sequence and calculate the q-statistic

You mentioned that you've already implemented random shuffling and the q-statistic calculation, so let's assume you have something like this:

```
python
```

```
Kopiëren
```

```
import numpy as np
import random
import math
from collections import Counter
```

```
# Helper function to shuffle the sequence
```

```
def shuffle_sequence(v: np.ndarray) -> np.ndarray:
    shuffled = np.copy(v)
    np.random.shuffle(shuffled)
    return shuffled
```

```
# Function to calculate q statistic
```

```
def getQStat(v: np.ndarray) -> float:
    segments = segmentVector(v) # Assuming you have this function defined
    counts = [Counter(segment)[1] for segment in segments]
```

```
assert len(counts) == len(segments)
```

```
N = len(segments)
```

```
A = sum(counts) / N # Average number of methionines in a segment
```

```
variances = [(count - A) ** 2 for count in counts]
```

```
return math.sqrt(N * sum(variances)) / sum(counts)
```

2. Generate shuffled sequences and compute q-statistics

Next, you generate multiple shuffled versions of your sequence, calculate their q-statistics, and then compare them to the observed value.

python

Kopiëren

```
def compute_p_value(observed_q: float, original_sequence: np.ndarray, n_shuffles: int = 10000) -> float:
```

```
    q_shuffled = []
```

```
# Generate random shuffles and compute q statistics
```

```
    for _ in range(n_shuffles):
```

```
        shuffled_sequence = shuffle_sequence(original_sequence)
```

```
        q_shuffled.append(getQStat(shuffled_sequence))
```

```
# Compute the p-value: fraction of q_shuffled values >= observed_q
```

```
p_value = np.mean(np.array(q_shuffled) >= observed_q)
```

```
return p_value
```

3. Putting it all together

Now, you can calculate the p-value for the observed sequence compared to the shuffled ones:

python

Kopiëren

```
# Example usage:
```

```
original_sequence = np.array([1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0]) # This is an example sequence with methionines (1's) and others (0's)
```

```
observed_q = getQStat(original_sequence) # Calculate q-statistic for the original sequence
```

```
p_value = compute_p_value(observed_q, original_sequence, n_shuffles=10000) #  
Calculate p-value  
  
print(f"Observed q-statistic: {observed_q}")  
print(f"P-value: {p_value}")
```

Explanation:

Random Shuffling: The `shuffle_sequence()` function takes the original sequence of ones and zeros, shuffles it, and returns a new sequence. This shuffling is repeated multiple times (in this case, 10,000 times).

Q-statistic Calculation: After shuffling, the `getQStat()` function calculates the q statistic for each shuffled sequence.

P-value Calculation: The p-value is the proportion of the shuffled q values that are greater than or equal to the observed q value. If the observed q is very large compared to the shuffled values, this suggests that the methionines are clustered, not randomly distributed.

Final Thoughts:

A low p-value (e.g., below 0.05) would indicate that the observed clustering of methionines is statistically significant, meaning it's unlikely to have occurred by chance.

A high p-value suggests that the observed clustering could reasonably occur under random distribution.

If you need to adjust any of the steps or have questions on any part of the code, feel free to ask!

Je zei:

Could you explain a bit more the final line in the `compute_p_value` function? Why are you using `np.mean`?

ChatGPT zei:

Absolutely! Let me explain in more detail.

The final line in the `compute_p_value` function:

python

Kopiëren

```
p_value = np.mean(np.array(q_shuffled) >= observed_q)
```

is calculating the p-value by determining how many of the shuffled q statistics are greater than or equal to the observed q statistic, and then computing the proportion of those instances.

Breaking it down:

`np.array(q_shuffled):`

This converts the list `q_shuffled` (which contains the q statistics for the shuffled sequences) into a numpy array for easier processing. While Python lists could be used, numpy arrays offer vectorized operations (which are typically faster) and convenient methods like `np.mean`.

`q_shuffled >= observed_q:`

This is a boolean comparison. For each q statistic in `q_shuffled`, it checks if that q is greater than or equal to the observed q statistic. This results in a numpy array of True and False values, where True means the shuffled q value is greater than or equal to the observed value, and False means it is not.

For example, if `q_shuffled` was [0.5, 0.7, 0.8, 1.2, 0.3] and `observed_q = 0.75`, the comparison would return: [False, True, True, True, False].

`np.mean(...):`

`np.mean` calculates the average of the array of boolean values. In Python, True is treated as 1 and False as 0, so calculating the mean of the boolean array effectively gives you the proportion of shuffled q statistics that are greater than or equal to the observed q.

For instance, with the array [False, True, True, True, False], the mean would be $(0 + 1 + 1 + 1 + 0) / 5 = 0.6$. This means that 60% of the shuffled q statistics were greater than or equal to the observed q.

Final Outcome:

The p-value represents the proportion of shuffled q statistics that are greater than or equal to the observed q statistic.

A low p-value (typically < 0.05) means that the observed clustering of methionines is unlikely to have happened by chance — the sequence's methionines are more clustered than would be expected in a random sequence.

A high p-value suggests that the observed sequence's methionine distribution is not significantly different from a random distribution.

By using np.mean on the boolean array, you're effectively counting the number of "successful" randomizations (those that have a q statistic greater than or equal to the observed value) and dividing by the total number of trials (10000 in this case).

Let me know if you'd like further clarification!

Je zei:

Thank you! That seems simpler than anticipated. The context of this question is as follows. It seems that methionine rich regions can facilitate binding to hydrophobic substances such as the plastic polyethylene, a widely used plastic. I am looking for multicopper oxidases that have hydrophobic methionine rich regions in order to identify novel candidates in the degradation of polyethylene. Currently I have two strategies to identify methionine rich regions. The one described above, and one where I use a sliding window over the amino acid sequence. If the window contains a proportion of methionine above a certain threshold, the region is considered methionine rich. I was wondering if you can come up with any alternatives to identify methionine rich regions along the primary sequence of proteins?

ChatGPT zei:

That's a fascinating and innovative line of research! Identifying methionine-rich regions that may facilitate binding to hydrophobic substances like polyethylene could definitely contribute to uncovering novel candidates for polyethylene degradation.

You've already got two good strategies in place: the q-statistic method and the sliding window approach. Both of these are effective ways to look at the distribution of methionines, but there are other approaches you can consider to capture different aspects of the amino acid sequence and to possibly identify methionine-rich regions more accurately or in a more nuanced way.

Here are a few additional strategies you could explore:

1. Cluster Analysis with Spatial Distributions

Since you're interested in methionine clustering, another method could be to identify clusters of methionines that are spatially close together. Instead of just looking for regions where methionines are present above a certain threshold, you could use spatial clustering algorithms like k-means clustering or density-based clustering (e.g., DBSCAN).

How to implement:

Convert your sequence into a binary array, where 1 represents methionine and 0 represents any other amino acid.

Use clustering techniques to find contiguous "clusters" of 1s (methionines). This could give you regions where methionines are densely packed.

For DBSCAN, you could treat methionines as points in a 1-dimensional space (the sequence), and DBSCAN will group nearby points (methionines) into clusters.

Pros: This can detect non-contiguous clusters that may be dense over short distances (i.e., clusters of methionines even if they are not all adjacent to each other).

2. Weighted Sliding Window (Dynamic Thresholding)

Instead of using a fixed threshold for the proportion of methionines in the sliding window, you can try a dynamic threshold that adjusts based on the overall methionine content of the sequence. For example, you could use a sliding window with a threshold relative to the overall methionine density in the whole sequence, or use an adaptive thresholding method that changes based on local properties of the sequence (like the frequency of methionine in local windows).

How to implement:

Compute the overall methionine density (i.e., the percentage of methionines in the entire sequence).

Set a window size (say 10-20 amino acids), and for each window, calculate the proportion of methionines.

If the proportion of methionines in that window exceeds a dynamic threshold (say 1.5x the overall methionine density), mark the region as methionine-rich.

Pros: This can help adapt to sequences where methionine distribution might vary across different parts of the protein.

3. Motif Search (Using Known Methionine-Related Sequences)

There may be known motifs or patterns of amino acids that tend to occur in methionine-rich regions, especially when methionine is involved in binding or interactions with hydrophobic substances. You could apply motif discovery algorithms (like MEME Suite or HMMER) to identify conserved motifs that may correspond to methionine-rich regions.

How to implement:

Run motif discovery tools on a large dataset of known proteins (or even just a set of proteins that are suspected to bind polyethylene).

Look for common motifs that contain high proportions of methionine.

Apply those motifs to your protein sequences to identify potential methionine-rich regions.

Pros: This approach may identify functional methionine-rich regions that are involved in specific biochemical activities like binding to hydrophobic substances.

4. Hydrophobicity and Accessibility Considerations

Since you're interested in the hydrophobic properties of methionine, you might want to take into account the hydrophobicity of the surrounding amino acids as well. Some regions with methionine may be considered "methionine-rich" because methionines are embedded in a context of hydrophobic residues, making the region more likely to be hydrophobic overall.

How to implement:

Use a hydrophobicity scale (such as the Kyte-Doolittle scale) to assign hydrophobicity values to each amino acid.

For each window or region, compute the overall hydrophobicity score.

Consider a region "methionine-rich" if it has a high methionine content and the surrounding amino acids are also hydrophobic.

Pros: This approach could highlight methionine-rich regions in hydrophobic environments, which may be more likely to interact with hydrophobic substances like polyethylene.

5. Hidden Markov Models (HMMs) for Sequence Segmentation

A Hidden Markov Model (HMM) can be used to model the underlying structure of your amino acid sequence, allowing you to identify regions of the sequence that are likely to be methionine-rich based on observed sequence patterns. HMMs are particularly useful for sequence-based tasks where you have some understanding of underlying "states" (such as methionine-rich or methionine-poor regions) and the transitions between them.

How to implement:

Use an HMM to model the sequence, with two states: methionine-rich and non-methionine-rich.

Train the HMM on sequences with known methionine-rich regions, or apply unsupervised learning on your dataset to find these regions.

Use the trained HMM to identify regions of your target sequences that are most likely methionine-rich.

Pros: This approach is particularly useful for modeling complex sequences and can capture probabilistic relationships between amino acids.

6. Topological or Structural Predictions

You could also explore predicting the secondary or tertiary structure of your proteins to look for methionine-rich regions that are likely to be exposed on the surface or involved in binding to hydrophobic surfaces. Some computational tools predict the solvent accessibility of amino acids or the likelihood of regions being involved in ligand binding.

How to implement:

Use tools like PSIPRED (for secondary structure prediction) or I-TASSER (for tertiary structure prediction) to predict the structure of your proteins.

Combine these predictions with the locations of methionine residues to identify methionine-rich regions that are likely exposed on the protein surface and could interact with hydrophobic substrates.

Pros: This method could add an extra layer of specificity, as it helps identify methionine-rich regions that are likely to be functional in interactions with hydrophobic substances.

Summary of Strategies:

Cluster Analysis (Spatial Clustering): Use clustering algorithms to identify contiguous or dense groups of methionines.

Weighted Sliding Window: Use a dynamic or relative threshold to identify methionine-rich regions based on local and global methionine densities.

Motif Search: Identify known motifs that contain high proportions of methionine using motif discovery tools.

Hydrophobicity and Accessibility: Consider the hydrophobic context of methionine-rich regions using hydrophobicity scales.

Hidden Markov Models: Use HMMs to model sequence segments and identify methionine-rich regions probabilistically.

Topological/Structural Predictions: Predict secondary or tertiary structures to find exposed methionine-rich regions that might interact with hydrophobic substances.

Each of these strategies can be complementary to your existing methods, and together they could provide a robust approach to identifying methionine-rich regions that are relevant to polyethylene degradation.

Let me know if you'd like to dive deeper into any of these methods!

.....

09/04/2025

Guidance on sliding window implementation.

”””

Je zei:

Hello, can you help me solve a coding problem in python?

ChatGPT zei:

Of course! I'd be happy to help you with your Python coding problem. Please share the details of the problem, and I'll assist you in solving it.

Je zei:

I'm implementing a method that loops over the characters of an amino acid string in a sliding window fashion. In every window, the proportion of methionine is calculated, and if it exceeds a certain threshold, the start and end coordinates of that region are stored. The method works fine, except that sometimes overlapping windows are returned, and I would like them to be merged into one window. If two consecutive windows exceed the threshold, the function makes sure that they are merged. However, if one window exceeds the threshold, the next does not, and the following one does again, these are not merged.

This is how the function looks now:

```
def getMetRichSW(aa_seq:str, n:int, t:float) -> tuple:  
    """  
        This function takes an amino acid sequence as input and returns a tuple of two lists.  
        The first list contains tuples of (start,end) coordinates for each MetRich region in the sequence.  
        The second list contains a list of strings. Each string is a MetRich region in the sequence  
        A region is MetRich if the proportion of methionine in a given window exceeds a certain threshold t.  
  
    :param aa_seq: Amino acid sequence.  
    :param n: Size of the sliding window.
```

```
12     :param: t: Threshold for methionine rich region. Should be between zero and one
13
14 Returns:
15     :rtype: tuple: (list of start:end indices as tuples, sequences as a list of strings)
16 """
17 # Assert validity of input:
18 assert n <= len(aa_seq), 'Window size cannot be larger than amino acid sequence'
19 assert (t > 0) and (t <= 1), 'Threshold must be between 0 and 1'
20
21 # Initiate empty list to store indices of Met-rich regions:
22 met_rich_indices = []
23 met_rich_sequences = []
24
25 # Set current start and end:
26 start = None
27 end = None
28
29 # Loop over the aa_seq in a sliding window fashion:
30 for i in range(len(aa_seq)-n+1):
31     # Define the current window
32     window = aa_seq[i:i+n]
33
34     # If proportion of Met exceeds threshold:
35     if getMetProportion(window) > t:
36         # If we are not already in a MR region, start a new one. Otherwise, extend the
37         # previous one
38         if start is None:
39             start = i
40             end = i+n-1
41         else:
42             # Extend the current region:
```

```
38     end = max(end, i+n-1)
37
36     # If window is not MetRich, finalize the current region:
35     else:
34         if start is not None:
33             met_rich_indices.append((start,end)) # Store the start and end positions
32             #met_rich_sequences.append(aa_seq[start:end+1])
31             # Reset start and end positions:
30             start = None
39             end = None
28
27     # Add the last region if it is there
26     if start is not None:
25         met_rich_indices.append((start,end))
24         #met_rich_sequences.append(aa_seq[start:end+1])
23
22     # At this point we have a list of tuples with each tuple containing the (start,end)
coordinates of an MR region.
21
20     # Return None if no MetRich regions were found:
19     if len(met_rich_indices) == 0:
18         return None
17
16     # If only one was found, return it
15     elif len(met_rich_sequences) == 1:
14         return (met_rich_indices, [aa_seq[start:end+1] for start,end in met_rich_indices])
13
12     # If more were found, we have to check if they do not overlap
11     else:
10         met_rich_indices_merged = []
9         for i in range(1, len(met_rich_indices)):
```

```
8      # If start position of i is between start and end position of i-1, merge them
7      if (met_rich_indices[i][0] > met_rich_indices[i-1][0]) and (met_rich_indices[i][0] <
met_rich_indices[i-1][1]):
6          met_rich_indices_merged.append((met_rich_indices[i-1][0],
met_rich_indices[i][1]))
```

The problem is thus that sometimes two or three regions such as ([200,250], [225,275]) are returned. I would then like to merge them into one region as [200,275]. As you can see, I'm trying to merge them in the last part, but how do I make sure that multiple overlapping regions are merged into one?

ChatGPT zei:

To address the issue you're encountering, you can modify the merging logic to handle multiple overlapping regions. The current approach only merges two consecutive regions, but we need to make sure that if multiple regions overlap, they are all merged into one continuous region.

Here's how you can modify your function:

Sort the regions based on their start coordinates (if they aren't already sorted).

Iterate through the regions and merge any overlapping or adjacent regions into one.

Store the merged region.

Here's how you can implement the changes:

```
def getMetRichSW(aa_seq: str, n: int, t: float) -> tuple:
```

```
    """
```

This function takes an amino acid sequence as input and returns a tuple of two lists.

The first list contains tuples of (start, end) coordinates for each MetRich region in the sequence.

The second list contains a list of strings. Each string is a MetRich region in the sequence.

A region is MetRich if the proportion of methionine in a given window exceeds a certain threshold t.

```
"""
# Assert validity of input:
assert n <= len(aa_seq), 'Window size cannot be larger than amino acid sequence'
assert (t > 0) and (t <= 1), 'Threshold must be between 0 and 1'

# Initiate empty list to store indices of Met-rich regions:
met_rich_indices = []
met_rich_sequences = []

# Set current start and end:
start = None
end = None

# Loop over the aa_seq in a sliding window fashion:
for i in range(len(aa_seq) - n + 1):
    # Define the current window
    window = aa_seq[i:i + n]

    # If proportion of Met exceeds threshold:
    if getMetProportion(window) > t:
        # If we are not already in a MR region, start a new one. Otherwise, extend the
        # previous one
        if start is None:
            start = i
            end = i + n - 1
        else:
            # Extend the current region:
            end = max(end, i + n - 1)
    else:
```

```

# If window is not MetRich, finalize the current region:
if start is not None:

    met_rich_indices.append((start, end)) # Store the start and end positions
    #met_rich_sequences.append(aa_seq[start:end + 1])
    start = None
    end = None

# Add the last region if it is there
if start is not None:

    met_rich_indices.append((start, end))
    #met_rich_sequences.append(aa_seq[start:end + 1])

# Now, we have a list of MetRich regions in `met_rich_indices` .

# If no MetRich regions were found:
if len(met_rich_indices) == 0:

    return None

# If only one region was found, return it:
elif len(met_rich_indices) == 1:

    return (met_rich_indices, [aa_seq[start:end + 1] for start, end in met_rich_indices])

# Merge overlapping regions
met_rich_indices_merged = []
met_rich_indices.sort() # Sort by start position to simplify merging logic

# Initialize the first region as the current one to merge
current_start, current_end = met_rich_indices[0]

for i in range(1, len(met_rich_indices)):

    next_start, next_end = met_rich_indices[i]

    # Check if the next region overlaps or is adjacent to the current one

```

```

if next_start <= current_end:
    # Merge the regions by extending the end of the current region
    current_end = max(current_end, next_end)
else:
    # No overlap, finalize the current region and start a new one
    met_rich_indices_merged.append((current_start, current_end))
    current_start, current_end = next_start, next_end

# Add the last merged region
met_rich_indices_merged.append((current_start, current_end))

# Return the merged indices and the corresponding sequences
return (met_rich_indices_merged, [aa_seq[start:end + 1] for start, end in
met_rich_indices_merged])

```

Key Changes:

Sorting the regions: We sort `met_rich_indices` to ensure that the regions are processed in order of their start positions. This simplifies the merging logic, as we can simply check for overlaps or adjacencies in order.

Merging the regions: We iterate over the sorted regions and check if the next region overlaps or is adjacent to the current one (`next_start <= current_end`). If they do, we merge them by updating the `current_end` to the maximum of the two ends. If they don't overlap, we finalize the current region and start a new one.

Returning merged results: The merged regions are stored in `met_rich_indices_merged`, and we return this list along with the corresponding sequences.

Example:

For input sequences like [200, 250], [225, 275], the function will merge these two regions into [200, 275] as desired.

Let me know if you need further clarification or adjustments!

16/04/2025

Help with generating code for a plot.

.....

Je zei:

Hello, can you help me make a plot with matplotlib?

ChatGPT zei:

Absolutely! I'd be happy to help you make a plot with Matplotlib. Just let me know:

What kind of plot you want to create (line, bar, scatter, histogram, etc.).

What data you're working with (you can paste the data here or describe it).

Any specific customization you want (titles, labels, colors, gridlines, etc.).

Once I have that info, I can give you the exact code.

Je zei:

I would like to make a stacked bar chart. I want one bar for the following four items: bacteria, fungi, viridiplantae and metazoa. I then want each bar to be labeled with a amino acid that occurs at a certain position in a sequence for that species. I have five labels: None, Met, Leu, Phe and Other. Each bar should thus have 5 labels, how would you do this?

ChatGPT zei:

Got it! You're aiming for a stacked bar chart where:

The x-axis represents the four species groups: bacteria, fungi, viridiplantae, and metazoa.

Each bar is stacked with segments that represent the counts (or proportions?) of five amino acid labels at a specific sequence position: None, Met, Leu, Phe, Other.

To make this chart with matplotlib, you'll need:

A 2D dataset (e.g., a dictionary or list of lists) representing counts/proportions of each amino acid per group.

A stacked bar chart using plt.bar() with the bottom= parameter to stack the bars.

Here's an example assuming you're working with counts (but we can change to percentages if needed):

Sample Data Structure:

```
data = {
    'bacteria': [10, 20, 5, 3, 2], # None, Met, Leu, Phe, Other
    'fungi': [7, 15, 10, 5, 3],
    'viridiplantae': [5, 18, 7, 6, 4],
    'metazoa': [4, 25, 8, 7, 1]
}
```

Full Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Categories
groups = ['bacteria', 'fungi', 'viridiplantae', 'metazoa']
amino_acids = ['None', 'Met', 'Leu', 'Phe', 'Other']

# Sample data: counts for each amino acid per group
data = {
    'bacteria': [10, 20, 5, 3, 2],
    'fungi': [7, 15, 10, 5, 3],
    'viridiplantae': [5, 18, 7, 6, 4],
    'metazoa': [4, 25, 8, 7, 1]
}
```

```
}
```

```
# Convert to numpy array for easier stacking
data_values = np.array([data[group] for group in groups]).T # shape: (5 amino acids, 4
groups)

# Plot setup
fig, ax = plt.subplots(figsize=(8, 6))
x = np.arange(len(groups))
bottom = np.zeros(len(groups))

# Colors (optional)
colors = ['gray', 'green', 'orange', 'purple', 'blue']

# Plot stacked bars
for idx, (aa, color) in enumerate(zip(amino_acids, colors)):
    ax.bar(x, data_values[idx], bottom=bottom, label=aa, color=color)
    bottom += data_values[idx] # Update bottom for stacking

# Labeling
ax.set_xticks(x)
ax.set_xticklabels(groups)
ax.set_ylabel('Count')
ax.set_title('Amino Acid Occurrence by Species Group')
ax.legend(title='Amino Acid')

plt.tight_layout()
plt.show()

Would you like to:
Use percentages instead of counts?
```

Customize colors or style?

Export the figure?

Add values inside the bars?

Let me know and I'll tailor it!

Je zei:

Thank you! Could you also make it such that the stacked bars are next to each other instead of in one bar? Also, how would you add percentages to the bars?

ChatGPT zei:

You're welcome! Let's tackle both of your requests:

1. Separate bars for each amino acid per group (i.e., grouped & stacked)

You want grouped and stacked bars — each group (bacteria, fungi, etc.) should have 5 bars next to each other, one for each amino acid label, and those bars are stacked (if needed). However, since you're only showing 5 labels per group, and each label only occurs once per group, it sounds more like you want grouped bars with segments showing composition, rather than stacked bars per amino acid.

But based on what you're asking — let me clarify:

If each group (e.g., "bacteria") has 1 bar, and that bar is split (stacked) into parts for None, Met, etc. → that's a stacked bar chart, which we've done already.

If you want multiple bars per group, each representing a different amino acid, then stacking inside those doesn't make sense unless there's another dimension.

But I think you're asking this:

"Keep the stacked bars, but space them apart so each group has its own bar, instead of all values being stacked into a single bar."

You already have that — each group (bacteria, fungi, etc.) is represented by one stacked bar. So you're good there!

If I misunderstood and you want side-by-side amino acid categories for each group, let me know and I'll pivot the code accordingly.

2. Add Percentages to the Bars

To show percentages inside each stacked segment, you can calculate the total per group and annotate each segment accordingly.

Here's an updated version of the previous code with percent labels inside each stacked segment:

```
import matplotlib.pyplot as plt
import numpy as np

# Data
groups = ['bacteria', 'fungi', 'viridiplantae', 'metazoa']
amino_acids = ['None', 'Met', 'Leu', 'Phe', 'Other']
data = {
    'bacteria': [10, 20, 5, 3, 2],
    'fungi': [7, 15, 10, 5, 3],
    'viridiplantae':[5, 18, 7, 6, 4],
    'metazoa': [4, 25, 8, 7, 1]
}

# Convert to numpy for processing
data_values = np.array([data[group] for group in groups]).T # shape: (5 amino acids, 4 groups)
totals = data_values.sum(axis=0)

# Plot
fig, ax = plt.subplots(figsize=(10, 6))
```

```

x = np.arange(len(groups))
bottom = np.zeros(len(groups))
colors = ['gray', 'green', 'orange', 'purple', 'blue']

# Plot stacked bars
for idx, (aa, color) in enumerate(zip(amino_acids, colors)):
    bars = ax.bar(x, data_values[idx], bottom=bottom, label=aa, color=color)

    # Add percentage text
    for bar_idx, bar in enumerate(bars):
        height = data_values[idx][bar_idx]
        if height > 0:
            percent = (height / totals[bar_idx]) * 100
            ax.text(
                bar.get_x() + bar.get_width() / 2,
                bottom[bar_idx] + height / 2,
                f'{percent:.1f}%',
                ha='center',
                va='center',
                fontsize=9,
                color='white' if percent > 10 else 'black'
            )

        bottom += data_values[idx]

# Labeling
ax.set_xticks(x)
ax.set_xticklabels(groups)
ax.set_ylabel('Count')
ax.set_title('Amino Acid Occurrence by Species Group')
ax.legend(title='Amino Acid', bbox_to_anchor=(1.05, 1), loc='upper left')

```

```
plt.tight_layout()
```

```
plt.show()
```

What You Get from This:

Each species group gets a stacked bar.

Each stacked segment shows its percentage contribution to the whole bar.

The text is dynamically placed and styled for readability.

If you want to group the amino acid bars side by side within each species group (instead of stacking), that's a different style (a grouped bar chart rather than stacked). I can whip that up too—just say the word.

Do you want that version as well?

Je zei:

Yes, I'd like the grouped bar chart as well, and just counts instead of percentages will do as well

ChatGPT zei:

Perfect! Let's build a grouped bar chart (also known as a clustered bar chart) where:

Each group (bacteria, fungi, viridiplantae, metazoa) has five bars next to each other, one per amino acid label.

We're using counts, not percentages.

This is useful for comparing the absolute occurrence of each amino acid across groups.

Here's the code:

```
import matplotlib.pyplot as plt
```

```

import numpy as np

# Data
groups = ['bacteria', 'fungi', 'viridiplantae', 'metazoa']
amino_acids = ['None', 'Met', 'Leu', 'Phe', 'Other']
data = {
    'bacteria': [10, 20, 5, 3, 2],
    'fungi': [7, 15, 10, 5, 3],
    'viridiplantae':[5, 18, 7, 6, 4],
    'metazoa': [4, 25, 8, 7, 1]
}

# Parameters
n_groups = len(groups)
n_labels = len(amino_acids)
bar_width = 0.15
x = np.arange(n_groups) # base x positions for each group

# Convert data to numpy array for easy indexing
data_values = np.array([data[group] for group in groups]) # shape: (4 groups, 5 amino acids)

# Plot
fig, ax = plt.subplots(figsize=(10, 6))
colors = ['gray', 'green', 'orange', 'purple', 'blue']

# Plot each amino acid as a separate series
for i in range(n_labels):
    offset = (i - n_labels / 2) * bar_width + bar_width / 2
    ax.bar(x + offset, data_values[:, i], width=bar_width, label=amino_acids[i],
            color=colors[i])

```

```
# Labeling  
ax.set_xticks(x)  
ax.set_xticklabels(groups)  
ax.set_ylabel('Count')  
ax.set_title('Amino Acid Occurrence by Species Group (Grouped)')  
ax.legend(title='Amino Acid', bbox_to_anchor=(1.05, 1), loc='upper left')  
  
plt.tight_layout()  
plt.show()
```

What You'll See:

Each species group (x-axis label) has a cluster of 5 bars.

Each bar in the cluster corresponds to a different amino acid.

Counts are displayed by bar height.

Let me know if you want to:

Add data labels above bars

Change the layout (horizontal, sorted, etc.)

Save/export the chart

Use your own dataset

I'm happy to tweak it however you like!

.....