Sequence Signals for Protein Expression

Benjamin Daniel Høyer

Kongens Lyngby 2012
IMM-M.Sc.-2012-128

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk
IMM-M.Sc.-2012-128

Typeset in LATEX2e using the IMM thesis template by Henrik Aalborg Nielsen, Thomas Fabricius, Jan Larsen and Finn Kuno Christensen.

Contents

Summary (English)

Optimal protein expression levels is of great interest for companies in the enzyme business. It is an in-house efficiency that takes nothing away from the customer. The task of predicting expression levels, given a nucleotide sequence, is hard. But given the cost of testing many sequences, it is well worth the effort.

This thesis documents an M.Sc. project aimed at describing the biological and machine learning background of the problem at hand. Additionally it aims to train a predictive model with a prediction correlation exceeding what is currently considered state of the art.

Making this thesis has also meant many useful R functions have been written. These are useful in themselves, but collecting them to a package will enhance portability and ease of use.

Summary (Danish)

Optimalt proteinudtryk har stor interesse for enzymvirksomheder. Det er en intern effektivitet der ikke fjerner noget fra kunden. At forudsige udtryksniveauet af en given nukleotid er en svær opgave, men omkostningerne ved at teste mange sekvenser taget i betragtning er det besværligheden værd.

Dette speciale dokumenterer et kandidatspeciale som stiler efter at beskrive den biologiske og machine learningsbaggrund for ovenstående problem. Ydermere søges at lave en forudsigelsesmodel med en forudsigelseskorrelation der overgår hvad der betragtes som det bedste.

Udarbejdelsen af dette speciale har medført en række nyttige R funktioner. De er brugbare i sig selv, men ved at samle dem i en pakke fremmer portabilitet og anvendelsen.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Mathematical Modelling and Computation. It was carried out in collaboration with Novozymes A/S.

The thesis deals with codon optimisation as a method for increasing protein yields in an industrial setting. The goal is to provide an accurate and sufficient description of DNA to protein translation for a mathematician to get a grasp of the topic. A parallel goal is to provide some insight into machine learning techniques in biotechnology for biologists. Finally, the goal is to develop a model from a machine learning standard, rather than necessarily a biological standpoint.

This thesis is loosely divided into three parts.

Part 1 Introduction and biological background. Topics from DNA transcription to RNA translation to proteins are covered.
Part 2 Prior works and machine learning methodology.
Part 3 Modelling building and validation.

Lyngby, 10th October 2012

Benjamin Daniel Høyer

Acknowledgements

I would like to thank my supervisors, Ole Winther from DTU and Thomas Poulsen from Novozymes A/S. They helped me frame and formulate the idea for this thesis. Particularly learning the biological background has been a big hurdle, so I am grateful for my time at Novozymes.

I would also like to that Joshua Plotkin and Peter Bjarke Olsen for supplying me with data. Without data, there really would have been no project at all.

No less worthy of thanks are Linzi, Rune and of course Anna, for technical help (biology), technical help (R) and all-round support, respectively.

Finally I would like to thank my parents and family for all the support they have provided me with over the years.

CHAPTER 1

Introduction

1.1 Setting

In today's energy-concious world, companies and consumers alike are seeking ways to diminish their energy requirements. They also seek to reduce their usage of harsh or expensive chemicals, both in the home and in all elements of the chain of production of day-to-day goods. A class of biological molecules called enzymes have proven to be advantageous towards both of these goals.

This goal — to be energy efficient — is very much in line with nature's own desires: in a sense, it is at the core of evolution by natural selection. Energy is nature's underlying currency, and spending as little of it as possible increases fitness.

Evolutionary pressure for energy efficiency means that nature has had a several-million-year head-start on us. Inside all cells, enzymes allow reaction rates to be sped up millions of times without resorting to high temperature or damaging chemicals. Given the range of organisms in the world found in every thinkable environment there seem to be enzymes out there for every given task.

Humans have been putting enzymes to work since perhaps as early as 9500 BC, most probably without knowing. In ancient Egypt and Mesopotamia cereal crops have been farmed since around this time. Archaeologists consider the most likely reason for this to be production of beer and bread. And the production of these is heavily reliant on enzymes at every stage.

Enzymes allow reactions to take place thousands of times quicker than they would otherwise, and at much lower temperatures. They are essentially to us physically and biologically, but also to our modern lifestyles. From washing powder to orange juice to animal feed, they have become an irreplaceable part of our lives.

Novozymes' business consists of discovering useful enzymes in nature and bringing them to market by manufacturing them efficiently on an industrial scale. The universal nature of the genetic code is a critical feature on which enzyme manufacturers rely heavily. It allows them to grow an enzyme found perhaps in a mould growing under the sea ice, but without having to nurture the original mould. Given that enzymes are discovered in a wide range of organisms, it is convenient that you do not need to grow each separate individual host organism in its preferred natural environment. Thankfully, you can extract the relevant DNA code, the gene, and splice it into a well-behaved and well-understood organism, such as a yeast or bacteria. This means you can focus on growing a single type of organism and optimise your industrial operation to suit it.

The process sounds simple ... at least conceptually. The biological language of genetics can be thought of as Lego pieces [1]. If you want to build copies of a Lego ship you found 'in nature', simply note down which pieces it uses and where they are. Then you can go home and recreate an identical copy of the Lego ship. The same is true of enzymes. You can probe the structure of an enzyme and discover which building blocks it is made of. In enzymes, these blocks are called amino acids. There is even the added bonus that enzymes have a linear, one-dimensional structure (admittedly embedded in three dimensions) — so all you need to note down is the order of the blocks. In genetics, the 'words' you use to describe the order of amino acids are called codons.

What complicates the issue is that there are 'accents' in the DNA code. To use the Lego analogy one last time, the blocks come in different colours. Structurally they perhaps the same role and can combine to create the same finished object. But what happens if you need to find a yellow 2×4 block and you have a big pile of red ones — with a single yellow one hidden in the middle somewhere? You will of course find it, but it will take you longer to create the final product.

The same problem is true for enzymes. The building blocks can be written in (up to six) difference ways, different codons referring to the same amino acid. Structurally this is not significant; the amino acids are after all the same. But if it is not matched to the host organism, you may end up with not very much enzyme at all.

But the rewards are high. Some experiments report a 250-fold variation between the highest protein yields and the lowest [2], entirely derived from re-writing of the gene. The efficiencies are entirely 'in-house'. That is to say that the end product is identical in all cases. The consumer sees no difference but the producer is able to produce more enzyme for a smaller price.

Note: Enzymes are a subclass of protein. The words are used somewhat interchangeably in this thesis, but generally protein will be used.

1.2 Objective

Being an engineer, I am inclined to accept the techniques and know-how that has been built-up (by others) in genetics and biotechnology. Genes are transferable between organisms and their meaning is conserved across all forms of life (almost, exceptions include some members of the *Candida* genus [3]). This is a useful property and it is my duty to extort as much benefit from it as possible, without worrying too much about how and why the biology works.

Since selecting 'good' ways to re-write genes is crucial to modern biotechnology, it is perhaps remarkable that no deterministic methodology exists. There is no proven method which will increase yield with certainty, let alone a way to predict what effect a changed codon (unchanged amino acid) will have [4]. Rewriting a gene and testing it will in some cases result in increased yield, but in the rest of cases, the yield drops. Since only positive results are reported [2, 5], there exists much literature claiming effective techniques. Not surprisingly, each significant positive result is followed by an article or two disproving the technique, and coming up with a new and improved way of predicting protein expression. Often devised by biologists, the ideas, models and methods are closely linked to biological phenomena and interpretations. In general, the suggestions are somewhat anecdotal: evidence exists to support and confirm hypotheses, but perhaps the evidence is of a limited type or range.

I have no advanced biological education beyond high school; a physical model is beyond my scope. Therefore we will tackle the problem of predicting protein expression levels for a given DNA sequence using statistical and machine-learning techniques. A wide variety of codon usage statistics exist [6, 7, 8, 9, 10] but they are often investigated singly. My objective is to investigate the predictive power of many statistics, some pre-existing and some that I will define, together, using statistical methods to find the most significant predictors. We will train models within datasets but also try to define global models perhaps applicable to several genes within common host organism, or even applicable across hosts.

Chapter 2

Biological Background

While this thesis is carried out in completion of a Mathematical Modelling degree, its subject matter is highly biological in nature. I have had to learn many technical intricacies about enzyme manufacture while preparing this thesis. Therefore, if terms like *codons* and *mRNA* are new to you, I suggest you read through the following section before proceeding. Note that since enzymes are a subset of proteins whose only distinguishing characteristic is that they catalyse reactions, the two terms are used interchangeably. Generally the term used will be consistent within specific context.

2.1 Why Enzymes?

Enzymes are nature's catalyst. They are a class of protein which catalyses biochemical reactions. They perform vital functions within cells and outwith them, from speeding up digestive processes to releasing energy stored in fat. Figure 2.1 shows the principle of enzymatic action. Note that individual enzymes catalyse only specific reactions and they are unchanged following use, which constitutes a large part of their attraction. This means very little is needed (low concentration) and this does not need regular replenishment. Enzymes are seen as an environmentally friendly way to reduce energy costs and as a replacement

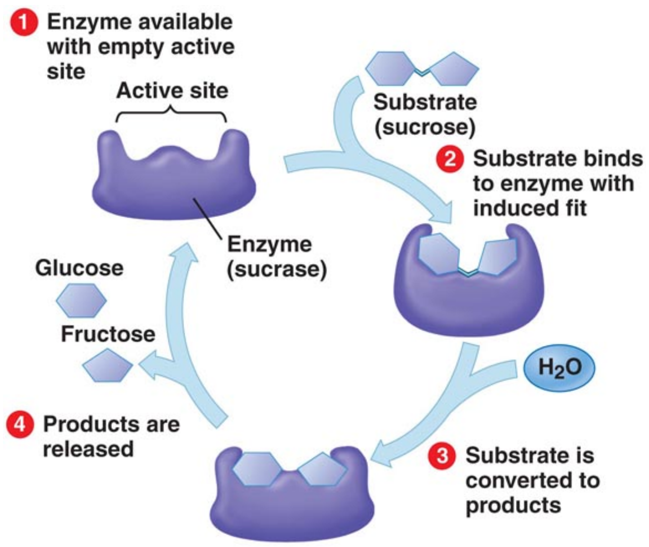

1 Enzyme available with empty active site
Active site
Substrate (sucrose)
2 Substrate binds to enzyme with induced fit
Enzyme (sucrase)
Glucose
Fructose
H_2O
4 Products are released
3 Substrate is converted to products

FIGURE 2.1: Schematic of enzyme action. The enzymes is sucrase which breaks sucrose (in the presence of H_2O) down to two products [11].

for hazardous chemical catalysts. They allow reactions, which might otherwise require extremes in temperature, pressure or pH, to take place in milder conditions. In terms of statistical thermodynamics, they reduce the activation energy for certain reactions, which really means they increase the probability of that reaction taking place [12]. At the macro scale, this translates to faster reaction rates — or effectively a reaction versus no reaction. Figure 2.2 shows how even a minor lowering of activation energy increases the number of particles sufficiently energised to undergo a particular reaction by a large factor. In common enzyme applications the reaction rate increase on the order of at least a million. A particularly effective enzyme, carbonic anhydrase, can increase reaction rates by ten million [12].

In some cases, they also widen the type of raw material that a manufacturing process can use. The primary example of this bioethanol production from cellulose (celulosic ethanol) rather than corn (corn ethanol). The primary advantage is being able to use a waste material rather than corn, which might otherwise be used as a food source [13].

All organisms produce and rely on enzymes. Many enzymes are shared among many organisms, such as those involved in basic energy transfers at the cellular level. Others are more specific, relating for example to digestion of specific food chemicals or dealing with unique environments.

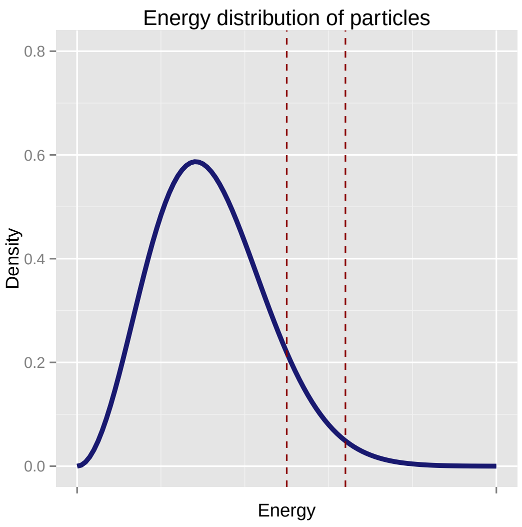


Figure 2.2: The curve shows the distribution of particles having a given energy. The area under the curve is one. The curve is a Maxwell–Boltzman distribution. The Maxwell–Boltzman distribution has the form $\sqrt{2/\pi}\, x^2 e^{-x^2/2a^2} / a^3$ (with $a = 1$) which shows the energy distribution of particles in a medium at a given temperature. An increase in temperature corresponds to moving the maximum to the right and lowering it (to maintain the unit integral). The lines represent activation energies. The line to the right represents the energy required for a reaction to take place in the absence of an enzyme. The total area under the curve to the right of this line represents the proportion of particles that will react. The line to the left represents the activation energy in the presence of an enzyme. The area under the curve to the right of this line is far larger. Hence the reaction happens far more quickly with an enzyme present.

Enzymes have an integral part to play in human history — beyond the enzymatic activities going on in all our cells. Desirable foods such as bread and beer represent probably the earliest explicit (but inadvertent) use of enzymes. There, the enzymes were housed in convenient little wrapper-organisms, yeast, which were not themselves interested in making bread or beer. But the by-products of their metabolic activities were just what we needed. Only recently (1859) have we become aware of what these bags of enzymes were, when Louis Pasteur's experiments led him to isolate and identify micro-organisms.

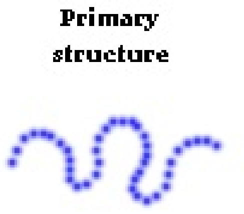
Nowadays we are able to grow yeast or bacteria that have been modified so they produce lots of a specific desirable enzyme. The enzyme can be isolated and purified so that, at the point of use, we need not worry about yeast health and nutrition; we can fully and directly control enzyme concentration with regard to the chemical process we wish to catalyse.

2.2 What Are Enzymes?

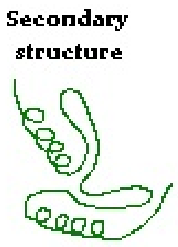
Enzymes are proteins, which means they consist of amino acids joined up in long chains by peptide bonding. The structure is linear and defined wholly by its sequence of amino acids. However, they do not exist in long, simple, rope-like configurations. They exist in minimum-energy, tangled globular or fibrous structures.

The specific structure the amino acid chain takes is critical to its proper functioning. The first stage of folding — mainly loops back onto itself — is called secondary structure and tertiary describes the fully, three-dimensional structure of the molecule. In certain cases, quaternary structures develop from multiple intertwined proteins, called protein complexes. See Fig. 2.3 for an indication of the stages of protein folding. For a given amino acid sequence, only one folding configuration represents the minimal possible energy structure. Given an amino acid sequence, the physical structure of the protein is unique, but not necessarily simple to find. Complex algorithms exist to determine the structure of a sequence which demand huge computer power [15]. Indeed, the issue of protein folding appears on Wikipedia's List of unsolved problems in chemistry [16]. Proteins are uniquely and solely identified by their linear (primary) structure. This is crucial for the methods of enzyme production in use today since it makes the problem one-dimensional rather than three-dimensional.

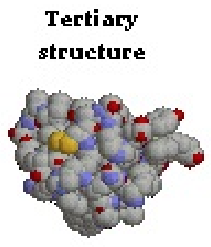
Given the linearity of enzymes, the recipe for them need only specify their amino acid order. This recipe is found in the deoxyribonucleic acid (DNA) of the cell. DNA consists of two complementary strands of nucleotide bases intertwined in



(A) Primary structure. Linear sequence of amino acids.



(B) Secondary structure, for example simple loops and helices.



(C) Tertiary structure.

FIGURE 2.3: Stages of protein folding [14].

a helix. There are four bases: A, T, G and C. They are paired so that having one half of the helix precisely identifies the other: A and T, and G and C are pair up, respectively, and are called complements. This linear structure uniquely decribes enzymes, although many stages are involved in the translation from DNA to enzyme.

2.3 Amino Acids

The building blocks which enzymes comprise are called amino acids. They are a class of molecule consisting of an amine group (-NH_2), carboxylic acid (-COOH) and a side-chain. The side-chain defines and characterises the amino acid.

Many amino acids exist (around 500 amino acids are known to occur naturally [17]), but there are just 22 'standard amino acids'. Proteins are made of these 22. 20 of these are encoded directly by DNA and the remaining two are rare and are not coded directly in DNA. The two exceptions both involve modifying the effect of a STOP codon to represent selenocysteine (existent in all kingdoms

TABLE 2.1: List of the 20 amino acids encoded in DNA.

alanine	glutamic acid	leucine	serine
arginine	glutamine	lysine	threonine
asparagine	glycine	methionine	tryptophan
aspartic acid	histidine	phenylalanine	tyrosine
cysteine	isoleucine	proline	valine

of life) [18] and pyrrolysine (in some methane-producing organisms) [19]. 30 or so artificial amino acids have been created in the lab and incorporated into proteins [20] but this is beyond the scope of this thesis. When referring to the standard amino acids in this thesis, we are referring to the 20 common amino acids. They are listed in Table 2.1.

That the 20 standard amino acids are common to all organisms is a vital characteristic which makes genetic engineering possible. It opens the door to transferring genes between organisms, and expressing certain desirable proteins in standard host organisms, regardless of where they come from.

2.4 DNA to mRNA: Transcription

Single-celled organisms can be classed broadly into two types: prokaryotes (mostly bacteria) and eukaryotes (animals, plants, fungi, ...). This thesis deals exclusively with eukaryotes; hence, for simplicity, we will limit discussion to their characteristics. We will highlight differences as necessary during the thesis.

In eukaryotes, the DNA is contained within the nucleus, along with certain other complex structures, see Fig. 2.4. The presence of a nucleus is what distinguishes them from prokaryotes. The process of actually piecing together the amino acids happens outside the nucleus, in the cytoplasm. But the DNA does not pass through the nucleus wall. Another genetic molecule, RNA — specifically mRNA (messenger ribonucleic acid) — does that.

The mRNA is generated by RNA polymerase (RNAP), a molecule which unzips the DNA molecule and produces a single-stranded complement of one side of the DNA. The bases of the DNA are transcribed to their complements as before, but a new base is introduced, uracil (U). See Table 2.2. For example the short DNA snippet CGA would be transcribed to mRNA as GCU. Since this swap from thymine to uracil is one–to–one, we lose nothing by keeping to thymine in sequence notations. From now on only the standard DNA bases will be used. In contrast to the double-stranded DNA, which takes up a very stable helical structure, mRNA is single-stranded and therefore free to form secondary and tertiary structures, like proteins. Note that not the whole DNA sequence, the genome, is transcribed. Only the section coding for the required protein (or other molecule) is copied, called a gene.

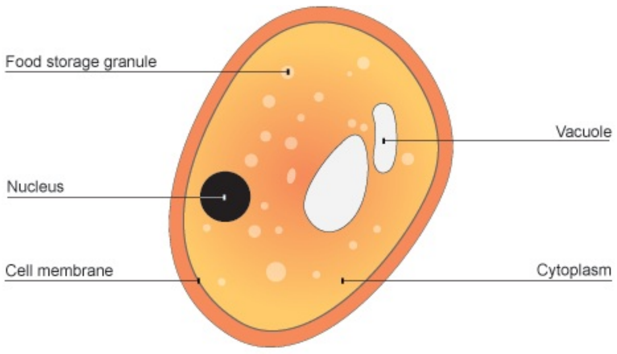


FIGURE 2.4: Diagram showing internal structure of a eukaryote (in this case, a yeast cell) [21].

TABLE 2.2: Conversion from DNA nucleotide bases to mRNA complements.

DNA base	mRNA base
C	G
A	U
T	A
G	C

2.5 mRNA

Once outside the nucleus, the mRNA is ready for translation to a chain of amino acids. A molecule called the ribosome is the key component in reading the mRNA nucleotide sequence. It is the analogue of the RNAP inside the nucleus. The part of the mRNA which actually encodes the enzyme is called the coding region. But mRNA contains several distinct regions before and after the coding region. At both ends of the molecule are special regions which signify the start and end of the molecule. At the start, there is a $5'$ (pronounced five prime) cap. This helps the ribosome know where to start. It also serves to protect the molecule from degradation. Next there is the $5'$ UTR, or the $5'$ untranslated region. This region is not translated to amino acids, but serves

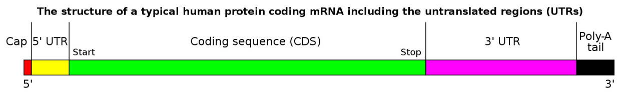

The structure of a typical human protein coding mRNA including the untranslated regions (UTRs)
Cap
5' UTR
Start
Coding sequence (CDS)
Stop
3' UTR
Poly-A tail
5'
3'

FIGURE 2.5: mRNA internal elements [22].

to promote, inhibit or otherwise regulate the translation. The coding sequence follows, which is signified by special start message: AUG. (This codon not only means START.) This is where the actual instructions are for building the enzyme. The coding sequence is stopped by a STOP message. Continuing along the mRNA we reach the 3′ UTR which contains sequences affecting, for example, the stability of the mRNA molecule. The molecule is finished with a poly(A) tail, a long series of A nucleotide bases. The poly(A) tail is important for getting the mRNA out of the nucleus, through the nuclear membrane into the cytoplasm (nuclear export) [23].

2.6 Ribosome

The ribosome is a complex of RNA and protein which catalyses the assembly of amino acids into enzymes, see Fig. 2.6. It is a relatively large molecule which envelops approximately nine nucleotide bases at a time. It travels along the mRNA from 5′ to 3′ and several ribosomes are able to travel along the same piece of mRNA simultaneously, each producing identical enzymes.

This process makes use of tRNA. tRNAs are molecules which exist in a charged or uncharged state. In their charged state (also called aminoacylated), they have a particular amino acid attached by covalent bonding at one end. At the other end is stretch of complementary bases. As the ribosome a passes along the mRNA, charged tRNAs are pulled in and attach momentarily to the three bases in the centre of the ribosome. At this point, the tRNA release their amino acid cargo, they detach from the mRNA, and disperse into the cytoplasm uncharged. Aminocylation replaces their lost amino acid. Meanwhile, the ribosome joins

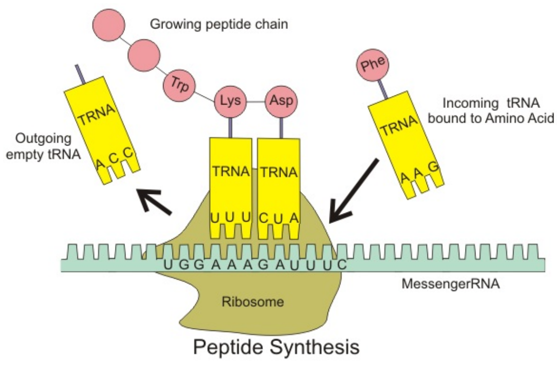


FIGURE 2.6: Diagram to show the action of the ribosome [24].

the amino acid up to form the enzyme. See Section 2.8 for more information about tRNAs.

As the enzyme becomes longer, the chain of amino acids becomes longer and extends into the cytoplasm, as process called elongation. As this happens, the chain folds into stable secondary structures.

Once the STOP signal is reached, the enzyme is complete and it detaches from the ribosome. Depending on its purpose, it may then be secreted through the cell wall into the cells external environment (secreted enzymes) or catalyse reactions within the cell (non-secreted).

2.7 Codons

It has been mentioned that the DNA code is interpreted by tRNA, via mRNA. There are four nucleotide bases in mRNA (A, U, G and C) which are read in triplets called codons. Hence there are $4^3 = 64$ possible codons. However there are only twenty amino acids, plus a STOP signal. All codons are in use. Hence, by the pigeon hole principle,[1] some amino acids necessarily have than one codon encoding. On average there are $64/21 \approx 3$ codons per amino acid (plus STOP). In reality the number of codons for each amino acid varies from one to six, methionine and tryptophan, and leucine, serine and arginine, respectively. Figure 2.7 maps out all the 64 codons of the standard genetic code and their corresponding amino acids (plus the STOP codon). To clarify to effect of multi-codon amino acids, consider the following two sequences.

ATGTCGGGGCTCTAG

ATGTCTGGATTGTGA

These sequences both have the same meaning, namely:

START · Serine · Glycine · Leucine · STOP.

An interesting feature of the degeneracy is that, often, the first two bases are sufficient to identify the amino acid. For example, the codons for proline are CCG, CCA, CCT and CCC: just the first two bases suffice. In the case of histidine and glutamine, the degeneracy is not four-fold, but the changes are still only in the final position. See Fig. 2.7.

[1]The pigeon hole principle is attributed to German mathematician Dirichlet. Given n items $I_{i=1}, \ldots, I_n$ to sort into m classes $C_{j=1}, \ldots, C_m$, where $n > m$ and all I_i must be sorted somewhere, at least one C_j must contain more than one element of I.

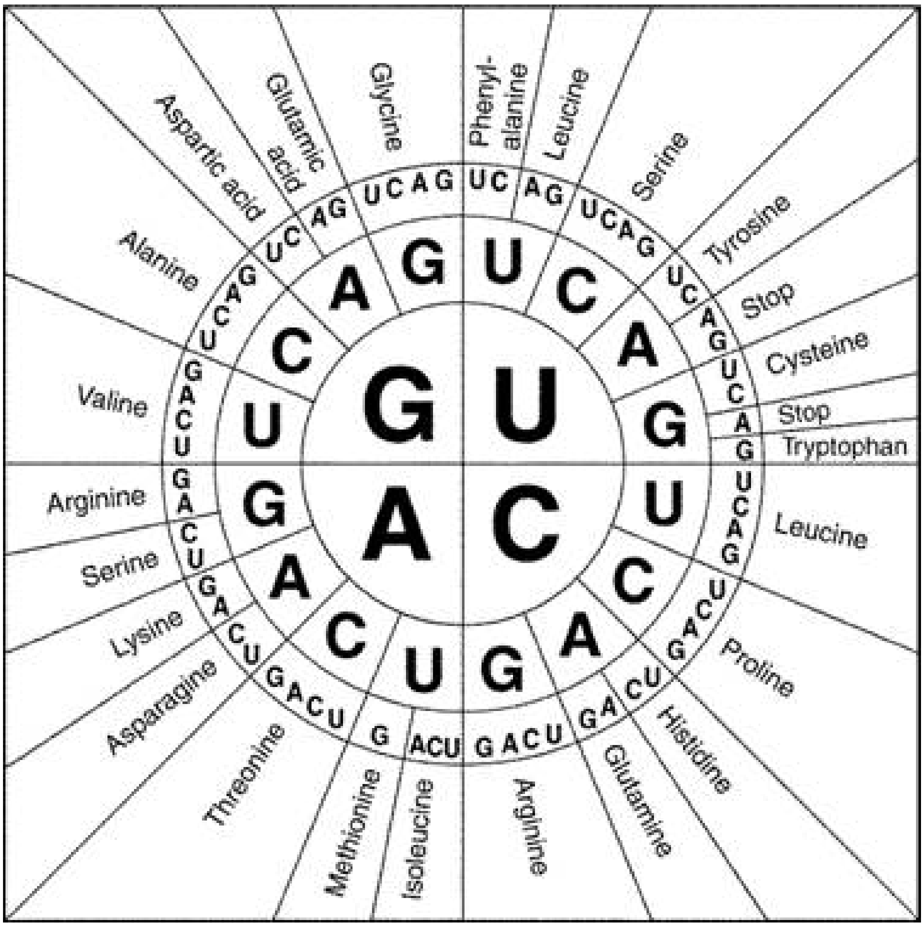


FIGURE 2.7: Codon table [25].

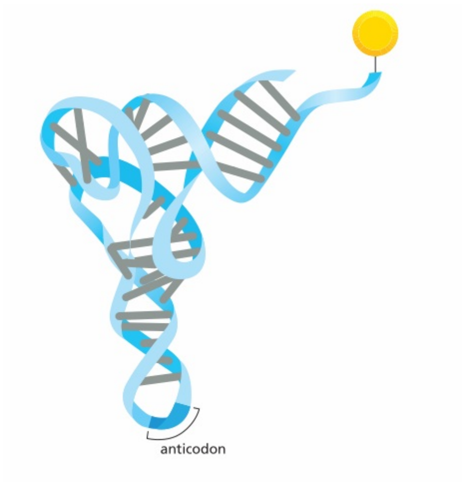
So why are there so many codons? What is clear is that reading codons in triplets is a good choice. There are 21 amino acids to code for, hence we need at least 21 possible ways combine our bases [23]. If we had only two, there would be 16 combinations, which falls short of requirements. Having some degeneracy reduces the sensitivity of the code to mutations. Mutations (mistakes) can occur at many stages of the process and are a real problem for survival. (Sometimes the effect of a mutation is positive. But this must be considered a rarity.) If we have six ways of encoding the same amino acid, we are effectively increasing the probability that a given mutation will not be deleterious to the final enzyme.

2.8 tRNA

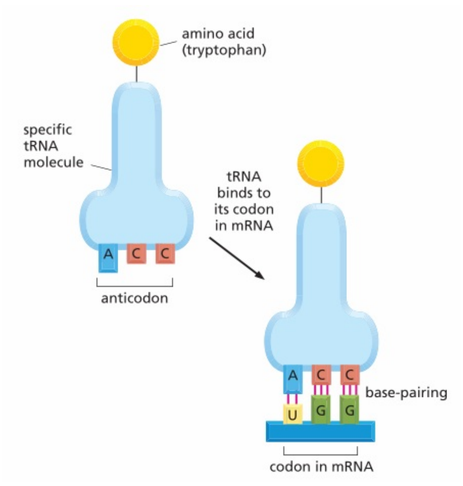
The mapping from codon to amino acid is done by transfer ribonucleic acids (tRNA). They are at the core of translation from the four-nucleotide base triplet language of DNA and RNA to the 20-amino acid language of enzymes. Put simply, they are molecules which carry a specific amino acid cargo. The cargo is unloaded inside the ribosome and attached to the growing amino acid chain, a process called elongation. (The amino acid chain, on completion, i.e. reaching a STOP codon, folds up and is then an enzyme.) Each particular tRNA carries one particular amino acid.

Naturally the amino acid sequence must reflect that encoded in the mRNA. This is achieved because the tRNAs bind onto mRNA inside the ribosome — and only one tRNA can bind onto any one codon. Hence — conceptually at least — at one end of the tRNA there is an amino acid and at the other there is a codon-reverse-complement, an anticodon. In reality, tRNA is a complex L-shaped molecule. See Fig. 2.8.

In the simplest cases, those of tryptophan and methionine, which have one codon each, the mapping from codon to tRNA to amino acid is straightforward. More often there are multiple codons which map to single amino acids. One guess might be that there is one tRNA associated with each codon, i.e. for lysine which has two codons there would be two tRNAs. They would each have a cargo of lycine, but their anticodon parts would differ: one would have TTT (corresponding to codon AAA) and the other would have TTC (corresponding to AAG). The other extreme guess would be that perhaps the tRNA is capable of binding onto both AAA and AAG, i.e. some sort of TTX anticodon, where the X is capable of binding to both A *and* G. Reality lies somewhere in between. Special bases which can bind to multiple bases exist, but also there are often more than one tRNA per amino acid.



(A) True diagram of tRNA molecule with anticodon and amino acid binding site shown.



(B) Sufficient diagram illustrating tRNA concept.

FIGURE 2.8: Diagrams of tRNA of varying detail [26, Fig. 1.9(A)].

TABLE 2.3: Table of wobbles [23, p. 744].

Last anticodon base (on tRNA)	Last codon base (on mRNA)
C	G
A	U
U	A or G
G	U or C
I	U, C or A

Often these multiply-binding codons are positioned in the third position. The amino acids which are encoded by four codons share one feature: the only part which changes is the final base. This means, for example, alanine's four codons GCA, GCC, GCG and GCT could all be translatable by a single tRNA, CGX, where X represents the 'wobble base'.

These special bases capable of binding to several distinct bases are tabulated in Table 2.3. Notice that some of the bases we already know (U and G) have this 'wobble' property. A new base, inosine (I), common in tRNA's structure, is able to bind to three bases.

Using Table 2.3 we can deduce that lysine in any form can bind to tRNA with a TTT anticodon. For alanine we require minimally two separate tRNA types:

CGI and CGC [12, 23].

Chapter 3

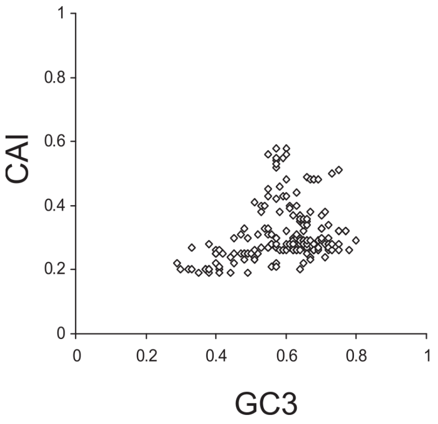
Prior Works

This thesis relies primarily on two datasets which have associated literature. The following is a summary of the nature of the datasets and original use in their respective articles. Note that some concepts will be mentioned in passing without a formal introduction or description. They will generally be covered later in the thesis.

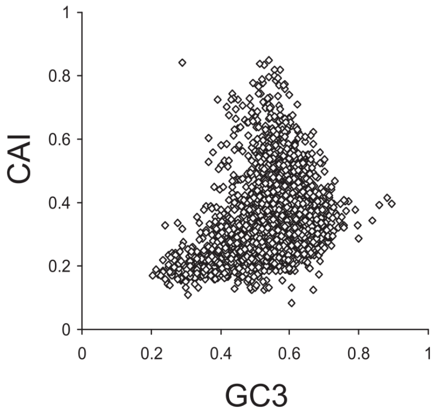
3.1 Dataset Literature

Genetic data is hard to come by. The processes involved in generating custom genetic sequences are expensive and time consuming. For this reason, many articles published on the topic of codon optimisation are analyses of already-existing data. This has the benefit that models can be compared and assessed more directly, but it runs the risk that these limit datasets may not be truly representative.

We have chosen to use two publicly available datasets for the bulk of my analysis and model development giving me a total of 210 genes. A sufficient dataset for my analysis is a list of genes and corresponding expression level. But these datasets do have some interesting features. Below is a summary of the articles from which these datasets originate.



(A) CAI versus GC3 content in the 154 sample genes.



(B) CAI versus GC3 content in 4288 real E. coli genes.

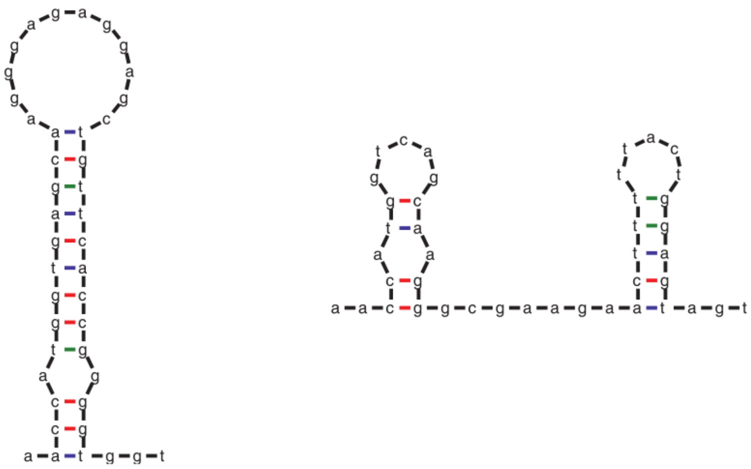
Figure 3.1: CAI versus GC3 content (G or C base in the third codon position). The 154 mutated synonymous genes represent much of the diversity present in real E. coli genes. [2, Figs. 1C and 1D]

3.1.1 Kudla et al.: Coding-Sequence Determinants of Gene Expression in Escherichia coli [2]

Description of dataset The dataset from this article consists of 148 genes all encoding GFP[1] and each with 240 codons. The codons are varied randomly and synonymously in the third base position, the wobble position. It was checked that there are no duplicates. Of the 240 codons, 226 are candidates for wobble-base mutations. (Some amino acids do not have codons that can be 'wobbled' in the final position: tryptophan only has one possible encoding, TGG.) For example, a codon encoding tyrosine can be replaced by either TAT or TAC with equal probability. The mean number of differences between two sequences was 114. The maximum difference was 180. The GC content in the third position (GC3) and CAI of the generated sequences indicate that they represent much of the diversity present in 4288 endogenous E. coli genes, see Fig. 3.1.

The cells were exposed to identical experimental conditions and consistency within replicates of identical sequences is high (reported Spearman $r = 0.98$). The host organism is E. coli. The range of protein expression varies 250-fold.

[1]GFP: green florescent protein. The protein is intracellular (non-secreted). Its expression level is measured by measuring the fluorescence of the cell.



(A) Free energy $-15.5\,\mathrm{kcal\,mol^{-1}}$ (B) Free energy $-4.5\,\mathrm{kcal\,mol^{-1}}$

FIGURE 3.2: Simple example to show folding energies vary. [2, Fig. 2B]

Summary of result While this article does investigate codon choices, its focus is rather on the influence of mRNA folding. It briefly considers other measures, such as CAI, FoP, number of rare codons (CAI < 0.1), number of consecutive rare codons and longest contiguous stretch of rare codons, but discounts theses in favour of its main premise: the minimum folding energy of the messenger RNA (mRNA) could be a predictor for protein expression.

Kudla et al. find that traditional global codon measures (i.e. calculated across the whole gene) are not significant factors when predicting protein expression. They also find that predicted global mRNA folding energy is not significant. However they find that, on carrying out a window-wise analysis, the folding energy becomes highly predictive. The window-wise GC content also becomes significant — but since this is closely related to the folding energy, they do not consider GC content interesting.

Folding energy, called free energy and measured in $\mathrm{kcal\,mol^{-1}}$, is a (usually negative) value which indicates how much energy is required to unfold the molecule. Longer sequences of looped mRNA will give a lower free energy (larger absolute value and more difficult to unfold). See Fig. 3.2. Given the the discrete and non-linear nature of the problem, with many local minima, such predicted values extremely intensive to calculate. For this reason we will not consider them in this thesis.

The particular window which they find to be significant is at the start: from codon position -4 to $+37$, relative to the start codon. They suggest therefore that the initiation of translation is significant, rather than the elongation. That is to say: it is important to what degree it is easy to get translation started, rather than how quickly or easily it reaches completion.

They proceed to suggest why this might be impacting on GFP expression levels. Since multiple ribosomes are able to be bound to a single strand of mRNA at any one time it seems reasonable that having a slower region, i.e. where the mRNA is tightly folded, will cause a build-up of ribosomes. Or alternatively, given a strand of mRNA that has a longer translation time, if the rate of ribosome attachment is constant, more ribosomes will be bound to the mRNA at any one time than another mRNA with a shorter translation time. Given that ribosomes are 'one size fits all', i.e. the same ribosome can translate any piece of mRNA, if you have many ribosomes held up in translating a single piece of mRNA, there will be fewer ribosomes available for other elongations. This will limit the expression of all (or at least some) other proteins within the cell, some of which might be crucial to cell health. Hence, having a poorly folded mRNA in your enzyme gene (which presumably is foreign to the cell and certainly not crucial to the cell's natural functioning) can negatively influence the translation of important proteins in the cell and ultimately lead to poor cell health. Poor cell health can in turn reduce the expression of the protein we are interested in, in this case GFP. Kudla et al. have thus reversed the direction of causality: rather than good codon choices causing good expression and thus high cell health, they say good codon choices should encourage high cell health, which in turn leads to good expression.

They do suggest that CAI plays a role in translation, but only in the rate of elongation (not initiation). But note that the rate of elongation does not directly affect protein expression rates (after an initial run-in period): only the initiation rate is important for this. They suggest genes that have quicker translation (due to more suited codons with respect to CAI) will sequester fewer ribosomes, leading to better cell health. Hence they propose that high CAI correlates with high cell health — which in turn supplies the potential for high expression.

The practical influence of this stance is that inserting weakly folded mRNA at the start of the gene has a positive effect on expression, even if this requires you to choose codons with low CAI values.

3.1.2 Welch et al.: Design Parameters to Control Synthetic Gene Expression in Escherichia coli [5]

Description of dataset This dataset contains genes coding for two distinct proteins: a DNA polymerase of bacillus phage Φ29 and a single chain antibody, scFv[2]. Both are expressed in E. coli. There are 32 synonymous genes for the polymerase protein and 30 for scFv. The genes were designed by back-translating from the amino acid sequence to a random codon sequence using Monte Carlo methodology. This resulted in an average pairwise identity between synonymous genes of 82 % and 79 % for the polymerase and scFv respectively. The range of protein expression ranged from undetectable to about 10 % and about 30 % of total cell protein, respectively: in general a variation of about 40-fold.

The technique of back-translating from an amino acid sequence to codon sequence is common in genetic research. It is often cheaper to sequence a protein than a genome or gene. It simply means that, given an amino acid, you select uniform-randomly from the possible codons for that amino acid.

Welch et al. use partial least squares to try to predict protein expression using codon usage frequencies as variables. Codon usage frequency is the frequency of occurrence of a particular codon relative to the frequency of the amino acid it encodes. Hence the sum of frequencies of codons encoding a given amino acid is one. The variables were pre-filtered to remove those with no variance (i.e. amino acids with only one codon representation) and STOP codons. Also, in the case of two-codon amino acids, one codon was removed. Since they must sum to one, they are directly inversely correlated.

The analysis sought to avoid over-fitting by using five-fold cross-validation. The modelling was repeated ten times and the number of components giving the lowest prediction error on the test set was noted.

Summary of result There were three sets of PLS modelling in this article: one for each protein and one for the combined dataset.

In modelling the polymerase data, six amino acids were found to be most significant. Of the six, five encode the most common amino acids in the polymerase gene. Also, five of the genes were used more often in the gene than in the genome on the whole. The model reached a cross-validation R^2 of 0.69.

[2]Single chain antibody scFv from OncoMed, Redwood City, California, USA.

For the scFv data, 9 genes had no detectable expression: this was not well-explained by any model. They chose to remove these from the model and found five components to be ideal under cross validation. The predictive power R^2 for this model was 0.68.

Combining the two datasets was done by normalising the expression within their subset to 3.00. There are a number of reasons for why expression levels might not be directly comparable between proteins, since their whole composition is potentially entirely different. The cross validation R^2 for this model was 0.65.

The method of this article will be looked at more closely in Section 6.1 where the analysis will also be applied to the dataset from Kudla.

3.2 Other Selected Literature

We now present a brief summary of a few articles that cover important issues, or explain the issues particularly clearly. They do not necessarily contain technqiues that we apply directly, but they contain ideas that have shaped some of the measures we choose to define in Chapter 5.

3.2.1 Gustaffson et al.: Engineering Genes for Predictable Protein Expression [27]

Gustafsson et al. give a gentle but thorough introduction to codon optimisation. Popular methods like CAI and tAI are reviewed, as well as more general scientific modelling methodology. The engineering hurdles of producing datasets are also mentioned. Looking toward the future, the authors believe that direct experimental design and synthesis of genomes, followed up by multivariate analysis, is the path to understanding protein sequence design principles.

3.2.2 dos Reis et al.: Solving the Riddle of Codon Usage Preferences: a Test for Translational Selection [28]

This is a seminal article in tAI codon usage analysis. The wobble-base theory for tRNA is covered. The authors discuss the parameters of the statistic and how they vary depending on the organism being studied. They also consider the problem gene length. (A shorter gene is more likely to exhibit extreme codon usage.) The usefulness of certain statistics is heavily dependent on gene length.

3.2.3 Fredrick and Ibba: How the Sequence of a Gene Can Tune Its Translation [29]

This article offers a good overview to the topic of codon optimisation. It also focusses on the variation of codon statistics along the length of the sequence. Specifically the authors talk about the translation speed along the sequence. It relates this mostly to tRNA abundance. The authors suggest that slow transcription may be an advantage that helps to ensure correct folding. Since a protein is identified solely by its folded structure (which is a result primarily is its linear structure), ensuring (or allowing the opportunity for) correct folding is an evolutionary advantage.

3.2.4 Navon and Pilpel: The Role of Codon Selection in Regulation of Translation Efficiency Deduced from Synthetic Libraries [30]

This article is considers both CAI and tAI, but focusses on the bottle-neck effect. The authors point out that two genes may have identical CAI and tAI values, but have different orders of codons which might affect translation. They rate bottleneck strength, which increases if low-CAI and -tAI codons are used. Specifically they define it as the mean of the inverse tAI value for the region in question. They also consider bottleneck location.

They analyse both the Kudla and Welch datasets. The Kudla set appears to have a strong 'bottleneck' effect in the relative region 0.16 to 0.28. The Welch set has a weaker dependence on bottleneck location.

3.3 Review of Some Existing Codon Measures

There are many ways to quantify various qualities in a genetic sequence [31]. This is evident in the large range of measures that seek to summarise sequences. There are two reasons for this.

First, the nature of the data being analysed allows it. Genetic sequences are long with many possible 'features': they can be described arbitrarily and ad nauseam. Given this potentially high dimensionality, and given that the output is relatively simple (an expression measure: one-dimensional, at least), many different measures will be able to describe a model of at least moderate predictive power.

Second, we simply do not understand the intracellular process that result in protein production, let alone know how to influence them to our advantage.

Current codon usages measures can broadly be split into two distinct types based on what the null hypothesis is. The first group holds measurements up against a random null hypothesis, i.e. that there is no underlying significance in the codon choice in the gene as a whole. The second tries to see how unusual the sequence is compared to how 'normal' genes look in, for example, that organism. Usually this is phrased as being measured against a (set of) reference gene.

3.3.1 Codon Adaptation Index

The codon adaptation index (CAI) is a widespread measure for codon usage bias. Indeed Fox [8] hails it as 'the gold standard' among bias indices. Developed by Sharp and Li in 1986 [32, 33], it essentially measures the distance from a given gene to a reference gene with respect to their within–amino-acid codon usages. This represented a departure from the existing methods, which could be split into two groups:

- various forms of counts of 'optimal codons' versus 'non-optimal codons', e.g. frequency of optimal codons (Fop) [34] and codon bias index (CBI) [35],
- distance measures relative to a random or equal-usage null-hypothesis, e.g. effective number of codons (Nc) [6] and relative codon usage bias (RCB) [36].

Precisely which reference gene is used does not seem to be considered in most articles. Does the reference gene have a strong influence on the predictive power of its dependent CAI statistics? In the R package `seqinr` [37] a reference gene is supplied (in the form of its w vector). But in the documentation, it becomes clear that questions have not been asked. One the one hand, if calculating CAI values from this reference produces good results, then there is no problem. On the other hand, it is conspicuous that no one has investigated it. The documentation says: "According to this source file, there were no reference for Escherichia coli .... It turns out that the data for Escherichia coli and Saccharomyces cerevisiae are identical to table 1 in Sharp and Li (1987) [33] ... ."

The CAI is covered in more detail in Section 5.1.1

3.3.2 tRNA Adaptation Index

The tRNA adaptation index (tAI) analogous to the CAI, but rather than measuring adaptation to a reference gene, it is measured relative to the supply of the tRNAs that are required for translation of codons to amino acids. The idea is that if there are fewer tRNAs available, the sequence will be more efficient by not using the corresponding codons. The tAI is covered in more detail in Section 5.1.3.

3.3.3 Effective Number of Codons

This measure, denoted Nc, quantifies how many codons a given gene uses [6]. Obviously a gene may use up to a maximum of 61 codons. The smallest Nc value is 20, where only one codon is used per amino acid. The principle is that highly expressed genes have highly tuned codon usages, and are likely to have a low Nc. Hence Nc can be seen as an indicator of a highly selected set of codons.

3.3.4 Frequency of Optimal Codons

This measure, Fop, is one of the earliest codon measures [34]. Optimality is defined from a combination of factors, including tRNA wobble-base pairing, tRNA abundance and nucleotide chemistry. The frequency is then simply the ratio of the counts of the optimal codon (for all amino acids) to the total number of codons.

3.3.5 Other Measures

There are many measures, and many slight variations on each of these measures. The purpose of this thesis is not to describe many existing codon measures, but rather to see whether good predictions can be gained using more neutral measures, i.e. without necessarily a biological story to defend them. Cannarozzi and Schneider provide a very good overview of codon measures [31].

Chapter 4

Machine Learning Methods

The Human Genome Project, declared complete in 2004 [38], is the pinnacle of data gathering. The project ran for 15 years and sequenced each of our approximately 20 thousand genes, identifying the 3 billion or so base pairs they comprise [39]. But this project is no longer unique. Research is under way to gather all genetic data regarding, for example: the Neandertal Genome Project to map the genes of our closest relative [40]; The Chimpanze Genome Project to map the genes of our closest living relative [41]; the Human Connectome Project to fully map pathsways and functions of our brain [42]; the Human Variome Project to better understand how certain genes lead to disease [43]; and 1000 Genomes to gain understanding for genetic variation across our species [44]. These technological projects themselves produce enormous datasets, but they also pave the way for others to perform smaller-scale analyses and data generation. The data generated by Kudla et al. [2] and Welch et al. [5] are examples of this.

4.1 General Model Definition

This thesis seeks to model protein expression levels using quantitative codon measures calculated from genes, with the aim to accurately predict protein expressions given a new gene. Linear modelling will be used for this. The basic form of a linear model is

$$y = \mathbf{X}^{\mathsf{T}}\beta, \tag{4.1}$$

where y is a vector of the N observed protein expression levels, β is a vector of the p parameter values and $\mathbf{X}$ is an $N \times p$ matrix containing N observations of the p variables. The intercept term will be assumed to be included in the observation matrix as a column of ones (and correspondingly an extra parameter at the start of β). This implies a perfect model with no prediction error. In reality, the only thing we can be sure of is $\mathbf{X}$ which contains the observations. Hence a realistic model cannot find y exactly, but only to within an error ε. The model parameter vector is also only an estimate, $\hat{\beta}$:

$$y = \mathbf{X}^{\mathsf{T}}\beta + \varepsilon \tag{4.2}$$

This can be written

$$\hat{y} = \mathbf{X}^{\mathsf{T}}\hat{\beta}, \tag{4.3}$$

where $\hat{y} = y - \varepsilon$ is the estimate of protein expression, ε is the real-life error, $\mathbf{X}$ is a matrix containing columns of parameters (and rows of observations) and $\hat{\beta}$ is a vector of estimated model parameters. The error term, i.e. the deviation of the model from real data points, must be minimised to ensure a good model, but it is assumed to be non-zero since there is some random variation in, for example, the measured protein expressions and in the measured predictors. However in general there will be true features in the model that are missed due to lack of understanding of the system. Hence the error term will contain more than just the measurement error.

The parameter vector β wholly defines the model. It describes the correlation of each column in $\mathbf{X}$ with the protein expression y. These values are estimated as $\hat{\beta}$ by training a model on some example data. The quantitative way to find an optimal $\hat{\beta}$ is to minimise the sum of the magnitude of prediction errors, $|y - \hat{y}| = |\varepsilon|$, for all N (test set) observations. In practice, the sum of ε^2 ($\varepsilon^{\mathsf{T}}\varepsilon$) is often minimised. This technique is known at least squares regression and is not only simple to conceptualise, but also gives the best possible linear unbiased estimates of β [45]. The term we seek to minimise is the residual sum of squares (RSS) for a given set of parameters $\hat{\beta}$:

$$\begin{aligned} \mathrm{RSS}(\beta) &= \sum_{i=1}^{N} \varepsilon_i^2 \\ &= \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} (y_i - x_i^{\mathsf{T}}\hat{\beta}_i)^2, \end{aligned} \tag{4.4}$$

where the sum is over all N observations. Rewriting to matrix notation, this becomes

$$\begin{aligned} \mathrm{RSS}(\beta) &= \varepsilon^{\mathsf{T}}\varepsilon && (4.5)\\ &= (y - \mathbf{X}\beta)^{\mathsf{T}}(y - \mathbf{X}\beta). && (4.6) \end{aligned}$$

$\mathrm{RSS}(\beta)$ is the quantity we need to minimise. With respect to β the function is a simple quadratic. Hence we know that a minium exists and that it can be found by differentiating with respect to β and finding the stationary point. This leads to the *normal equation* [46]:

$$\left.\frac{\partial \mathrm{RSS}}{\partial \beta}\right|_{\beta=\hat{\beta}} = \mathbf{X}^{\mathsf{T}}(y - \mathbf{X}\hat{\beta}) = 0. \tag{4.7}$$

Solving for $\hat{\beta}$ gives

$$\hat{\beta} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}y, \tag{4.8}$$

with the condition that $(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}$ exists, i.e. that $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ is non-singular. This condition demands that $\mathbf{X}^{\mathsf{T}}\mathbf{X}$ be full rank, i.e. that its columns and rows be linearly independent of one another. Inserting Eq. (4.8) into Eq. (4.3) gives

$$\hat{y} = \mathbf{H}y, \tag{4.9}$$

where

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}, \tag{4.10}$$

is known as the hat matrix (since it places the ˆ on y).

4.2 Higher/Lower Order Error Terms

A model is fitted by minimising an objective function. Often, as in this case, Eq. (4.4), the objective function sums the square of the residuals, RSS. The objective function can be defined in any way that ensures larger residuals (indication of poor fit) result in a high objective function value, so that in minimising the function we minimise the errors. Even if we restrict ourselves to objective functions of the form

$$f_{\mathrm{obj}}(\beta) = \sum_i (y_i - \hat{y}_i(\beta))^q, \tag{4.11}$$

where q is the exponent applied to the residuals, there is still a lot of variation possible. As $q \to 1$, the residual term becomes less sensitive to extreme values.

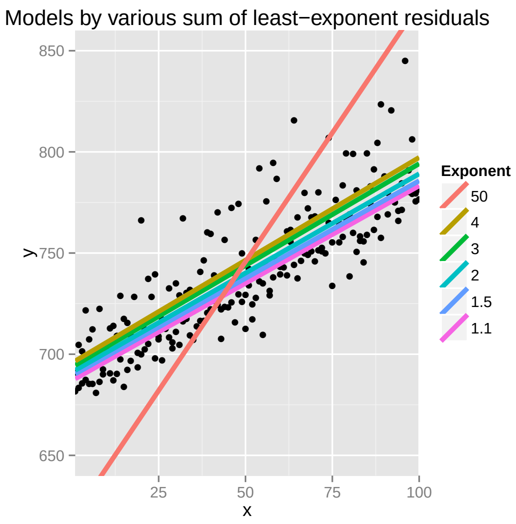


Figure 4.1: Various exponents q for residual–least–to-the-power-q models. Lower exponents try to fit *more* values *more* closely. Larger exponents focus on minimising the largest residuals.

The model will prefer to find a fit that places a line close to many points without worrying about points further away. If q becomes large, the importance of large residuals dominates. Hence the the model will minimise the largest residual, regardless of how many large residuals this results in. Figure 4.1 show the optimised linear fits for a range of residual–exponents. We see that for very large $q = 50$, the model tries to reach up to the outliers in the top right of the plot, despite this increasing the residuals for the majority of points.

Setting $q = 2$ is a remarkably good trade-off: it coincides with the *best linear unbiased estimator* by the Gauss-Markov theorem [45, Lecture 1]. But it is perhaps interesting to consider how we naturally would draw a line. Presumably we apply some sort of objective function when given the task of drawing a line through a cloud of points [47]. Restricting objective functions to the type in Eq. (4.11), carrying out a quick small-scale experiment gives the result shown in Fig. 4.2. It seems there were two kinds of people in my test: those with more care for large residuals and those who tend to prefer to match many points closely. The latter corresponds to q significantly under 2: $q \approx 1.1$. The former corresponds to something closer to $q = 4$; it seems they have over-compensated for the larger residuals, probably by trying to find the least squares solution they learnt about in statistics!

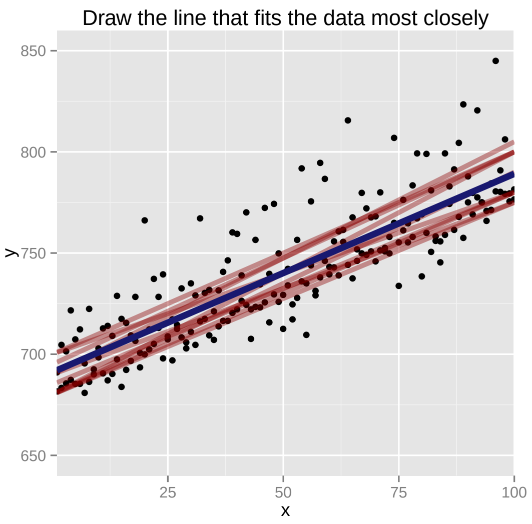


Figure 4.2: We asked a group of people (some with scientific training, some without) to draw a line through the data as best they could. Most people (by an insignificant margin, admittedly) drew a line close to an ℓ_1 error term. The blue line is a least squares fit.

4.3 Overparametrisation

In linear algebra terminology, y exists in a higher-order r space than does $\hat{y}$, $\mathcal{C}$. $\hat{y}$ is a projection of y into a p-dimensional model space $\mathcal{P}$. The difference between these is the error term $|\varepsilon|$, where the vector ε is in $r - p$ dimensional space. The dimension r of y is the *true* dimensionality of the problem at hand. Given that a model is usually a simplification of reality, p will be smaller than r. Figure 4.3 shows the relationship between y, $\hat{y}$ and ε for $r = 3$ and $p = 2$ (as best can be shown on two-dimensional paper).

Imagine that the *true* system that we are modelling is of dimension $r = 10$. That implies the true description of y requires ten linearly independent dimensions, or ten uncorrelated parameters. Given that we do not know what the system constitutes, we do not know that $r = 10$. However suppose we naïvely measure 15 quantities $x = \{x_1, \ldots, x_{p=15}\}$ that we believe to be correlated with y. Let us assume that we have captured the full dimensionality of the true system in the first ten parameters $\{x_1, \ldots, x_{10}\}$. Then the remaining five parameters must be correlated initial ten parameters. We have overparamatrised the model.

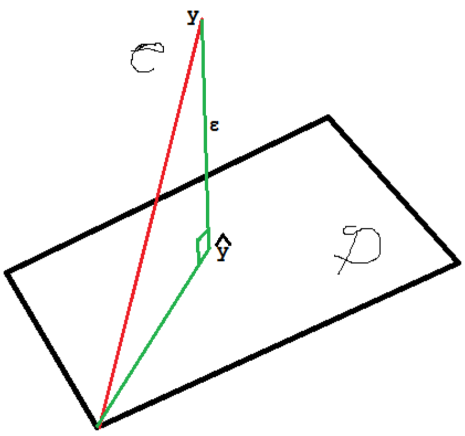


Figure 4.3: Projection of $r = 3$-dimensional y observation vector in $\mathcal{C}$ to $p = 2$-dimensional $\hat{y}$ in $\mathcal{P}$, with an $r - p = 1$-dimensional orthogonal error ε [48]. $\mathcal{P}$ is the column space of $\mathbf{X}$ and $\varepsilon \in \mathcal{C} \backslash \mathcal{P}$

In any real system there will be noise which, theoretically (if it is true noise, i.e. no structure in any dimension) cannot be described or predicted. In the situation described above, the extra five parameters could be used by the model to 'predict' the noise. (In that case, the system y would really have ten dimensions, plus noise.) Given that noise *has* no structure, these five parameters have no way of succeeding. In fact, they will decrease the predictive power of the model. This is because they will find structure that appears to be there in the limited training data which does not exist in real life, i.e. new data.

In this way, overparametrising a model can lead to fantastic fits to training data, but the model will generalise poorly and produce poor predictions on test data (it fits the noise term ε). In the extreme case when the number of observations is equal to the model size p the fit will be perfect (on training data), since each parameter in β has an entire observation devoted to it.

The predictors x can of course be correlated even if the model is not strictly speaking overparametrised. For example if we want to predict the weight of a person and we measure both the left and right feet: these measurements will be highly correlated. This is a problem since, when projecting y from $\mathcal{C}$ to $\mathcal{P}$ to get $\hat{y}$, the basis vectors for $\mathcal{P}$ — which are the columns of predictors in $\mathbf{X}$ — will not be orthogonal. This leads to high variance in the value of predictors, since the model essentially has two (almost) identical ways to identify $\hat{y}$. The model will erratically switch between highly correlated predictors.

This problem also manifests itself in the equations. The condition associated with Eq. (4.8) (that $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$ exists) requires that $\mathbf{X}^\mathsf{T}\mathbf{X}$ be invertible. The effect of correlated predictors is that $\mathbf{X}^\mathsf{T}\mathbf{X}$ becomes singular. Thus, Eq. (4.7) cannot be solved and the solution $\hat{\beta}$ is undefined.

4.4 $p > N$

Traditional statistical techniques struggle to produce useful predictive models when there are more parameters than observations. This is, in a sense, because they fit the given data too well — they fit the real-life situation (hopefully) *and* they fit the error. For example, given two observations, it is always possible to fit the data in two dimensions exactly (i.e. two points can always be joined by a line). However this model will also have taken error terms into account (i.e. the position of the two points only known to within an error term, but the model ignores this). Given that the error cannot be assumed to be well-represented by the two observations, the model will generalise poorly. The dimensionality that a model can represent is maximally equal to the number of dimensions.

The number of free parameters p in a model is related to a quantity called the number of degrees of freedom df. Since the model parameters are estimated from the observations, it is possible to quantify the number of observations used to estimate each parameter. Having more observations per parameter will result in a more stable β, since noise on the observations gets averaged out (assuming the noise is zero-mean). Given that individual observations are subject to noise, if each parameter is supported by only one observation, there will be considerable variance on the parameter, and the model will generalise poorly.

The difficulties are exacerbated when there are more variables in the model than observations available. In that case there will be infinitely many solutions that fit the test data perfectly. Again, the problem comes when trying to model a new data point. It is likely the model has captured error as a non-random term, which gives poor predictions. Think of the two observation situation again. If our model is three-dimensional (a plane) we can find any number of planes that pass exactly through the two points.

The case $p > N$ quickly arises in analysis of genetic data. The analysis of codon optimising is no exception: in its most basic form, the model developed in this thesis is trained on over 10 thousand variables, with only a total of around 200 observations. This situation is described as $p \gg N$.

4.5 Dealing With Correlated Variables and $p > N$

The problem with many parameters and correlated parameters is that the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ becomes poorly defined resulting in extremely large, unstable parameter values, or singular resulting in no solution at all.

4.5.1 The Ridge Penalty

It was this problem that Hoerl and Kennard wanted to solve [49]. To break the linear dependence of columns (correlated predictors) we add a diagonal (and therefore orthogonal) matrix $\lambda\mathbf{I}$ to $\mathbf{X}^\mathsf{T}\mathbf{X}$. This makes the columns independent and thus the inversion $(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}$ is possible. This technique is called ridge regression. Equation (4.8) then becomes

$$\hat{\beta}_{\text{ridge}} = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}y. \tag{4.12}$$

Using the ridge regression requires making a choice of λ. The effect of various choices is best seen by taking a step back to Eq. (4.4), which becomes

$$\text{RSS}(\lambda) = \overbrace{\sum_{i=1}^{N}(y_i - x_i^\mathsf{T}\hat{\beta}_{\text{ridge},i})^2}^{\text{error term}} + \overbrace{\lambda\hat{\beta}^2_{\text{ridge}}}^{\text{ridge term}}, \tag{4.13}$$

where $\beta^2 = \beta^\mathsf{T}\beta$. We see that the parameter vector $\hat{\beta}$ is now included in the objective function: specifically the inner product with itself $\sum_{j=1}^{p}\hat{\beta}_j^2$, or $\hat{\beta}^\mathsf{T}\hat{\beta}$, is included and scaled by the ridge parameter λ. As mentioned previously in the section about orders of errors, this will encourage the model to minimise the largest parameters in $\hat{\beta}$. Once parameters become small they will tend not to be minimised further.

This has an implication for correlated predictors. Suppose we have included two highly correlated predictors in a model and we use the ridge regression. The ridge terms means the parameters associated with those two predictors will both be minimised together. For example, suppose a we are interested in predicting a person's weight, y. We might think foot length is a relevant predictor. But for some reason we include the length of each foot in the model, β_l and β_r. Suppose the actual parameter associated with foot length is $\beta_{\text{foot}} = 1$. But given that we have two predictors containing the same information, they must 'share' the parameter weight, hence $\beta_l + \beta_r = 1$. Suppose the minimisation of the objective functions Eq. (4.13) starts with $\{\beta_\text{l}, \beta_\text{r}\} = \{1, 0\}$. Minimising the sum of squares will lead to the solution $\{\beta_\text{l}, \beta_\text{r}\} = \{0.5, 0.5\}$, i.e. the parameters share the weight equally amongst them.

In the general case, there can be many correlated predictors. Hence the parameter estimates for each individual predictor can end up being very small. This leads to difficulties in interpreting the parameters. For example, in the above case, we would be lead to believe that each foot is a relevant predictor of weight. Really, one foot is sufficient. Also, since parameter values can become extremely small, there can be problems of numerical stability [46].

The averaging of parameter weight happens regardless of the value of λ. But the extent to which is happens is controlled by λ. Specifically, λ controls whether the residual sum of squares term or the parameter size term dominates in Eq. (4.13). $\lambda = 0$ corresponds to a normal least–squares solution. $\lambda \to \infty$ corresponds to a solution where the prediction residuals have no importance. Only minimising the parameter values counts. This is minimal when all are zero, which is a meaningless solution.

4.5.2 The Lasso Penalty [50]

Changing the exponent of the β term in the objective function changes the behaviour of the model. Consider Fig. 4.1 on page 32. There, a lower exponent q means the line moves closer to many points. This corresponds to the line looking for a place where many residuals are zero, without worrying about a few individually large residuals. In the ridge regression, the penalty form means that large residuals are avoided and many small (non–zero) residuals are preferred.

Analogously, changing the parameter penalty to an L_1–norm, i.e. $\lambda \sum_{j=1}^{p} |\hat{\beta}_j|$, causes many parameters to go to 0, allowing for a few larger ones. Using the weight prediction model described above, this would mean one foot parameter would go to 1 and the other to 0. This is positive in two ways and negative in one.

good: It makes the resulting model easier to interpret, since there are fewer non–zero parameters with numerically large weight.

good: The number of non–zero parameters will (ideally) correspond to the true dimensionality of the problem.

bad: Successive runs of the model (successive minimisations of the objective function) during cross-validation may result in a different set of (non–zero) parameters, which can hinder interpretation.

There is no closed–form term for the minimum of the lasso objective function, since the $L1$ norm is not differentiable, but it can be expressed as a constrained optimisation problem:

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \left(y_i - x_i^{\mathsf{T}}\beta\right)^2 \tag{4.14}$$

$$\text{subject to} \quad |\hat{\beta}|_1 \leq t. \tag{4.15}$$

In Langrangian form, its link to the ridge becomes obvious:

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{4.16}$$

As for the ridge regression, the parameter λ must be chosen. The extremes, 0 and ∞ are the same as for the ridge regression. In signal processing literature the lasso method is called *basis pursuit* [51].

4.5.3 Higher Order of Regularisation

The general form for the penalty function class of which the lasso and ridge are members is

$$\frac{1}{2} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \tag{4.17}$$

where q controls the behaviour of the model fitting and the way that elements of β approach zero [52]. Larger q allows more complex models that do not select against multiple correlated predictors. Lower values encourage sparsity in predictors, i.e. many β elements are zero. The case of $0 < q < 1$ is called the *bridge regression* [53]. In this case the objective function is not convex and hence a minimum does not exist [54]. Fu has carried out a comparison of the bridge and lasso techniques [55]. The case $q = 0$ is called *subset selection* [56]. Here, the l_0-norm means adjusting λ really means concretely choosing how many parameters to search for. The sparsity of the β is fixed. The lasso l_1 is represents lowest norm (encourages sparsity) for which the objective function is convex (solvable).

4.6 Ridge or Lasso? A Compromise: The Elastic Net

The ridge regression is more gentle. It is ideal when you believe your parameters are all useful and they have comparatively small contributions [46]. It limits the absolute value of parameters, which offers better predictions. Correlated values will tend to approach a common value and no predictors will ‘drop out’ of the model (by getting a zero–parameter). Increasing λ does not help this. Rather, it encourages the parameters to approached a single shared value [50, 57].

The lasso is appropriate when you have many correlated variables (which is necessarily the case when $p > N$ [52]. Among a group of correlated predictors, a single predictor will be chosen and the rest will fall out of the model (parameter value zero). Likewise, predictors not correlated with the response are removed from the model. This improves interpretation of the model [46]. In the case of, for example, gene expression data [58], where there are many parameters this property is extremely valuable.

Some datasets are distinct in having: a) few observations; b) many predictors; c) highly correlated subgroups of predictors, and; d) many irrelevant predictors. Many genomic applications share these traits. Here we may wish to remove many predictors from the model entirely (lasso) while shrinking remaining parameter values to aid predictive power (ridge). A trade–off call the *elastic net* [59] attempts to cover this case.

Suppose we combine Eqs. (4.13) and (4.16) to give a new objective function:

$$\mathrm{RSS}(\hat{\beta}, \lambda_\mathrm{r}, \lambda_\mathrm{l}) = (y - x_i\beta)^2 + \lambda_\mathrm{r}|\beta|^2 + \lambda_\mathrm{l}|\beta|_1, \tag{4.18}$$

where λ_r and λ_l weight the ridge and lasso penalties, respectively. This would appear to solve the problem. For the sake of tidiness, we can rewrite Eq. (4.18) by setting

$$\alpha = \frac{\lambda_\mathrm{r}}{\lambda_\mathrm{r} + \lambda_\mathrm{l}}, \tag{4.19}$$

so that the objective function becomes

$$\mathrm{RSS}(\hat{\beta}, \lambda_\mathrm{r}, \lambda_\mathrm{l}) = (y - x_i\beta)^2 + (1 - \alpha)|\beta|^2 + \alpha|\beta|_1, \tag{4.20}$$

since the values λ_r and λ_l themselves are arbitrary. This allows us to weight the ridge and lasso constraints against each other. This penalty formulation is known as the *naïve* [59] or *vanilla elastic net* [54].

In practice, good performances for this model structure requires that either λ_r or λ_l are small, i.e. the model either performs like a ridge or like a lasso. This is related to the fact that parameters are effectively shrunk twice, once under the ridge regime and once under the lasso. The solution is to *undo* the ridge shrinkage [54].

Friedman, Hastie and Tibishirani's R package `glmnet` [60] uses the following implementation:

$$\beta^{\mathrm{e}} = \min_{\beta} \left[\frac{1}{2N} \left(y - \mathbf{X}^{\mathsf{T}} \beta \right)^2 + \lambda_e P_\alpha(\beta) \right], \tag{4.21}$$

where

$$P_\alpha(\beta) = (1 - \alpha) \frac{1}{2} |\beta|_2^2 + \alpha |\beta|_1. \tag{4.22}$$

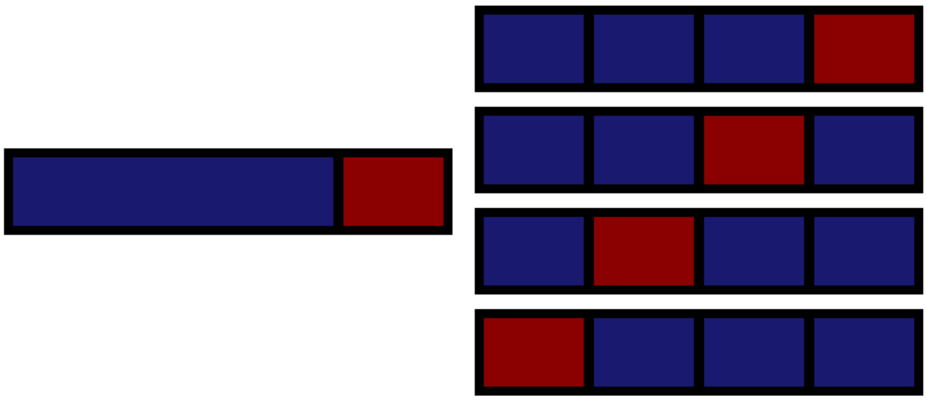
In this form, the λ_e term allows us to weight the effect of the parameter penalty against the residual contribution. The elastic net mixing parameter α term controls the relative weights of the ridge and lasso terms. Lasso corresponds to $\alpha = 1$ and ridge to $\alpha = 0$.

4.7 Training the Model

A model is selected based on its residuals. Since a model can in many cases fit the response perfectly (zero residual), this cannot be a useful measure of model acuity. However, fitting a model and testing it on new data will keep over-fitted models in check. This requires a larger set of observations: one part to train on, another to test. This situation is shown in Fig. 4.4a. However this only gives us one change to 'get it right'. It is dependant on the training set and test set both being highly representative of the underlying true data, both in terms of correlations with predictors but also in model noise. This could be repeated, of course, with new sets of data. But data is expensive and often fairly limited.

Therefore data can be split up into several chunks and multiple models can be trained [46]. Say we split the data into four chunks. Each chunk is removed one at a time, and the model is trained on the remaining three chunks and tested on the one removed chunk. See Fig. 4.4b. The model parameters can then be averaged over all the runs to give a more realistic model, and one that hopefully is better able to predict new data.

This process is called cross-validation. Depending on how we split the data (for example just two splits, or as many splits as there are observations), we will need



(A) Simple training/test set split. (B) Cross-validation [52].

FIGURE 4.4: Methods for preventing overfitting.

to run the model multiple times. The more times we run the model, the more information we can extract from each data point. But the computation time also increases. Depending on the model complexity, this can become so large that it is impractical. The number of splits is called the number of folds. Depending on the data size, cross validations of five– to twenty–fold are common, while N–fold cross validation, called jack knifing, is also common.

When datasets are small we cannot entirely save ourselves by cross validating. Suppose we start with a reasonably small dataset of 30 observations. Performing five-fold cross-validation on this will give us four models each trained on 25 observations and tested on 5 observations. Since 25 is not very many points, there may be significant variation — 25 points are not enough to 'absorb' outliers in the training data, so the model will be affected. Likewise, if we have unusual in the test set, they will not be drowned out by the rest of the points. Hence quite large errors can appear [46]. Figure 4.5 illustrates the effect of this principle.

4.7.1 Choosing λ

λ is a hyper-parameter in the model. It is not used to predict the data directly. Rather, it is used to tune how complex the model becomes, i.e. how many effective parameters the model contains.

In searching for an appropriate value for λ we are really searching for the value which minimises the prediction error on the test set. Hence we run the model

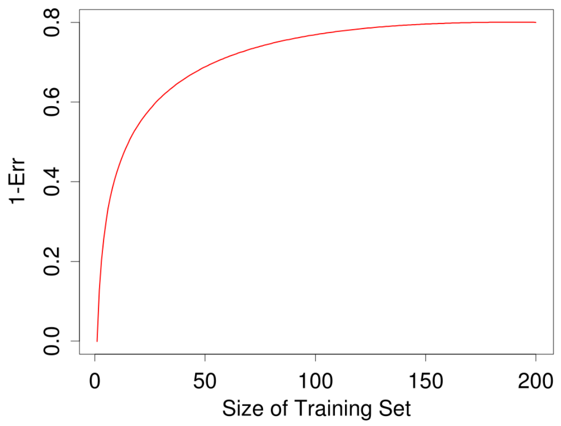


FIGURE 4.5: The model accuracy, $1 - \text{Err}$, increases with dataset size. For small datasets, the effect can be very large [46, Fig. 7.8]. Err is the expected test error, which is the mean of errors on test data in cross-validation runs.

multiple times for various λ. The value which consistently scores a low prediction error is chosen.

This is often combined with cross validation. The reason is that an ideal λ exists for each set of data. However our interest is not in modelling the data itself — we want to be able to model new data. If a model is well-trained on a set of training data, there will be bias. The training data is systematically and permanently different from the true data distribution, and the difference in the two sets is the bias. The term bias describes the fact that the model is fit to the training data, which is an imperfect representation of reality. In terms of the ridge regression, this implies that the objective function consists mostly of the residuals, i.e. a small λ.

A large λ reduces model complexity, which means fewer parameters are trained. With fewer parameters, the model is less likely to capture the underlying error in the data, so bias is reduced. However successive modellings are likely to land on different parameter values — hence prediction variance is increased.

Figure 4.6 illustrates the trade-off. Practical values for λ are often extremely small. Therefore $\log \lambda$ values are often given.

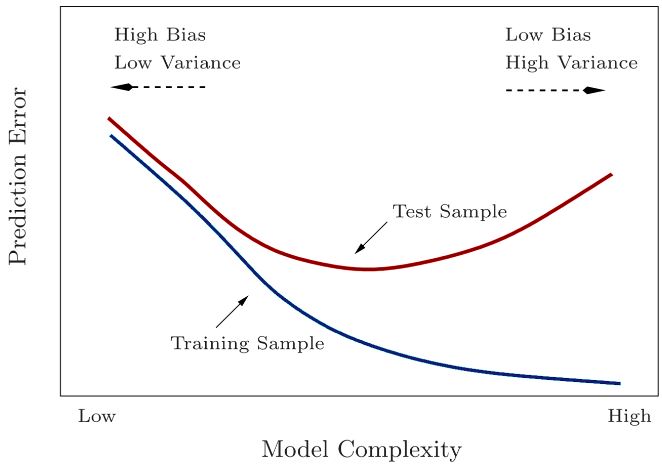


FIGURE 4.6: Bias–variance trade-off [46, p. 38].

Chapter 5

Parameter Investigation and Definitions

This section will introduce the specific codon measures that we will be calculating and using as predictors in the model. Some are popular and common in literature today [31] while others are ones that we define here. The newly defined measures will generally lack an *a priori* biological justification. This is in line with the aim of the thesis.

5.1 Reference-Based Measures

Two of the codon measures that I will be using rely on a comparison between the given gene and a reference for its class. Here we introduce the measures and consider what effect the choice of reference has on the measure.

5.1.1 Codon Adaptation Index (CAI)

Developed by Sharp and Li in 1986 [32, 33] the codon adaptation index measures the distance of a gene from a reference gene. Its formula is

$$\mathrm{CAI}(L(g)) = \exp\left(\frac{1}{L}\sum_{l=1}^{L}\log w_{c(l)}\right) = \left(\prod_{l=1}^{L} w_{c(l)}\right)^{\frac{1}{L}}, \tag{5.1}$$

where L is the length of gene g and $w_c(l)$ is the relative adaptiveness of the codon c in the reference genes (not g). Relative adaptiveness is defined

$$w_c = \frac{f_c}{\max(f_s)},\ s \in \{\mathcal{C}_a\}, \tag{5.2}$$

where f_c is the frequency of codon c which is the l^{th} codon in g. a is the amino acid encoded by c and $\{\mathcal{C}_a\}$ is the set of synonymous codons encoding amino acid a. Certain codons will appear multiple times in the gene. Each time a certain codon appears, the contribution to the sum in Eq. (5.1) will be the same. Hence we can rewrite the equation to sum over codons rather than length, and use counts rather than frequencies. This makes the dependence on the actual gene more clear. The more usual form is

$$\mathrm{CAI}(o(g)) = \exp\left(\frac{1}{o_{\mathrm{tot}}}\sum_{c\in\mathcal{C}} o_c \log w_c\right) = \left(\prod_{c\in\mathcal{C}} o_c w_c\right)^{\frac{1}{o_{\mathrm{tot}}}}, \tag{5.3}$$

where o_{tot} is the total number of unique codons in the sequence and o_c is the number of occurrences of codon c.

In general, only *sense* codons are included in the count. This means STOP codons are removed. Often also single–codon amino acids are removed, since they have no freedom to choose a codon and therefore cannot deviate from any reference gene.

A CAI ranges from zero to one. A value of one implies that the optimal codon (according to the reference gene) is used every time. Going toward zero, more rare codons are used.

5.1.2 Two–Codon Adaptation Index (CAI$_2$)

The two–codon CAI measure is defined entirely analogously to Eqs. (5.1) and (5.3):

$$\mathrm{CAI}_2(\phi(g)) = \left(\prod_{c \in \mathcal{D}} \phi_c \omega_c \right)^{\frac{1}{\phi_{\mathrm{tot}}}} \tag{5.4}$$

$$\omega_c = \frac{f_c}{\max(f_s)}, \; s \in \{\mathcal{D}_a\}, \tag{5.5}$$

where ϕ is the two–codon analogy of o, c and a are the paired equivalents of their earlier definitions and $\mathcal{D}$ is the set of all codon pairs.

The CAI$_2$ is interpreted just as the CAI measure.

5.1.3 tRNA Adaptation Index (tAI)

This is closely related to the CAI index. But rather than comparing the codon use in a given gene to the codon use in a set of reference genes, codon use is related to the pool of tRNA available for translation [28, 31, 61]. The premise is that if you use a codon for which there are only very few tRNA available to carry out the translation, translation rates will be slowed and the protein encoded by the gene will be produced in smaller amounts.

The tRNA availability is not known directly. tRNA is a biological molecule which is encoded in DNA so levels are estimated by counting the number of copies of tRNA DNA there is in a given organism's genome. This is called the tRNA gene copy number (tGCN) [61] which correlates strongly with actual tRNA levels within the cell [34, 35, 62].

Just like CAI, tAI is calculated as the geometric mean of codon weights w^t along the sequence g:

$$\mathrm{tAI}(L(g)) = \exp\left(\frac{1}{L} \sum_{l=1}^{L} \log w^t_{c(l)} \right) = \left(\prod_{l=1}^{L} w^t_{c(l)} \right)^{\frac{1}{L}} \tag{5.6}$$

$$\mathrm{tAI}(o(g)) = \exp\left(\frac{1}{o_{\mathrm{tot}}} \sum_{c \in \mathcal{C}} o_c \log w^t_c \right) = \left(\prod_{c \in \mathcal{C}} o_c w^t_c \right)^{\frac{1}{o_{\mathrm{tot}}}} . \tag{5.7}$$

However the definition for the relative adaptiveness w^t is quite different in this case. First absolute relative adaptiveness W is calculated for each codon c as

$$W_c = \sum_{j=1}^{n}(1 - s_{cj})\mathrm{tGCN}_{cj}, \tag{5.8}$$

where j indexes the n tRNAs that match codon c and s is a selective coupling constraint between codon c and its j^{th} tRNA. Here we set $s = 0$ if it is a direct match and $s = 0.5$ if there is a mismatch at the final position [61], although more involved constraints have been defined [28]. Then

$$w_c = \begin{cases} \frac{W_c}{\max(W_s)} & \text{if } W_c \neq 0 \\ \bar{w}_s & \text{else} \end{cases}, \quad s \in \{\mathcal{C}_a\}, \tag{5.9}$$

with s, $\mathcal{C}$ and a as for Eq. (5.2).

5.2 Dependence On Reference Set In Reference–Based Measures

There is a dependence on the particular reference set in each of the three methods described in the w, w_2 and w_{tAI} vectors. Often software supplies standard w–vectors for use in calculations [10, 37], but sometimes researchers supply or calculate their own [63].

In the case of the R package `seqinr` [37] the w vector for CAI calculation is supplied for a number of organisms (Eschericia coli, Bascillus subtilis and Saccharomyces cerevisiae). All data are referenced to the codonW program [10], but: "According to this source file, there were no reference for Escherichia coli [...]. It turns out that the data for Escherichia coli [...] are identical to table 1 in Sharp and Li (1987) [33]." This adaptation index was calculated from 24 highly expressed genes where half are ribosomal and half are metabolic.

24 is quite a small number of genes and the small number was presumably accepted because not many genes had been sequenced at the time when the CAI was first described [64]. An essential property of the reference set is that it consists of genes with somehow good codon usage. Since more essential genes are presumably written in an ideal, they are good candidates for representing the ideal codon usage of an organism.

Beyond the loose phrase "highly expressed genes" there is little information on selecting a good reference set. Researchers that select their own reference set often start with the entire genome and extract only the ribosomal genes, on which they base their w vector [63]. This is not a large problem, since research suggests that CAI values are robust and largely insensitive to the reference set [64]. This indicates there is indeed a consistency in codon usage among all (any) highly expressed genes.

We investigate this assumption by calculating w values for a number of E. coli genomes. The built-in w vector from `seqinr` and codonW [33] will also be considered.

5.2.1 Reference Dependence for CAI

Five genomes were downloaded from the Nucleotide database at the National Center for Biotechnology Information [65]. The genomes are all E. coli:

1. K-12 substr. MG1655 (NC_000913)
2. BL21-Gold(DE3)pLysS AG (NC_012947.1)
3. O26:H11 str. 11368 (NC_01336.1)
4. SE11 (NC_011415.1)
5. O157:H7 str. Sakai (NC_002695.1)

See Figs. 5.1 and 5.2. We see that there are clear differences between the references from different genes. However we cannot quantify whether this is significant, since we do not know the scale of the difference.

5.2.2 Reference Genes for CAI$_2$

Using the same reference genes as for the standard CAI, we perform a similar multidimensional scaling for the CAI$_2$, see Fig. 5.3 on page 51.

5.2.3 Reference Genes for tAI

12 sets of tRNA copy numbers were downloaded. A multi-dimensional scaling was performed on these, too. We see clear grouping of some of the adaptiveness vectors. We would expect the grouped copy number sets to contribute equally

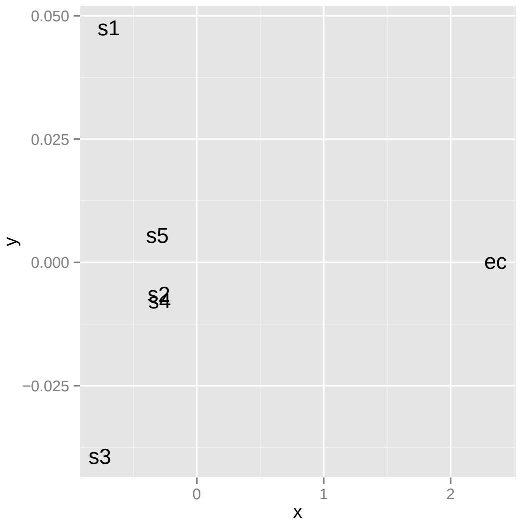


FIGURE 5.1: Multidimensional scaling of w vectors for five E. coli genomes, plus the common 24–gene reference.

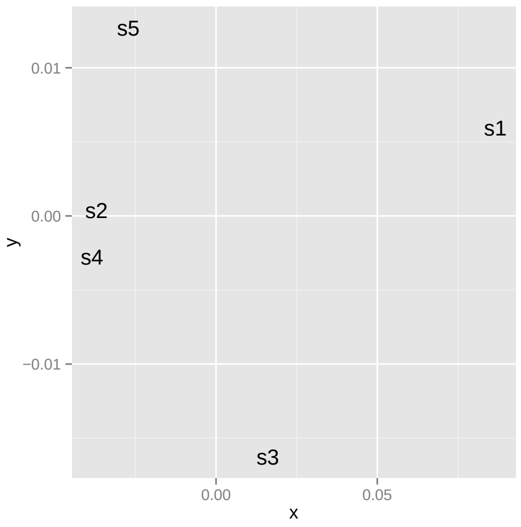


FIGURE 5.2: Multidimensional scaling of w vectors for five E. coli genomes, without the common 24–gene reference.

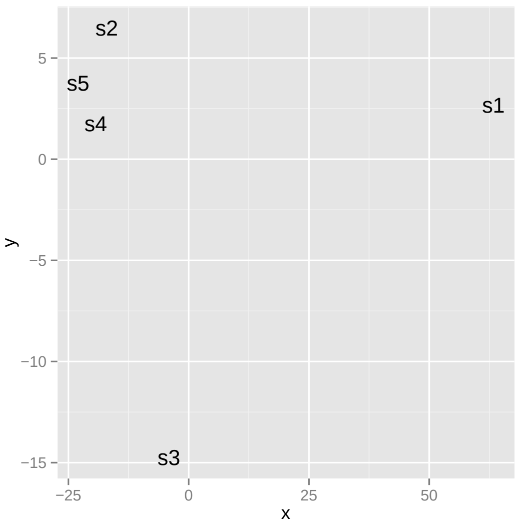


Figure 5.3: Multidimensional scaling of ω vectors for five E. coli genomes.

in a model. Again, we do not have a sense of scale in this plot. Hence it may be that all the measures are extremely correlated with each other, compared to correlations between a response variable and other predictors. Hence it is possible that they either all are included at a low level (ridge–style) or that on random one is included each time (lasso–style).

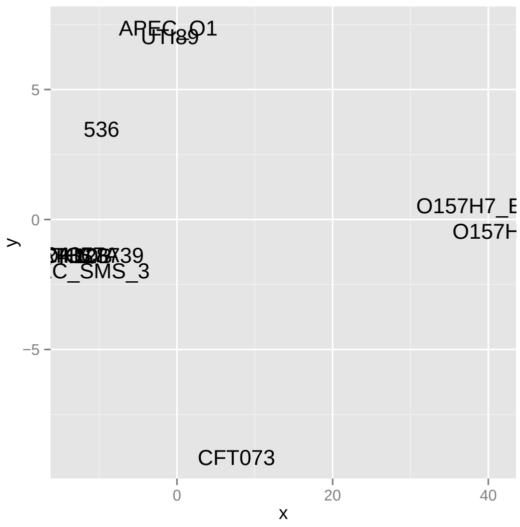


Figure 5.4: Multidimensional scaling of w^t vectors for 12 sets of E. coli tRNA copy number tables.

5.3 Non–Reference-Based Measures

The bulk of the predictors calculated for this model are not measurements relative to a reference set. They are simpler to define without having to worry about whether the reference set is a good reference. Also, given the data structure we have (many identically-coding genes expressed in the same host), the relativity concept loses power — after all, each gene is measured *in reference to* the same thing.

Here briefly describe the new measures without much discussion. The aim of the thesis is to find relevant predictors *without* requiring a biological justification for them. After all, we do not understand all the biology in the system, so it is entirely possible that some predictor, that does not come with a biological justification, turns out to be highly predictive. Many predictor are grouped into natural families.

Codon counts Single and double.

$$o_c = \text{number of } c \text{ codons} \qquad o_\delta = \text{number of } \delta \text{ codons-pairs} \tag{5.10}$$

Codon frequencies Single and double.

$$f_c = \frac{o_c}{L} \qquad f_\delta = \frac{\phi_\delta}{L} \tag{5.11}$$

Relative synonymous codon usage (RSCU) Single and double [66].

$$\mathrm{RSCU}_c = \frac{f_c}{1/A} = f_c A \qquad \mathrm{RSCU}^2_\delta = \ldots = \phi_\delta D, \tag{5.12}$$

where A is the number of synonymous codons associated with c, and analogously for D and δ.

GC counts Total GC content, as well as in positions 1, 2 and 3.

Top and bottom codon within each amino acid.

$$T_a = \max_{c \in \mathcal{C}_a} f_c \qquad B_a = \min_{c \in \mathcal{C}_a} f_c. \tag{5.13}$$

Top and bottom pair counts and scaled by length.

Longest contiguous stretch of top bottom and scaled by length.

5.4 Window-Wise Analysis

Inspired by articles investigating bottleneck effects [2, 29, 30] we decide to include a window-wise aspect to the modelling. Each predictor will be calculated for a segment of the sequence. The data will be split into thirds as follows: the first third, the second third, the third third, the two final thirds and the whole sequence, see Fig. 5.5. Figures 5.6 and 5.7 show a multi-dimensional scaling of the observations in all the datasets. Not that the predictors are not weighted in this analysis. It is purely exploratory and just to see whether there might be a window-effect in the data.

We do not see any proof of a window dependance of predictors. This is most notable (and surprising) in the Kudla plot (facet K in Fig. 5.6), where we might

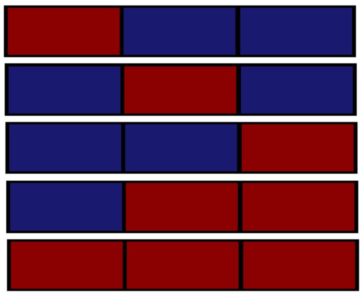


Figure 5.5: Window-wise analysis will be carried out in the splits show here.

expect to see separation, given the conclusions in the article. However it may well be that the distinguishing predictors are being overwhelmed and their influence hidden by all the other predictors. This show that on average (in some sense) across all predictors defined, there is no window effect.

The the facet containing all data, we do see some grouping, partly separating the different proteins, but also apparently separating the first window. Certainly the red dots seem more tightly grouped than the blue and green. (Admittedly they appear to be grouped *within* the blue and green dots.)

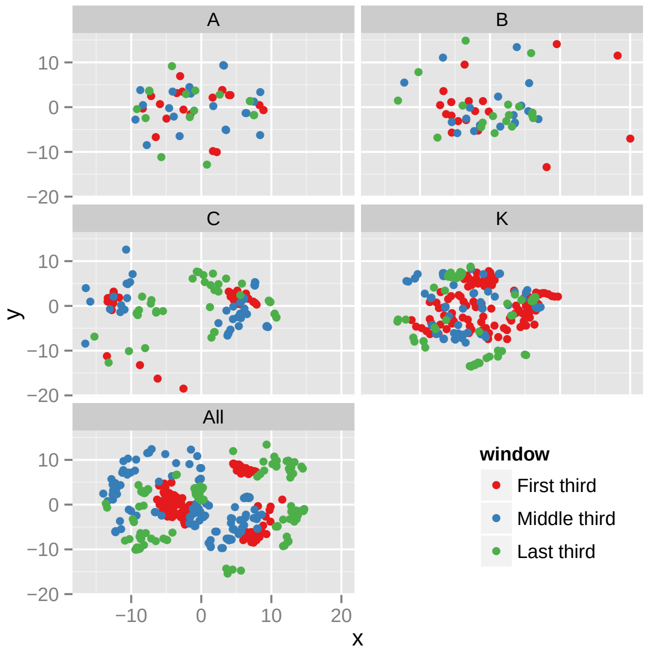


FIGURE 5.6: Multi-dimensional scaling displayed on two-dimensional scatter plots. There is some distinct grouping of the first window (red dots) in the full dataset plot. In the Kudla set, where the article claims that the first third is significantly different, there is separation under this view of the MDS.

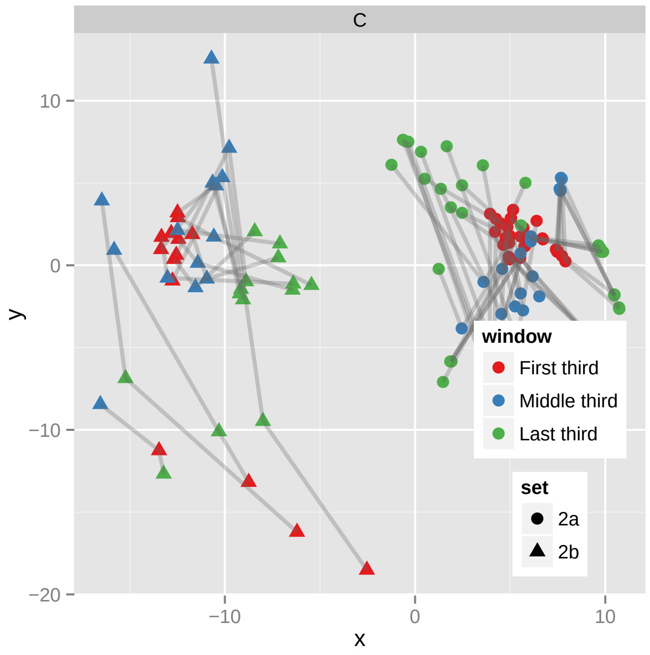


Figure 5.7: Multi-dimensional scaling for the combined Welch dataset. The thirds comprising a single sequence are joined up. We see complete separation of the two proteins.

Chapter 6

Model Training and Validation

6.1 Walk-through of Welch et al. analysis

This paper carries out a partial least squares (PLS) analysis aimed at predicting the expression of a set of synonymous genes. The data is for two different proteins, as described in Section 3.1.2. Three models are trained: one for each protein and one for the combined set.

We carry out the modelling in R using the partial least squares regression function `plsr` from the `pls` package [67].

The predictors are codon frequencies defined as the frequency that each codon is chosen for each particular amino acid. Hence, the codon frequencies within each amino acid sum to one. Therefore, in the case of two-codon amino acids, using just one codons frequency is sufficient. Generally the final codon frequency within each amino acid will contain redundant information (the last codon frequency is not needed, since we know they must sum to one). But in these cases, all codon frequencies are retained, since otherwise we are relying on the model to build this in.

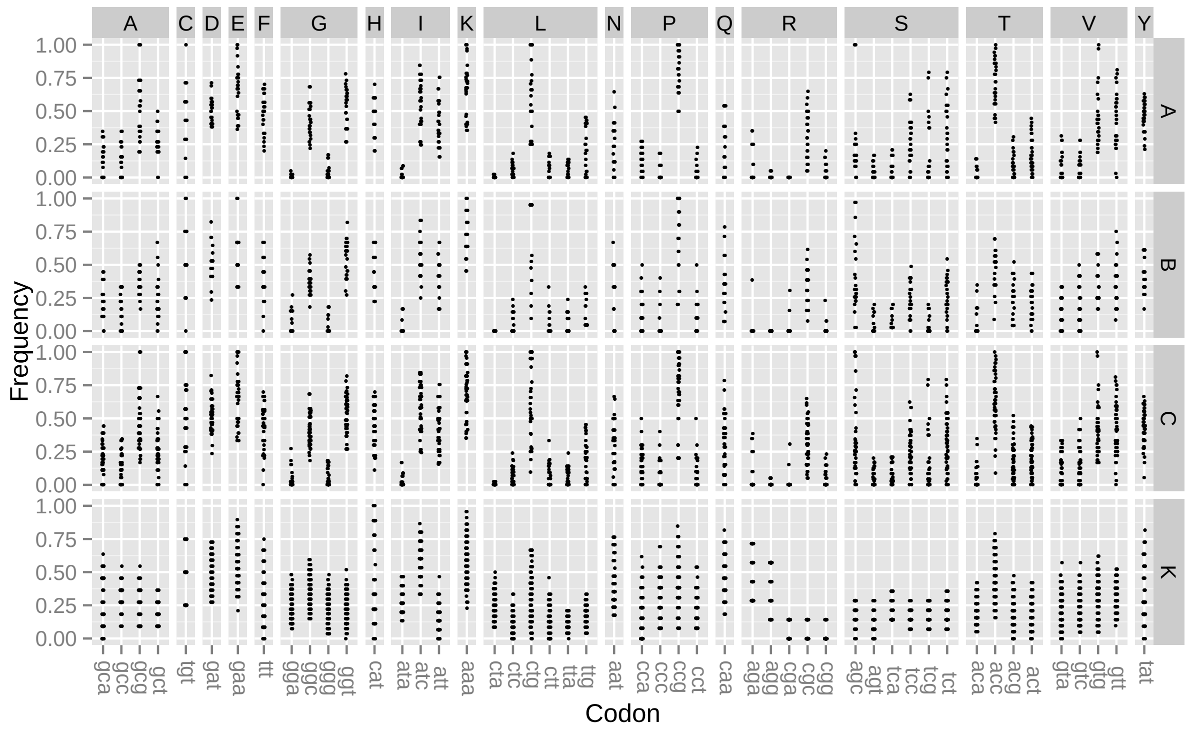


Figure 6.1: Codon frequencies within their amino acid per data set. Similar to RSCU, but not normalised by uniform frequency. Within each amino acid and gene, frequencies sum to one. Hence dots that are vertically closely grouped across the codons for a single amino acid means even codon usage, i.e. each codon is used the average number of times. See amino acid serine (S) for data set K: there are six possible codons and their frequencies appear grouped around $1/6 = 0.16$. The vertical variation within a single codon tells us about the variation in usage of that codon within that data set. For example, amino acid threonine (T) for data set A is commonly representative by codon ACC, although in some specific genes this preference is less strong (perhaps all way right down to parity with ACT). While sequences of both experiments are 'designed', we can see how they have been designed differently. Mutations in the Kudla sequences are uniform random, hence a more uniform distribution of points across the full range of the spectrum. Welch sequences were mutated randomly but according to probabilities from an E. coli codon usage table. (Note: Single–codon amino acids are excluding. Two–codon amino acids have one codon removed, since it adds no extra information. This pattern is not extended to other amino acids, since it puts extra strain on the model and renders parameters less interpretable.)

6.1.1 Quick Look At the Predictors

Figure 6.1 shows the Welch–style codon measures for each of the datasets in Welch (A and B) and Kudla. The predictors are calculated as codon frequencies normed within their group of synonyms. Specifically the Welch–style frequency f^W is

$$f_c^W = \frac{f_c}{\sum_{c'} f_{c'}},\ c' \in \mathcal{C}_a. \tag{6.1}$$

6.1.2 Modelling Process

The data will be group in five data sets:

1. Set A from Welch.
2. Set B from Welch.
3. Set C from Welch (a combination of sets A and B).
4. The Kudla dataset.
5. The complete datasets combined.

In the article we have analyses on datasets a–c, which allow us to make direct comparisons and confirmations [5]. For each dataset 1–5, five-fold cross-validation is performed

6.1.2.1 Model Complexity

The `pls` modelling method allows a total number of parameters equal to the number of observations. However, there is a limit to the usefulness of many parameters. Using the cross–validation to test the models, the number of parameters can also be selected. This will ensure the highest predictive power possible. Figures 6.2 to 6.4 show models with an increasing number of components (one to nine) for each of the Welch data subsets (a, b and c). Below each set of models is a plot showing the variation of the predictive power of the model with increasing model complexity.

The incredibly low cross-validation R^2 values in (in particular in dataset B) may be due to the very small datasets. There are 24 points in set B. Under

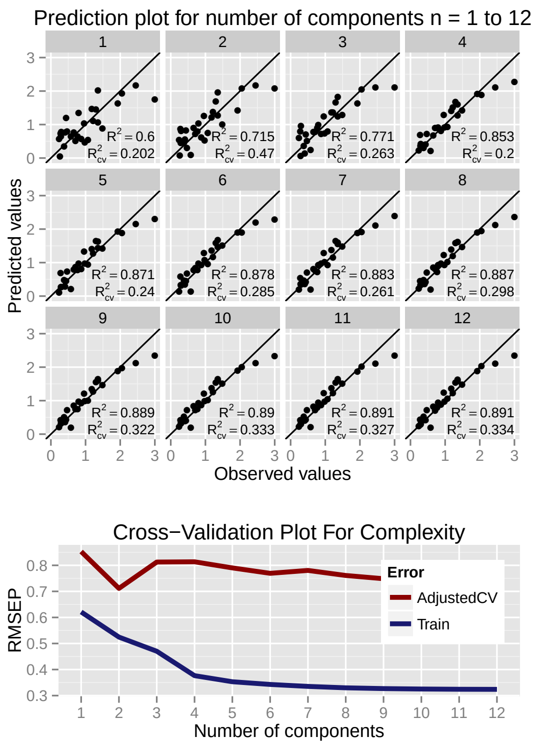


Figure 6.2: Welch set A. There is a sharp minimum in error (root of mean of squared error of prediction, RMSEP) at 2 components. Owing to the very small dataset, we put this down to variance in the cross-validation folds. We choose to use 6 components.

five-fold cross validation, this leaves just approximately 20 points for modelling and 4 or 5 points for testing. Small datasets in both the training and testing parts lead to high variance (and potentially very high errors). It may be the case that taking the average of the five estimated sets of model parameters does give a good model. Hence, the models may perform better than their cross-validation R^2 values suggest. This is a natural consequence of small datasets: the cross-validation error is exaggerated [46, Sec. 7.10.1], see Fig. 4.5.

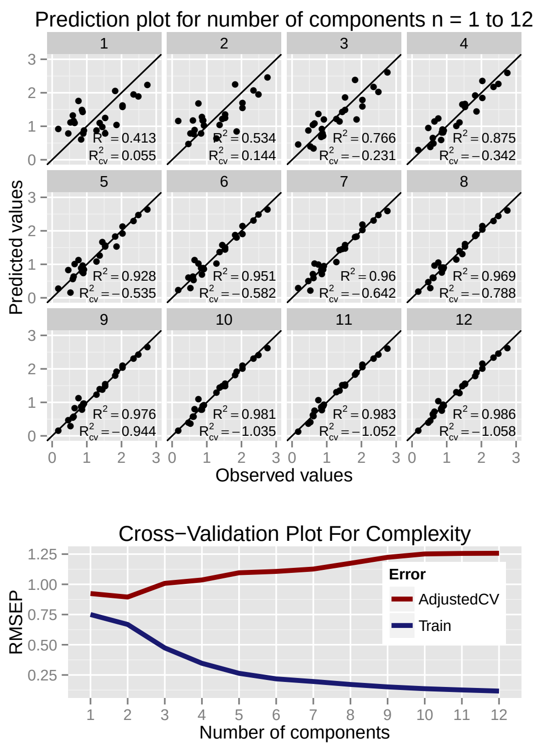


FIGURE 6.3: Welch set B. The cross-validation optimum is very low at around 2 components. To avoid a case of a local minimum (or false minimum due to variance) we choose the number of components to be 4. Cross-validation R^2 are extremely low everywhere. This cannot be explained at this stage.

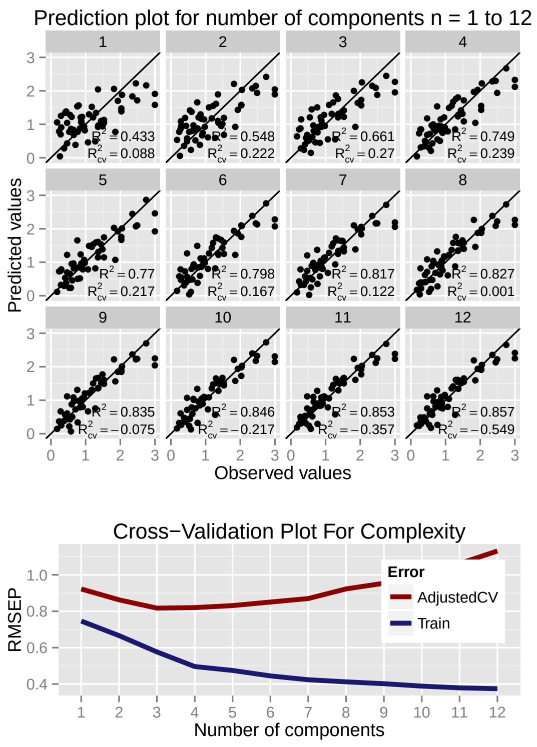
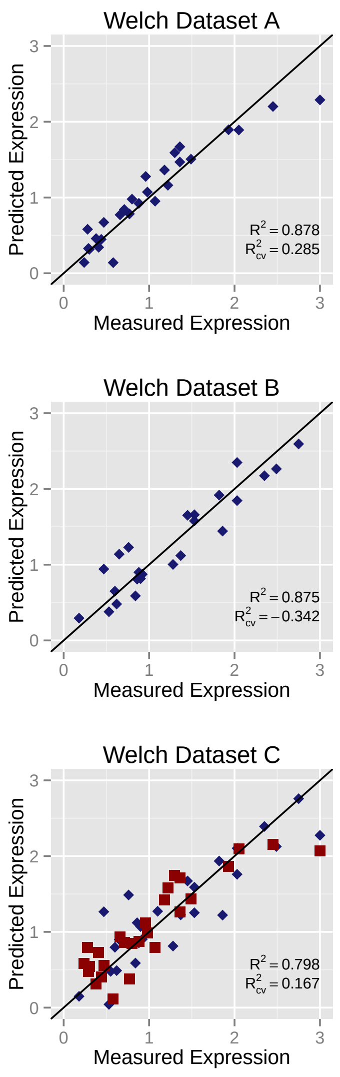
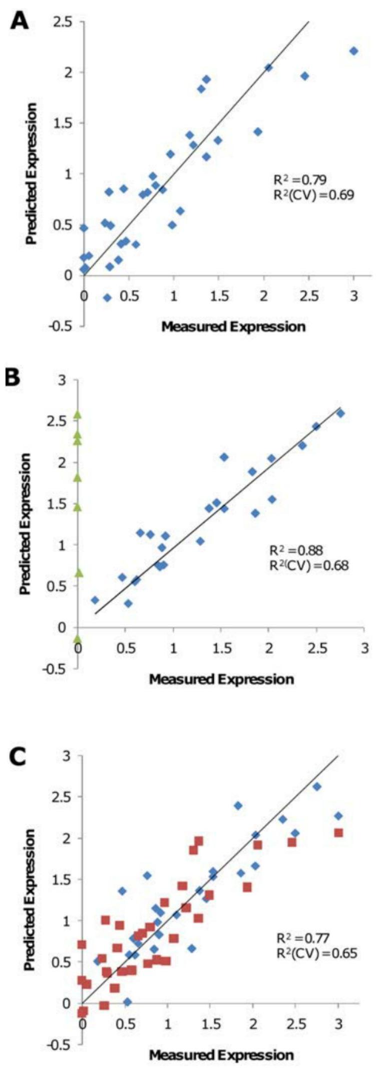


Figure 6.4: Welch set C. The cross validation curve is extremely shallow. Optimal number of components is chosen to be 6. Like set B, this set suffers from extremely poor apparent cross-validation R^2, peaking at a little over 0.2.





(A) Recreation of Welch analysis on the data in the article.

(B) The modelling result as summarised in the article [5, Fig. 2].

FIGURE 6.5: The two results are very similar, right down to very small details. Given this similarity of result, it is surprising that our cross-validated R^2 values are so low. The model is trained on the data by five–fold cross-validation. The trained model is then used to predict itself, as in the article. The colours in the lower plots represent dataset: the red points are from set A and the blue from B.

6.1.3 Extending the Welch Technique to Kudla Data

The data from Kudla et al. [2] was pre-processed as described in the Welch article: the response variable was scaled to have a maximum of 3 and then each dataset (A, B and C) was column–centred and –scaled, i.e. the response and predictors. A model was then trained using five–fold cross-validation. This process was repeated ten times and at each stage the number of components giving the lowest cross-validation error was noted, see Fig. 6.6 for a view of each modelling. Figure 6.7 shows the full situation for a single modelling run. We see clearly that training error falls off as model complexity rises, while cross-validation error ultimately rises.

Figure 6.8 shows the model predicting on its own training data. It appears to have projected the data into a space where the data lies fairly close together. It has a fair cross-validation $R^2 = 0.445$, but this does still not compare with the numbers reported from the Welch article.

Something which was not done on in the Welch article was to keep a separate test set to judge the model performance. We did this for the Kudla data set, withholding 20 observations selected at random (out of the 154 total). The result of this testing is shown in Fig. 6.9. There is certainly a good correlation: $R^2 = 0.69$.

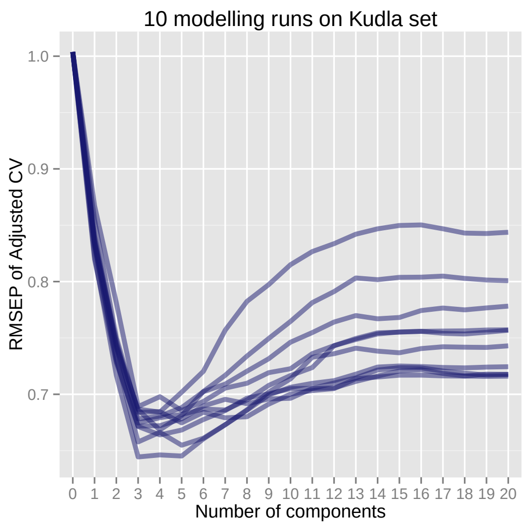


Figure 6.6: Cross-validation error for 10 model runs on Kudla data. The smallest error occurs around 5 components, although there is a bit of variance. From this we suggest that 5 components should be used for predictions.

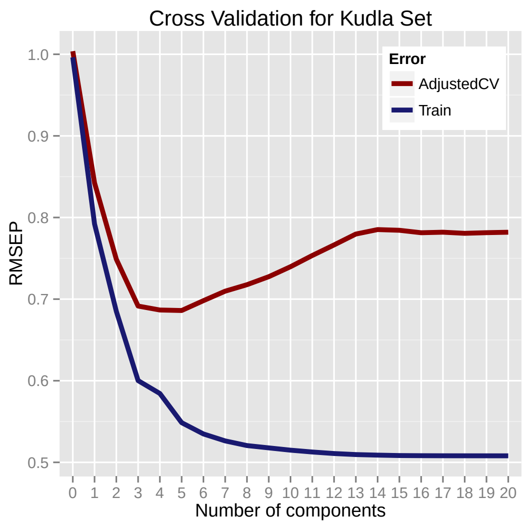


Figure 6.7: Cross validation plot. A larger number of components results in a better training data fit (smaller root mean squared error of prediction, RMSEP). The cross validation error has a fairly flat minimum between 3 and 7. A sensible and reasonably conservative choice for the number of component is 5.

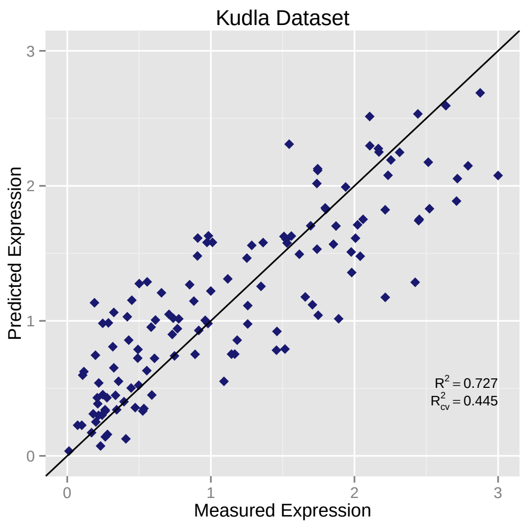


FIGURE 6.8: An indication of model performance: prediction on training data. This plot corresponds to the plots in Fig. 6.5.

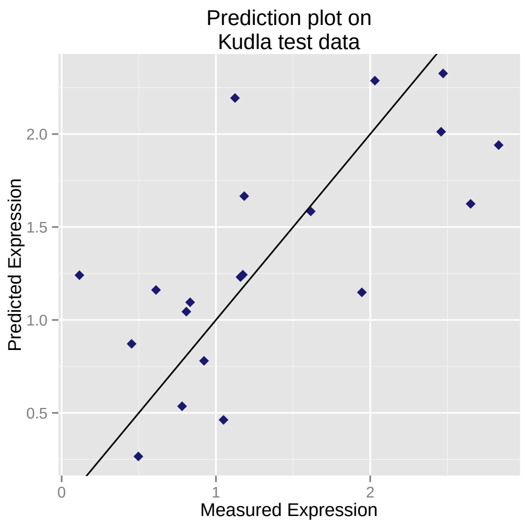


FIGURE 6.9: Testing the model on fresh data. Line indicates perfect fit. The prediction correlation on the test data is $R^2 = 0.69$

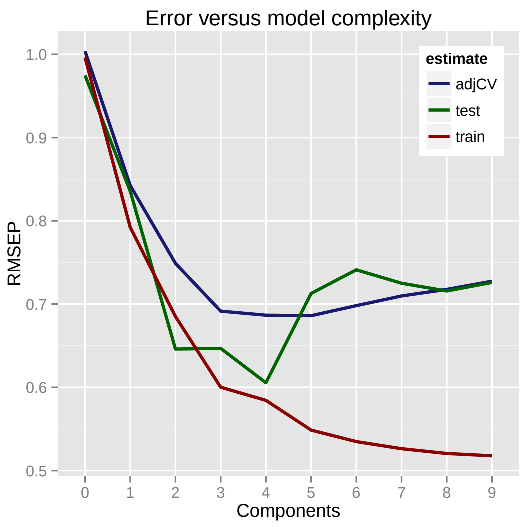


Figure 6.10: Having completed the modelling, we are free to ask — hypothetically — if we picked the best number of components. From this plot we see that we could have avoided some overfitting by choosing a smaller model. However, the dip in test error is likely due to uncertainty in the parameters. Given another set of test data, it might not have looked so good.

6.1.4 Training a full model

All the datasets are now combined and a full model is trained. A random subset of 40 observations is withheld for testing. The response is normalised to have a maximum value of three within each data set. The cross-validation plot of error against model complexity for 10 model runs is shown in Fig. 6.11. Figure 6.12 show the trace of cross-validation performance for a single model run.

Testing the model on the unseen test set yields a result shown in Fig. 6.13, with $R^2 = 0.68$. This is slightly better than for just the Kudla data set. On the one hand, we had far more points on which to train a model. On the other hand, there was a greater variety of genes. This is a good result

From the figure, it does seem that the model is better at predicting the lower expressions. This indicates that the residual has a dependence on the response; or at least that they share a cause. This could indicate that the response variable should be pre-processed in some way. We move on to carry out a full modelling with new predictors.

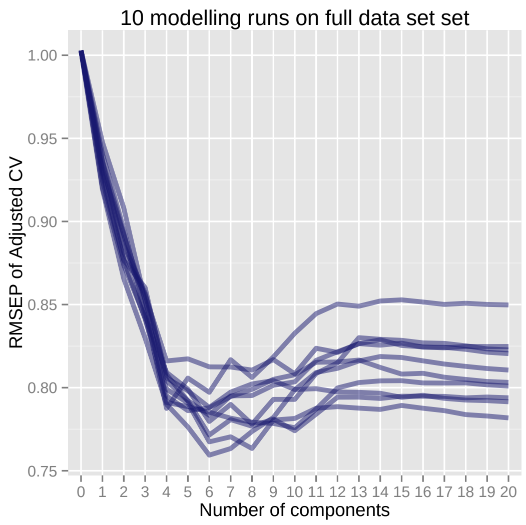


FIGURE 6.11: Modelling the combined dataset of Kudla and Welch. The optimal number of components for prediction is between 5 and 9. We choose 5 as a conservative estimate.

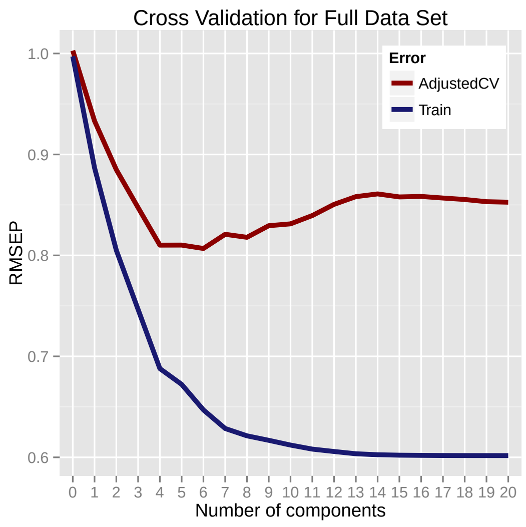


FIGURE 6.12: The prediction performance is best at about 6 components, but we choose in favour of a simpler model.

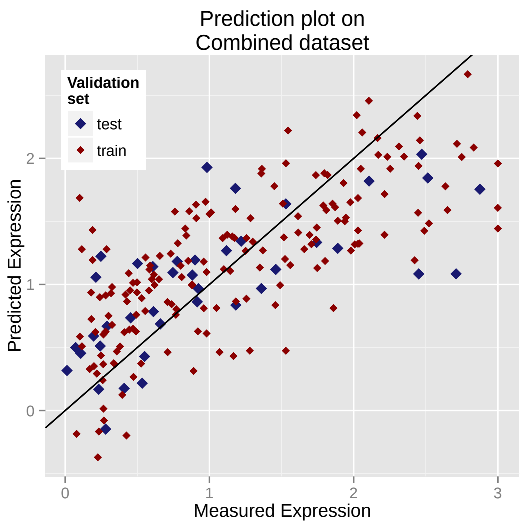


FIGURE 6.13: Validation plot for the combined dataset of Welch and Kudla. Welch-style analysis has been applied using a five–component partial least squares model. This results in a prediction correlation of $R^2 = 0.68$. The test data are shown as well as the training data. The test data is projected into the cloud of points from the training set. This indicates good model structure. However the points are quite dispersed. There is a lot of residual variance in the data.

6.2 Model Definition

6.2.1 Choosing an R Package

There are many choices for linear modelling when using R [68]. We considered `sparsenet` [69, 70], `lars` [71] and `glmnet` [60]. We decided to use `glmnet`, since it has a simpler interface and considerably more on-line support than the newer `sparsenet` package. In general `sparsenet` might be preferable for reasons of time efficiency. But at this stage, waiting a few extra moments for a model to be trained is not considered a big expense.

6.2.2 Selecting α and λ

Since the datasets we are have contain relatively few samples and a large number of predictors, we have decided to use the elastic net modelling technique. This requires that we find an appropriate α value. In principle the model also requires a choice of λ, but in practice `glmnet` calculates the model for many choices of λ and chooses the best one by cross-validation. However in order to find a choice of α we must decide on a range of λs over which α will be assessed.

Performing a walk through the parameter space enclosed by $\alpha = (0, 1]$ and $\log \lambda = [-5, 1]$ at a resolution of 150 points in each direction, we total 22,500 combinations. At each point a model is fitted to the training data with five-fold cross-validation. Each data subset of treated separately. (α, λ) pairs are then compared based on their mean cross-validation residual. The result is plotted with a heatmap in Fig. 6.14. The plot for Welch dataset B is shown again in Fig. 6.15, since its residuals are better shown on a separate scale.

The elastic net mixing parameter α is often chosen on qualitative grounds based on a knowledge of the underlying data structure [46]. Hence choosing one value for α is deemed sufficient, since the nature of the underlying data is the same. From Fig. 6.14 this we conclude that an appropriate value for the elastic net α is 0.25. Taking Fig. 6.15 into account, one might be inclined to choose a lower value. However the data in Welch set B has already been found to be particularly hard to model (see Fig. 6.3) and it certainly seems different to the other sets. In the interests of parsimony, we keep $\alpha = 0.25$ for all data sets.

Note that the λ parameter is essentially free for each model, hence we might assume that the model for Welch set B will search through areas of relatively small λ. Judging by the purple shade in the leftmost edge of Fig. 6.15, we hope that residuals are lower for these lower λ values, and that the lower residuals stretch up into slightly higher α values. If this is the case, setting $\alpha = 0.25$ should give a fairly good modelling result.

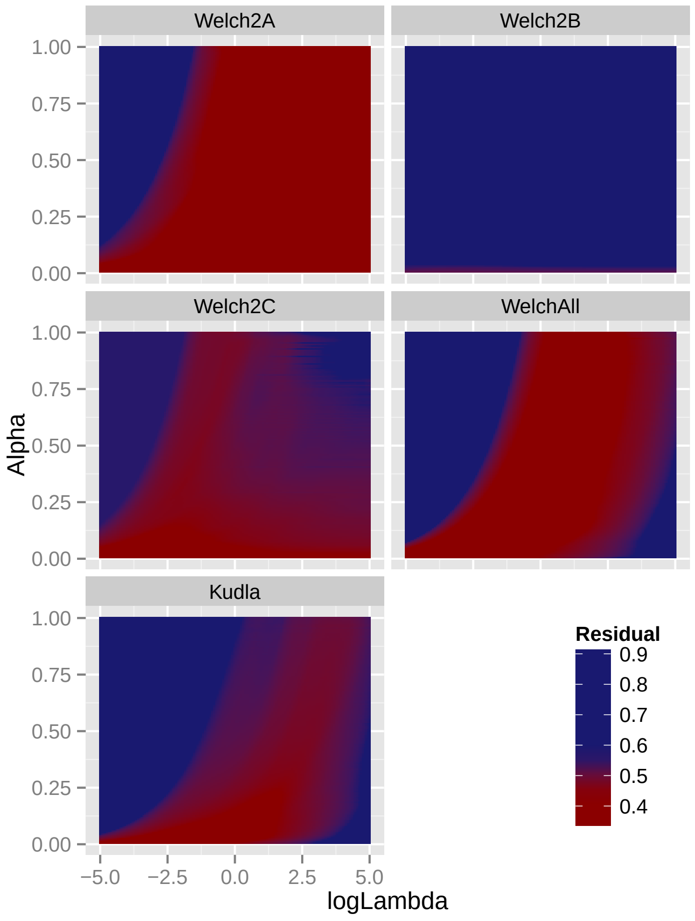


Figure 6.14: α and λ cross-validation plots for the datasets. The horizontal line $\alpha = 0.25$ seems to cross through mostly red areas, which indicate a low cross-validation residual.

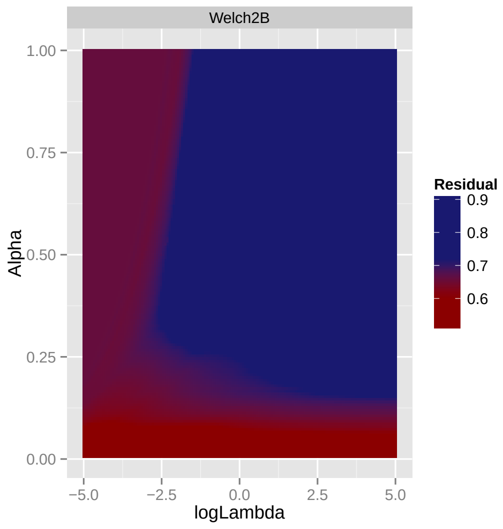


FIGURE 6.15: α and λ cross-validation Welch dataset 2B. Here we see more detail. Keeping α might not be ideal for this particular dataset, but in the interest of parsimony, we keep to a constant α value.

6.3 Model Training and Evaluation

6.3.1 Test Sets

The data consists of three sets (Welch set A, Welch set B and Kudla) and these are analysed in five collections (Welch set A, Welch set B, Kudla, Welch sets A and B and all sets together). Test sets were extracted before any analysis was carried out. Test observations were picked at random and if excluded from one set, excluded from all others. Figure 6.16 shows the principle.

This is done partly to ensure that at no point was a model trained on any data that is later used for testing. Training a model on one set and seeing certain features performing well as predictors may influence our predictor selection on other sets (containing some of the same observations). But it was also done to ensure valid comparisons between modelling. If, for example, there is an unusual data point with a hard-to-predict expression level, this will appear in all test sets and will mean less variability in test statistics.

The actual data splits are summarised in Table 6.1.

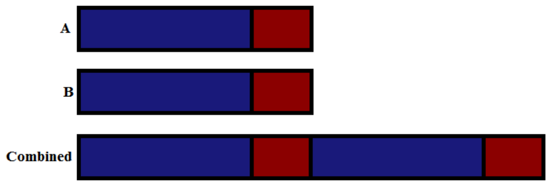


FIGURE 6.16: Graphical explanation of test–set selection methodology. The red areas of A and B contain the same data points at the red areas in the combined dataset. In reality, the test set is not contiguous; it is defined as a random proportion of the full set.

TABLE 6.1: Splits of training and test sets, and number of points in cross-validation folds for each dataset. Five-fold cross-validation. Cross-validation values are rounded. Actual partitions may differ by one datapoint.

Set	Total	Train	Test	% Test	CV train	CV test
Welch A	28	20	8	29	16	4
Welch B	24	18	6	25	14	4
Welch C	52	38	14	27	30	8
Kudla	154	125	29	19	100	25
All	206	163	43	21	130	33

6.3.2 Modelling Parameters

The full model has access to a total of 8,523 predictors (discounting window-wise predictors). Many of these are codon measures adapted to 'double' measures, i.e. rather than perhaps count occurrences of a particular codon, we count occurrences of a particular par of codons (in a particular order). Given that there are 61 sense codons (64 minus three STOP codons), this gives a total of 3,721 features in this measure alone. With such a large number of possible features, we also have the chance that some features will not be present in any genes. We only have a a total of 216 genes of maximum length 575 codons (the polymerase from Welch set A; the others, scFv from Welch set B and GFP from Kudla, have 281 and 240 codons, respectively), so it is not unlikely that some combinations will not appear.

The actual problem is that if all observations have the same value of a particular measure (0, or some other value), then it cannot possibly be of value to the model. This parameter will be conflated with an intercept term. Therefore any predictor whose variance is zero can be omitted from the model.

This situation can be thought of as ill-posed: we have some features that do not appear in our dataset, but they may actually be valid in reality. Likewise, suppose a rare feature is present in our data and found to be highly correlated with the response. This will potentially not be of very much predictive value for any new data, since that rare feature may not be present.

However, we will not pre-process the data by filtering out zero-variance predictors. During modelling, this adds an extra layer to the data handling, and the model type we are using will immediately find that zero-variance predictors contribute nothing that is not already accounted for by the intercept term. The product of this thesis will be a model — and hence (only) the required predictors can be calculated.

6.3.3 Increasing Model Complexity

In order to see whether the selected variables have good predictive value we will investigate them one at a time on the full dataset. The (relatively) large number of observations will encourage stable (low variance) cross-validation results. Once they have been investigate individually, we will build more complex models. We should see the cross-validation error drop with increased predictors.

Before we begin, we note that the modelling power will only be additive (in some sense) if each newly added set of predictors is uncorrelated with the already-present ones. Otherwise their contributions will be averaged (by the ridge part) and many individual predictors will be set to zero (removed by the lasso part). Their reduction in error rate will be less than might be expected.

6.3.4 Side Note: Effect of λ on Model Complexity

The effect of λ on the number of parameters can be been explicitly in Fig. 6.17. We see that as λ increases, the number of parameters falls dramatically. A larger λ gives more weight to the parameter penalty term ($P_\alpha(\beta)$ in Eq. (4.21)) compared to the residual term, which means parameters are increasingly forced to be smaller. Many predictors' parameters become exactly zero due to the ℓ_1 regularisation of lasso. Decreasing model size helps avoid overfitting, but too simple a model will fail to pick up structure in the data. Hence an optimum exists. This is found with by tracking the cross-validation error at each λ. The minimum is at the solid line in the figure, corresponding to 128 non-zero parameters in the model. Allowing for variance, and erring to the side of a simpler model and away from overfitting, the dotted line suggests a 92 parameter model is sufficient in this case. Figure 6.18 shows this more clearly.

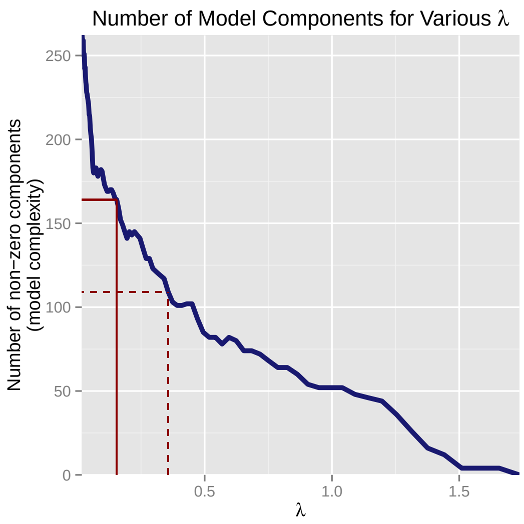


Figure 6.17: Number of components drops at λ increases. λ is effectively a penalty on model complexity.

Figure 6.18 shows λ (the logarithm of λ, for scale convenience) plotted against the cross-validation error for the same data and model. We see the minimum and the uncertainty associated with this value ($\pm\,1$ s.d. by convention [60]). The value chosen is the value to the right of the minimum (larger λ) whose error is equal to the upper bound of the minimum. Hence predictions would be made with $\log\lambda = -0.75$, or $\lambda = 0.47$, corresponding to a model of (non-zero) 92 parameters.

In certain cases the error rises only slowly and the uncertainty of the error is large. In this case, there will not be a $\lambda(\text{Err} + 1\,\text{s.d.})$ value — the error does simply not rise to this level within the range of sampled λ values. This implies that the highest λ value will be picked, which inevitably corresponds to a zero–parameter model. To deal with this, we use the minimum-error λ value in these cases.

6.3.5 One Parameter At a Time

The 8 thousand or so predictors are readily grouped. For example, the predictors concerning GC content could reasonably be grouped together, as could those identifying the top codon used within each amino acid. Therefore we start by

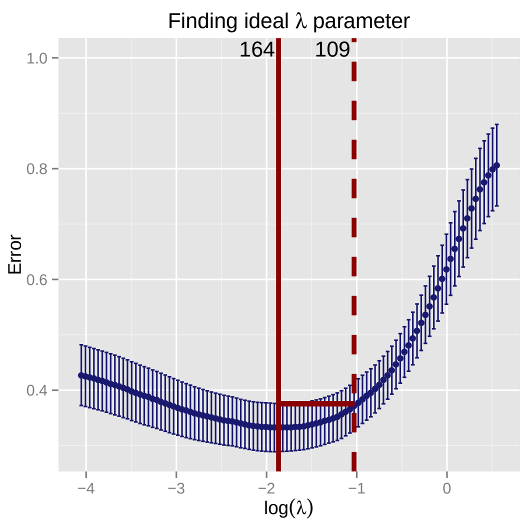


Figure 6.18: The cross-validation error falls with increasing regularisation until a minimum (solid line), at which point it increases as the model suffers from over-fitting. The λ for prediction is chosen to be within one standard deviation on the error of the minimum, in favour of a simpler model.

introducing parameters to the model in groups. This also provides an efficient way to get an overview of the more predictive codon measures.

Table A.1 contains a long table of model performances for small groups of parameters. Each data set is listed (A, B, C are from Welch, K is from Kudla and X is the total dataset). Each model is trained on a variety subsets of codons. To investigate the window effect, that codon preferences might be more important in certain parts of the sequence, each codon measure is calculated on various subsets of the gene, based on splits into three parts. In the table they are coded with a hyphen meaning that third is included and a dot meaning it is not:

Notation	Included
---	whole sequence
-··	first third only
·-·	middle third only
··-	final third only
·--	final two thirds only

The predictor groupings were:

- GC: GC1, GC2, GC3 and GCT.
- cai1: All codon adaptation indices (one standard and five calculated from ribosomal genes)
- cai2: All codon-pair adaptation indices (five calculated)
- tai: All tRNA adaptation indices (12 calculated from tRNA gene copy number tables)
- ai: All the three previous groups (any predictor with 'ai' in its name)
- freq: All frequency-based predictors, single and double.
- rscu: All RSCU-based predictors, single and double.
- eff: All count-based predictors, single and double.
- topC: Top codons per each amino acid.
- lowC: Lowest codon per each amino acid.
- succ: All measures related to successive pairs of top (and low) codon.
- contig: All measures related to lengths of successive (contiguous) top (and low) codons.
- sin: All single versions of freq, rscu and eff.
- dub: All double versions of freq, rscu and eff.
- top2: All succ and contig measures for top codons only.
- low2: All succ and contig measures for low codons only[1].

6.3.5.1 Summary of Single–Paramater-Group Models

Even though a model using each parameter set was trained on each dataset, many ended up with no parameters. This was often the case for the double codon measures. For example the parameter sets contig and top2 rarely resulted in a meaningful predictive model. (The predictor set low2 never resulted in a useful model, so was removed from all modellings, as noted earlier.) This is because codon pair features are exceedingly sparse. Many features do not occur, particularly in shorter sequences. Also, if they occur the same number of times

[1]The predictor group 'low2' did not produce any models with any non-zero parameters, hence this was excluded entirely early on.

we have a predictor with no variance, which can only ever produce a model identical to a simple intercept model.

Fewest meaningful models were found for dataset B. This is also the smallest dataset. But this cannot be the sole driver: dataset A was well-described by many models, and but their joint dataset, C, was considerably less well-described than A on its own. This, despite a doubling of observations. Clearly there is something inherently difficult to model in dataset B — and certainly very different from A.

The predictor sets noted with 0 means that for that dataset and that subsequence and for the given predictors, no useful models could be trained. B again is extreme in that only two sequence regions were found to give useful models. They are the full sequence ('ai' significant) and the middle third (tai significant).

As an aside, we can investigate how the parameters weights are shared in these two models. For the full-sequence model, a little under half of the parameters are set to zero. The non-zero parameters are: the standard CAI statistic and *all* of the tAI statistics. This suggests that the ridge has been working primarily on the tAI parameters (they were shrunk to a similar size) and the lasso worked primarily on the CAI parameters (all but one was set to zero).

For the middle-third–sequence model only one parameter is set to zero. This suggests either that the 12 tAI measures are all superbly independent (i.e. they were not eradicated by penalisation), or that the λ for the model was very low. We confirm that lambda was very small: its true value is $\lambda = 2.16 \times 10^{-3}$.

The variation in λ values says something about the level of correlation within each subset of predictors. Larger λ values is an indication of less dependence between predictors. This is more likely to be the case for small predictor sets, since there is more 'room' for them to be independent. The GC set is a good example of this: their λ values are consistently close to 1, whereas for other groups it is often lower.

Comparing the number of successful models per sequence window is interesting. We see that all the datasets from Welch, A, B and C, are best predicted by the middle third and last two thirds sequence segments. Of all the 47 models trained on those sets, only 3 are trained on the first third. (26 models were on some third-of-a-subset sequence.) This is in contrast to the Kudla set, which seems comparatively well-described by the first third of the sequence: 11 of its 40 models are on the first third. (23 models were on some third-of-a-sequence subset.) Clearly the first third sequence is meaningful for the Kudla data. This is also what was found in the article [2]; but it seems it is not true in general.

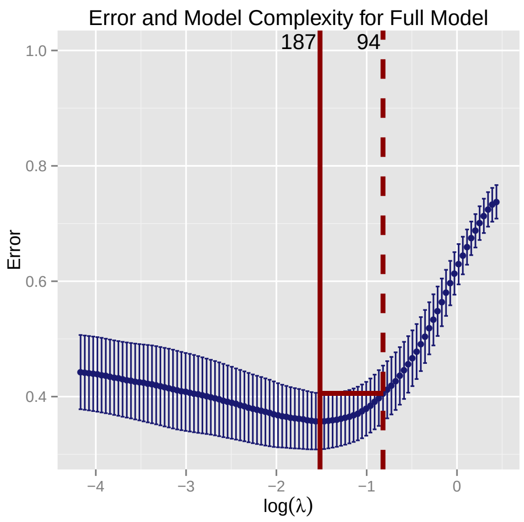


FIGURE 6.19: λ selection for the full model. The training set error minimum occurs with 187 parameters. In favour of parsimony, we choose a smaller model (to within one standard deviation on the errror). We choose $\lambda = 0.441$, corresponding to 94 parameters.

6.3.6 Combining Parameters for the Full Model

We are most interested in generating a general model. Ideally, it will be able to predict expression of any gene[2], not just for a gene that you have already expressed (i.e. not just one of the three specific genes we have here). Therefore we concentrate our efforts on training a model on the full dataset. This will also give more support for the model, since the combined dataset contains many more observations.

We combine all predictors for all sub-sequences datasets for observations from Welch and Kudla. This gives a model set-up with 163 observations and 42,615 predictors. The model is trained and found to give a minimum cross-validation error at $\lambda_{\text{min}} = 0.219$, see Fig. 6.19. Allowing for uncertainty in error and erring toward a simpler model, we settle for a prediction lambda of $\lambda_{1\text{sd}} = 0.441$. This brings the number of components from over 42 thousand to 94. Of the 94 significant parameters, it is interesting to note not only *what* they measure, but *where* they measure it. Tabulating this information, Table 6.2, shows that

[2]We are limited to making predictions for E. coli, since that is all we have data for. While generalisations may be possible, these datasets cannot support the predictive abilities of such a model to other organisms.

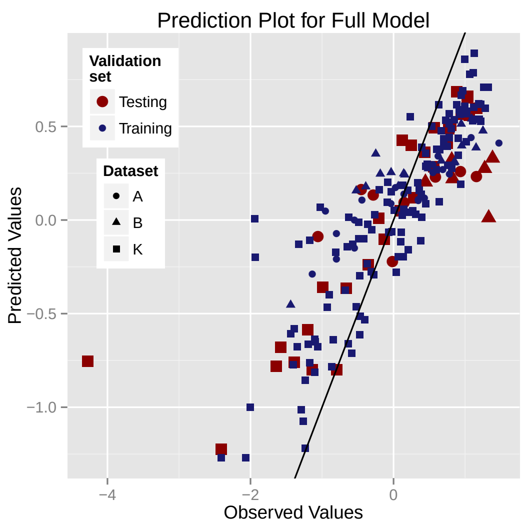


Figure 6.20: Validation plot on unseen test data. The red points are predicted based on the model trained on the blue points. The genes come from different datasets, as indicated by shape. There is one extremely outlying point in the test set, the red triangle to the lower left of the plot. It originates from the Kudla data. Setting this point aside, visually we have an extremely satisfactory fit. Objectively we tend to underestimate the more highly expressed genes and overestimate the lesser.

the first third is by far the most influential sub-sequence. We must weigh this against the fact that the Kudla data account for 77 % of the observations.

Table 6.2: Breakdown of the 94 parameters identified in Fig. 6.19 by which sequence segment they are based on.

Full	First $^1/_3$	Middle $^1/_3$	Last $^1/_3$	End $^2/_3$
---	-··	·-·	··-	·--
22	42	7	10	13

A more thorough discussion about the selected parameters will take place in Section 6.4.

6.3.7 Model Validation

Having chosen a model we can check how well it performs in predicting the test data we set aside at the beginning of the modelling process. Figure 6.20 looks good. The model appears to have projected the new points well within the envelope of the training points. This indicates that the model has found structure in the training data which is also present in the test data, and uses this to predict the response. The prediction correlation is $R^2 = 0.85$, which is very impressive. It is an improvement on the Welch-style analysis of the Kudla and Welch combined dataset in Fig. 6.13, which achieved an R^2 of 0.68.

6.4 Biological Interpretations

See Table A.2 for a print-out of the modelling parameters.

The purpose of this thesis is not necessarily to assign biological meaning to results that work. That being said, it is preferable if some justification for the model structure can be found. This is also one of the main advantages of the lasso technique: to clear out correlated variables and leave relevant ones behind that hopefully are easily interpretable.

Reverse-engineering a biological justification from a black-box model is fraught with difficulty and very open to confirmation fallacy. However, we will try to make some generalisations about the parameters identified in the final model, without giving *reasons why* things are like that.

Of all the predictors calculated on the first sequence third, there seems to be very many double-codon counts of low-GC content codons. Given that GC content is related to folding energy, this could be related to Kudla's result: that strong RNA folding in the first third of the gene correlates with strong expression.

It seems that GC content is the driving underlying factor in many of the predictors. Many of the codon (pairs) selected over the whole gene sequence are very low on GC bases. Given this, it seems strange that GC3 is only chosen in the last third sequence segment.

While 94 components is certainly more interpretable that several thousand, we are not able to assign meaning to these components. They do have a good prediction ability, but investigating the parameter values, Table A.2, we see that many predictors have very small parameters associated with them. A few have larger values. This could be a case of the ridge regression keeping some parameters from becoming exactly zero, and of the lasso keeping some parameters rather large.

6.5 Testing On New Data

In order to test the model fully, we have some data that is completely unrelated to the learning sets we have been using this far. The data is from a group at Novozymes A/S investigating the endogenous (intra-cellular) protein content of cells a various times during a fermentation process [63]. The cell is not Escherichia coli, but a Lactobacillus.

6.5.1 Data Description

There are five datasets. A sample was collected at five times through the fermentation process in order to track the genetic activity within the cell at various stages of life. Until now we have assumed a *steady state* — that the expression levels were taken in optimal conditions that are not time dependent, i.e. any change in protein expression was assumed to be a direct result of its codon sequence.

Our previous datasets consist of many replicates of the same gene with silent mutations. Thus we had many observations of the same gene (three genes in total). This dataset is very different. We have estimated protein levels within a single cell, for many different proteins. Thus we have no repeats for proteins.

Testing our model on this will present various challenges:

- We have no repeats. Any noise on the data points will used directly to predict, since we have nothing to average over.
- It is in a different organism. The model parameters may well not translate across to different organisms.
- There is a time-effect. We are not interested in this, so we will simply treat each dataset as independent. If one is very poorly predicted, it may indicate the organism was under stress at the point in time.

The response is not directly protein expression. It is a series of emPAI values [72]. They are known to correlate well with actual protein levels. Therefore we will take them as a good proxy for protein levels, and we will not discuss it further.

The sequence data is read from Nucleotide database at the National Center for Biotechnology Information [65]. This means we cannot be absolutely certain that the sequences are actually the ones that encoded the protein being expression. Mutations happen. But we will assume this effect is small. Again, we will not discuss this further.

6.5.2 Parameter Estimation

The data was read in using the same script as for the Welch and Kudla datasets. That being so, we calculated several parameters that need not have been calculated. However they were removed after parameter calculation and will not influence the analysis.

Each of the five time repeats was normalised and kept as independent response values for testing model performance. Sixth and seventh sets of responses were constructed by taking the means and sums of the first five, respectively. On close inspection of the responses, we saw that many of the emPAI values were zero. That caused problems with the scaling of the responses. We decided only to use the mean and sum responses. (Of course there are directly in proportion, so there should be no difference in predictive ability.)

6.5.3 Expression Prediction

The model from the Welch and Kudla datasets was used to predict expression values for the new data. The result is shown in Fig. 6.21.

We decided to try to train a model on the dataset, using the same predictors. These predictors were not defined with this kind of data in mind. We suspect that the model will perform poorly, since there are not many predictors that focus on the length of the gene. In a dataset with only one protein encoded multiple times, the length is constant. Hence it has not yet made sense to worry about length effects.

Figure 6.22 shows the model training for the Lactobacillus dataset. The model immediately ‘converges’ to an intercept-only model. That is to say, the predictors have no significance in estimating a protein expression level for a given gene (protein) in Lactobascillus.

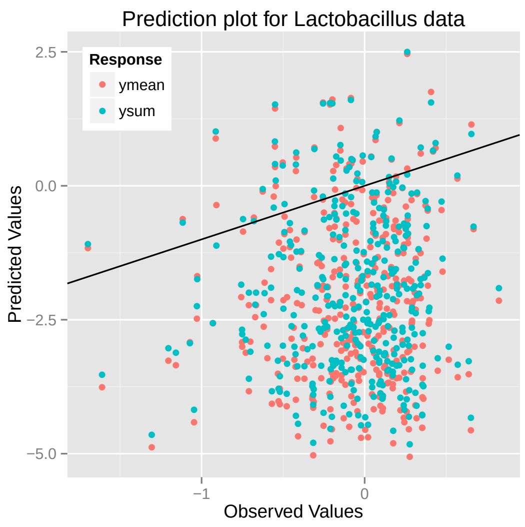


FIGURE 6.21: Prediction of Lactobacillus data using E. coli model. The prediction correlations for the mean and sum response are 0.037 and 0.033, respectively. This model has no predictive value.

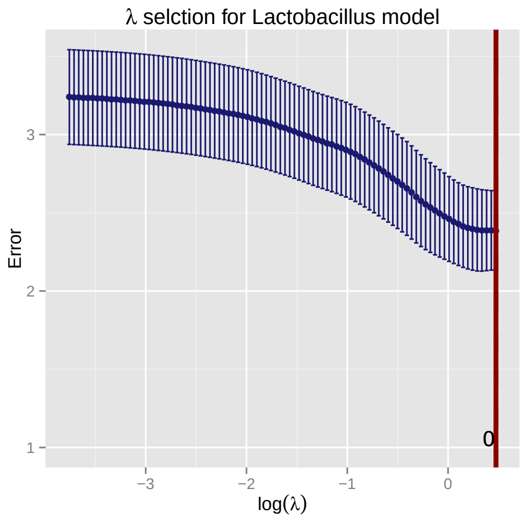


FIGURE 6.22: The model has minimum error at zero components (just intercept). This tells us we are more likely to get a good estimate by taking an average of previous measurements than we are by using this model. Clearly the predictors are deficient.

CHAPTER 7

The R Package: `codonTools`

In order to carry out thesis we have produced a number of R functions. They act as short-cuts for calculating codon statistics and enable new sequences to be analysed with ease.

The the functions are written as S4 classes which has quickly become the standard in bioinformatics. At the core of the functions is the `Org` class. This object stores sequence data along with other properties of the gene.

```
all.dat[[1]]


## Genetic sequence date (class 'Org')
## Sequence length:  720
## Number codons:    240
## Sequence type:    DNA
## Expression:       7673 flourescence
## Sequence label:   GFP_002
## Organism:         e.coli
## Belongs to:       Kudla
## Notes:            no


str(all.dat[[1]])
```

```
## Formal class 'Org' [package ".GlobalEnv"] with 8 slots
##   ..@ seq     : chr "atggttagtaagggtgaggaacttttcactggagttgtcccaattcttgttgaattagatg
##   ..@ label   : chr "GFP_002"
##   ..@ exp     : num 7673
##   ..@ exp.type: chr "flourescence"
##   ..@ belongs : chr "Kudla"
##   ..@ alph    : chr "DNA"
##   ..@ organism: chr "e.coli"
##   ..@ notes   : chr "no"
```

A full set of extractor functions exist. And on assigning a new `Org` content is checked to make sure it is valid. A modified `seQ` function allows the sequence to be extracted and formatted as either a vector of single characters, a single character string or split into codons.

Most functions are adapted so that they can accept either an `Org` object, a character string (in any of the three formats mentioned above) or a list of `Org`s (or characters). Output is appropriate for, for example, incorporation into a matrix of predictors.

Modelling functions from `glmnet` and `pls` are handled. And objects created by these functions are plotted nicely. Almost all the plots in this thesis have been made using the new functions. Also, plotting functions can accept a list of models and they figure is then facetted to show all the models in a neat way. This makes use of `ggplot2` [73].

The functions will be collected into a package called `codonTools` soon, with full R help documentation and package vignettes.

The source code for the thesis will be made available either online or otherwise.

CHAPTER 8

Conclusion

Bioinformatics is a fast-moving industry where machine learning is becoming a key tool. The amounts of data are enormous; it would be impossible to digest without efficient modelling techniques.

Preparing this thesis has entailed three parts. First, learning about the biology of the system. It is an exciting topic where it is commonly accepted that no one knows the full story. This makes it an excited area of work for a mathematician. Second, we have spent a good deal of time describing the mathematics implemented in the modelling. The aim of these two parts has been to ensure that the thesis is readable by, understandable by and useful to, not just applied mathematicians and computer scientists, but also biologists.

We set out to produce a model that did not rely heavily on pre-existing codon statistics, which are described to a large extent by their biological significance. We wanted to investigate whether sparse modelling techniques could identify structure in (relatively small) datasets consisting of synonymous genetic sequences and protein expression responses.

We achieved a prediction correlation of $R^2 = 0.85$ which compares very favourably to Welch et al.'s 0.68. We did not constrain ourselves by requiring a biological justification while defining predictors.

Unfortunately we were not able to get a completely fresh test set that suited our situation. This is certainly something we are keen to explore more.

8.1 Future Work

An interesting area to explore further would be how the model translates across to other organisms. We would ideally have similar data to the E. coli data that we have now, but for a variety of other organisms.

It would also we interesting to spend more time on the window-wise analyses, in particular looking for bottlenecks and their significance.

Other features that might be tested include off-frame statistics. This would be easy to implement given the R modelling structure we set up. But we feel we may have exhausted the data. The real limit here is a lack of data, which unfortunately is costly to make and has fairly limited use outside of ours.

The test we carried out toward the end of the project was not ideal for our model. However it would be interesting to develop a model for this purpose. This seems like a more natural way of predicting. On the one hand, you might predict entirely from a model and a set of sequences, in order to find the highest expression sequence. However often the model will perform better if trained a little on some data. In our current situation that would mean some examples of genes synonymously encoding the protein that you are interested in. It would be more general if the training data could come from any protein (from the same organism).

The scope for advanced machine learning applications in biotechnology is huge. The problem is that, to predict protein expressions, we currently need a quite specific sort of data. We must develop methods that can generate predictive models from a wider class of data.

Appendix A

Data Tables: Predictors and Parameter Values

A.1 Summaries of Single–Parameter-Set Models

Table A.1: Pertaining to Section 6.3.5. Model performances for various subsets of predictors (identified in column Predictor set). p is the number of non-zero parameters in the model for the given λ value. % of P is the percentage p/P, where P is the number of predictors in the current set. Err(CV) is a cross-validation error (as on the y-axis of, for example, Fig. 6.18 on page 78) and R^2 is the correlation of the true response with predictions of the data on itself using the model

Set	Window	Predictors set	λ	p	% of P	Err(CV)	R^2
A	---	GC	0.77	3.00	75.00	0.46	0.63
A	---	cai1	0.06	6.00	100.00	0.48	0.45
A	---	ai	0.06	17.00	73.91	0.42	0.62
A	---	freq	0.99	18.00	14.06	0.48	0.74
A	---	rscu	1.25	16.00	0.38	0.51	0.93
A	---	eff	1.25	16.00	0.38	0.51	0.93
A	---	sin	1.04	27.00	14.06	0.48	0.74
A	---	dub	1.14	29.00	0.35	0.50	0.93
A	-··	cai1	0.10	6.00	100.00	0.53	0.29
A	-··	ai	0.23	14.00	60.87	0.52	0.42

Table A.1: Pertaining to Section 6.3.5. Model performances for various subsets of predictors (identified in column Predictor set). p is the number of non-zero parameters in the model for the given λ value. % of P is the percentage p/P, where P is the number of predictors in the current set. Err(CV) is a cross-validation error (as on the y-axis of, for example, Fig. 6.18 on page 78) and R^2 is the correlation of the true response with predictions of the data on itself using the model

Set	Window	Predictors set	λ	p	% of P	Err(CV)	R^2
A	·–·	GC	0.98	2.00	50.00	0.51	0.58
A	·–·	cai1	0.22	5.00	83.33	0.50	0.37
A	·–·	ai	0.53	5.00	21.74	0.53	0.37
A	·–·	freq	0.84	20.00	15.62	0.47	0.78
A	·–·	rscu	1.56	5.00	0.12	0.52	0.84
A	·–·	eff	1.56	5.00	0.12	0.52	0.84
A	·–·	lowC	1.18	2.00	10.00	0.52	0.71
A	·–·	sin	0.88	24.00	12.50	0.48	0.77
A	·–·	dub	1.63	10.00	0.12	0.53	0.83
A	··–	cai1	0.07	6.00	100.00	0.52	0.41
A	··–	topC	0.04	17.00	85.00	0.19	0.98
A	·––	GC	0.86	2.00	50.00	0.51	0.54
A	·––	cai1	0.06	6.00	100.00	0.47	0.46
A	·––	topC	0.20	13.00	65.00	0.33	0.94
B	–––	ai	0.25	13.00	56.52	0.49	0.41
B	–··	0	0.00	0.00	0.00	0.00	0.00
B	·–·	tai	0.00	11.00	91.67	0.35	0.74
B	··–	0	0.00	0.00	0.00	0.00	0.00
B	·––	0	0.00	0.00	0.00	0.00	0.00
C	–––	GC	0.90	1.00	25.00	0.48	0.55
C	–––	rscu	1.21	10.00	0.24	0.52	0.74
C	–··	GC	1.10	1.00	25.00	0.53	0.42
C	·–·	tai	0.00	12.00	100.00	0.47	0.54
C	·–·	ai	0.13	11.00	47.83	0.51	0.37
C	·–·	rscu	0.55	66.00	1.59	0.41	0.92
C	·–·	eff	0.73	58.00	1.39	0.44	0.88
C	·–·	sin	0.69	22.00	11.46	0.52	0.74
C	·–·	dub	0.73	89.00	1.08	0.44	0.88
C	··–	GC	0.96	2.00	50.00	0.52	0.46
C	··–	rscu	1.14	11.00	0.26	0.52	0.68
C	··–	eff	0.98	40.00	0.96	0.51	0.78
C	··–	lowC	0.08	17.00	85.00	0.43	0.86
C	··–	dub	0.99	28.00	0.34	0.52	0.76
C	·––	GC	0.98	1.00	25.00	0.51	0.50
C	·––	rscu	0.73	34.00	0.82	0.47	0.88
C	·––	eff	0.53	84.00	2.02	0.45	0.92

Table A.1: Pertaining to Section 6.3.5. Model performances for various subsets of predictors (identified in column Predictor set). p is the number of non-zero parameters in the model for the given λ value. % of P is the percentage p/P, where P is the number of predictors in the current set. Err(CV) is a cross-validation error (as on the y-axis of, for example, Fig. 6.18 on page 78) and R^2 is the correlation of the true response with predictions of the data on itself using the model

Set	Window	Predictors set	λ	p	% of P	Err(CV)	R^2
C	·--	dub	0.70	49.00	0.59	0.47	0.88
K	---	cai1	0.00	5.00	83.33	0.72	0.38
K	---	tai	0.17	5.00	41.67	0.81	0.11
K	---	ai	0.03	8.00	34.78	0.72	0.41
K	---	freq	0.28	50.00	39.06	0.50	0.75
K	---	rscu	0.20	84.00	2.02	0.35	0.94
K	---	eff	0.20	83.00	2.00	0.35	0.94
K	---	sin	0.30	72.00	37.50	0.50	0.75
K	---	dub	0.19	146.00	1.77	0.36	0.95
K	-··	GC	1.17	2.00	50.00	0.74	0.48
K	-··	cai1	0.01	5.00	83.33	0.64	0.53
K	-··	ai	0.39	14.00	60.87	0.72	0.37
K	-··	freq	0.56	28.00	21.88	0.56	0.71
K	-··	rscu	0.39	50.00	1.20	0.39	0.90
K	-··	eff	0.39	51.00	1.23	0.39	0.90
K	-··	succ	1.28	3.00	75.00	0.75	0.47
K	-··	contig	1.28	2.00	50.00	0.76	0.47
K	-··	sin	0.56	42.00	21.88	0.56	0.71
K	-··	dub	0.39	83.00	1.01	0.40	0.91
K	-··	top2	0.88	3.00	100.00	0.70	0.47
K	·-·	rscu	1.05	7.00	0.17	0.77	0.47
K	·-·	eff	1.05	7.00	0.17	0.77	0.47
K	·-·	dub	1.10	6.00	0.07	0.77	0.46
K	··-	cai1	0.78	5.00	83.33	0.79	0.26
K	··-	cai2	0.98	3.00	60.00	0.80	0.30
K	··-	tai	0.00	10.00	83.33	0.75	0.37
K	··-	ai	0.39	4.00	17.39	0.75	0.29
K	··-	freq	1.28	10.00	7.81	0.77	0.46
K	··-	rscu	1.28	11.00	0.26	0.77	0.48
K	··-	eff	1.28	11.00	0.26	0.77	0.48
K	··-	sin	1.28	9.00	4.69	0.77	0.45
K	··-	dub	1.22	19.00	0.23	0.77	0.49
K	·--	GC	0.90	2.00	50.00	0.74	0.44
K	·--	cai1	0.00	5.00	83.33	0.74	0.31
K	·--	cai2	0.08	2.00	40.00	0.77	0.22
K	·--	ai	0.05	9.00	39.13	0.76	0.29

Table A.1: Pertaining to Section 6.3.5. Model performances for various subsets of predictors (identified in column Predictor set). p is the number of non-zero parameters in the model for the given λ value. % of P is the percentage p/P, where P is the number of predictors in the current set. Err(CV) is a cross-validation error (as on the y-axis of, for example, Fig. 6.18 on page 78) and R^2 is the correlation of the true response with predictions of the data on itself using the model

Set	Window	Predictors set	λ	p	% of P	Err(CV)	R^2
K	·--	freq	1.24	14.00	10.94	0.77	0.45
K	·--	rscu	1.16	12.00	0.29	0.76	0.50
K	·--	eff	1.16	13.00	0.31	0.76	0.50
K	·--	sin	1.24	15.00	7.81	0.77	0.45
K	·--	dub	1.21	19.00	0.23	0.77	0.50
X	---	cai1	0.00	6.00	100.00	0.71	0.28
X	---	ai	0.01	9.00	39.13	0.74	0.28
X	---	freq	0.17	30.00	23.44	0.50	0.72
X	---	rscu	0.47	49.00	1.18	0.46	0.83
X	---	eff	0.39	63.00	1.51	0.44	0.86
X	---	lowC	0.62	4.00	20.00	0.72	0.36
X	---	sin	0.13	50.00	26.04	0.48	0.75
X	---	dub	0.47	77.00	0.93	0.46	0.84
X	-··	GC	0.86	2.00	50.00	0.70	0.40
X	-··	freq	0.73	11.00	8.59	0.63	0.54
X	-··	rscu	0.48	37.00	0.89	0.46	0.81
X	-··	eff	0.48	38.00	0.91	0.45	0.82
X	-··	succ	0.82	2.00	50.00	0.70	0.38
X	-··	contig	0.82	1.00	25.00	0.70	0.38
X	-··	sin	0.73	17.00	8.85	0.63	0.55
X	-··	dub	0.51	52.00	0.63	0.46	0.81
X	-··	top2	0.82	2.00	66.67	0.70	0.38
X	·-·	GC	0.88	2.00	50.00	0.72	0.36
X	·-·	cai1	0.00	6.00	100.00	0.72	0.25
X	·-·	cai2	0.02	5.00	100.00	0.74	0.20
X	·-·	ai	0.56	11.00	47.83	0.72	0.24
X	·-·	freq	0.93	5.00	3.91	0.71	0.39
X	·-·	rscu	1.07	5.00	0.12	0.71	0.43
X	·-·	eff	1.06	6.00	0.14	0.71	0.43
X	·-·	topC	0.94	1.00	5.00	0.74	0.30
X	·-·	sin	1.02	6.00	3.12	0.72	0.38
X	·-·	dub	1.12	7.00	0.08	0.72	0.43
X	··-	GC	0.78	2.00	50.00	0.70	0.37
X	··-	freq	0.79	6.00	4.69	0.66	0.46
X	··-	rscu	1.03	8.00	0.19	0.70	0.47
X	··-	eff	0.99	9.00	0.22	0.69	0.48

TABLE A.1: Pertaining to Section 6.3.5. Model performances for various subsets of predictors (identified in column Predictor set). p is the number of non-zero parameters in the model for the given λ value. % of P is the percentage p/P, where P is the number of predictors in the current set. Err(CV) is a cross-validation error (as on the y-axis of, for example, Fig. 6.18 on page 78) and R^2 is the correlation of the true response with predictions of the data on itself using the model

Set	Window	Predictors set	λ	p	% of P	Err(CV)	R^2
X	··-	sin	0.91	4.00	2.08	0.67	0.44
X	··-	dub	1.04	8.00	0.10	0.70	0.48
X	·--	GC	0.79	2.00	50.00	0.67	0.45
X	·--	cai1	0.00	6.00	100.00	0.70	0.30
X	·--	ai	0.03	7.00	30.43	0.73	0.25
X	·--	freq	0.82	9.00	7.03	0.68	0.45
X	·--	rscu	1.12	7.00	0.17	0.71	0.44
X	·--	eff	1.06	12.00	0.29	0.71	0.44
X	·--	topC	0.64	5.00	25.00	0.70	0.43
X	·--	sin	0.88	12.00	6.25	0.68	0.45
X	·--	dub	1.12	8.00	0.10	0.72	0.43

A.2 Final Model Parameters

Table A.2: Final model parameter values

Predictor	Parameter value
(Intercept)	0.09
full_dub.eff.ttgaca	-0.00
full_dub.eff.ctgccc	-0.04
full_dub.eff.acgctg	0.00
full_dub.eff.atcggt	-0.00
full_dub.eff.gtagta	0.00
full_dub.eff.gttgta	0.03
full_dub.eff.ctagtc	-0.03
full_dub.eff.ggtgtt	0.02
full_dub.eff.gtagtt	-0.07
full_dub.eff.gatttt	-0.03
full_dub.rscu.gatttt	-0.00
full_dub.rscu.ctggac	-0.00
full_dub.rscu.cttaaa	0.00
full_dub.rscu.ctgccc	-0.01
full_dub.rscu.ttgaca	-0.00
full_dub.rscu.ctagtc	-0.00
full_dub.rscu.tttaaa	-0.02
full_dub.rscu.gtagta	0.00
full_dub.rscu.gttgta	0.00
full_dub.rscu.gtagtt	-0.00
full_lowC.E	-0.05
full_lowC.A	0.01
first_sin.freq.aca	1.86
first_sin.freq.agt	0.07
first_sin.freq.gtt	1.65
first_dub.freq.aca	0.02
first_dub.freq.gtt	0.01
first_sin.eff.aca	0.02
first_sin.eff.gtt	0.01
first_dub.eff.agcaag	-0.05
first_dub.eff.cacaag	-0.05
first_dub.eff.ttgaca	-0.00
first_dub.eff.ctgccc	-0.04
first_dub.eff.gaactt	0.04
first_dub.eff.ctggac	-0.04
first_dub.eff.gaagga	0.05
first_dub.eff.aagggg	-0.08

TABLE A.2: Final model parameter values

Predictor	Parameter value
first_dub.eff.atggta	0.01
first_dub.eff.gtagta	0.00
first_dub.eff.gttgta	0.03
first_dub.eff.atggtg	-0.09
first_dub.eff.gtagtt	-0.07
first_dub.eff.gaattg	-0.13
first_sin.rscu.aca	0.02
first_sin.rscu.agt	0.03
first_sin.rscu.gtt	0.00
first_dub.rscu.gaagga	0.01
first_dub.rscu.gaactt	0.01
first_dub.rscu.gaattg	-0.02
first_dub.rscu.cacaag	-0.01
first_dub.rscu.ctggac	-0.00
first_dub.rscu.cttaaa	0.00
first_dub.rscu.ctgccc	-0.00
first_dub.rscu.ttgaca	-0.00
first_dub.rscu.aagggg	-0.01
first_dub.rscu.atggta	0.00
first_dub.rscu.atggtg	-0.02
first_dub.rscu.agcaag	-0.00
first_dub.rscu.gtagta	0.00
first_dub.rscu.gttgta	0.00
first_dub.rscu.gtagtt	-0.00
first_top2succ.s	-0.06
first_top2succ.s	-0.06
first_topcontig.s	-0.06
mid_dub.eff.aagaat	-0.00
mid_dub.eff.tatatc	-0.06
mid_dub.eff.acacta	-0.01
mid_dub.rscu.gatttt	-0.00
mid_dub.rscu.aagaat	-0.00
mid_dub.rscu.acacta	-0.00
mid_topC.E	0.01
last_sin.freq.ggc	0.30
last_sin.freq.tca	-2.01
last_dub.freq.tca	-0.04
last_sin.eff.tca	-0.04
last_dub.eff.acgaaa	0.03
last_dub.eff.cagaag	-0.01
last_sin.rscu.tca	-0.00

TABLE A.2: Final model parameter values

Predictor	Parameter value
last_dub.rscu.cagaag	-0.00
last_dub.rscu.aagttc	0.00
last_dub.rscu.acgaaa	0.00
end_dub.eff.aagaat	-0.00
end_dub.eff.tatatc	-0.01
end_dub.eff.acacta	-0.01
end_dub.eff.ttggtg	-0.13
end_sin.rscu.gaa	-0.01
end_sin.rscu.gag	0.01
end_dub.rscu.gatttt	-0.00
end_dub.rscu.ttggtg	-0.00
end_dub.rscu.aagaat	-0.00
end_dub.rscu.acacta	-0.00
end_dub.rscu.gtcgag	0.00
end_GC3	0.29
end_topC.K	0.06

Bibliography

[1] Laura Beil. 'Unnatural Selection'. In: *Science News* 178.8 (Oct. 2010). URL: http://www.sciencenews.org/view/feature/id/63605/title/Unnatural__selection (cit. on p. 2).

[2] Grzegorz Kudla et al. 'Coding-Sequence Determinants of Gene Expression in Escherichia coli'. In: *Science* 314.5924 (Apr. 2009), pp. 255–258. DOI: 10.1126/science.1170160 (cit. on pp. 3, 20, 21, 29, 53, 64, 80).

[3] MA Santos et al. 'The Non-Standard Genetic Code of Candida spp.: an Evolving Genetic Code or a Novel Mechanism for Adaptation?' In: *Molecular Microbiology* 26 (3 Oct. 1997), pp. 423–431. DOI: 10.1046/j.1365-2958.1997.5891961.x (cit. on p. 3).

[4] Joshua B Plotkin and Grzegorz Kudla. 'Synonymous But Not the Same: the Causes and Consequences of Codon Bias'. In: *Nature* 12 (Jan. 2011), pp. 32–42. DOI: 10.1038/nrg2899x (cit. on p. 3).

[5] Mark Welch et al. 'Design Parameters to Control Synthetic Gene Expression in Escherichia coli'. In: *PLoS ONE* 4.9 (2009). DOI: 10.1371/journal.pone.0007002 (cit. on pp. 3, 23, 29, 59, 63).

[6] Frank Wright. 'The 'Effective Number of Codons' Used in a Gene'. In: *Gene* 87 (1990), pp. 23–29 (cit. on pp. 3, 26, 27).

[7] Frank Supek and Kristian Vlahoviček. 'Comparison of Codon Usage Measures and Their Ability in Prediction of Microbial Gene Expressivity'. In: *BMC Informatics* 6 (2005), p. 182. DOI: 10.1186/1471-2105-6-182 (cit. on p. 3).

[8] Jesse M Fox and Ivan Erill. 'Relative Codon Adaptation: A Generic Condon Bias Index for Prediction of Gene Expression'. In: *DNA Research* 17 (2010), pp. 185–196. DOI: 10.1093/dnares/dsq012 (cit. on pp. 3, 26).

[9] Josep M Comeron and Montserrat Aguadé. 'An Evaluation of Measures of Synonymous Codon Usage Bias'. In: *Journal of Molecular Evolution* 47 (1998), pp. 268–274. DOI: 10.1007/PL00006384 (cit. on p. 3).

[10] John Peden. *CodonW*. 1997. URL: http://codonw.sourceforge.net/ (visited on 12/09/2012) (cit. on pp. 3, 48).

[11] Antranik. *Organic Compound 3: Proteins*. 2012. URL: http://antranik.org/organic-compound-3-proteins/ (cit. on p. 6).

[12] Hans Sejr Olsen. *Enzymes at Work*. Novozymes A/S. Customer Communications. No. 2001-00269-04. 2004 (cit. on pp. 6, 17).

[13] David J Tenenbaum. 'Food vs. Fuel: Diversion of Crops Could Cause More Hunger'. In: *Environmental Health Perspectives* 116 (6 2008), A254–A257 (cit. on p. 6).

[14] R Bownen. *The Structure of Proteins*. 2002. URL: http://www.vivo.colostate.edu/hbooks/genetics/biotech/basics/prostruct.html (cit. on p. 9).

[15] Ken A Dill et al. 'The Protein Folding Problem: When Will It Be Solved?' In: *Current Opinion in Structural Biology* 17 (2007), pp. 342–346. DOI: 10.1016/j.sbi.2007.06.001 (cit. on p. 8).

[16] Wikipedia. *List of unsolved problems in chemistry — Wikipedia, The Free Encyclopedia*. 2012. URL: http://en.wikipedia.org/w/index.php?title=List_of_unsolved_problems_in_chemistry&oldid=496555994 (visited on 12/09/2012) (cit. on p. 8).

[17] Ingrid Wagner and Hans Musso. 'New Naturally Occuring Amino Acids'. In: *Angewendte Chemie* 22 (11 1983), pp. 816–828. DOI: 10.1002/anie.198308161 (cit. on p. 9).

[18] Linda Johansson, Guro Gafvelin and Elias SJ Arnér. 'Selenocysteine in Proteins — Properties and Biotechnological Use'. In: *Biochimica et Biophysica Acta* 1726 (1 2005), pp. 1–13. DOI: 10.1016/j.bbagen.2005.05.010 (cit. on p. 10).

[19] Joseph A Krzycki. 'The Direct Genetic Encoding of Pyrrolysine'. In: *Current Opinion in Microbiology* 8 (6 2005), pp. 706–712. DOI: 10.1016/j.mib.2005.10.009 (cit. on p. 10).

[20] Jianming Xie and Peter Schultz. 'Adding Amino Acids to the Genetic Repetoire'. In: *Current Opinion in Chemical Biology* 9 (2005), pp. 548–554 (cit. on p. 10).

[21] BBC. *KS3 Bitesize*. 2012. URL: http://www.bbc.co.uk/schools/ks3bitesize/science/organisms_behaviour_health/disease/revise2.shtml (cit. on p. 11).

[22] Molecular Station. *mRNA UTR Sructure Exon Intron Cap.* 2007. URL: http://www.molecularstation.com/molecular-biology-images/503-rna-pictures/66-mrna-utr-structure-exon-intron-cap.html (cit. on p. 11).

[23] Lubert Stryer. *Biochemistry.* 3rd ed. W. H. Freeman and Company, 1988. ISBN: 0716719207 (cit. on pp. 12, 15–17).

[24] Hana Kucera. *Translation.* 2010. URL: http://www.genomebc.ca/education/articles/translation/ (cit. on p. 12).

[25] RBSS Biology. *Codon Wheel.* 2012. URL: http://rbssbiology11ilos.wikispaces.com/Codon+Wheel (cit. on p. 14).

[26] B Alberts et al. *Molecular Biology of the Cell.* 4th ed. 2004 (cit. on p. 16).

[27] Claes Gustafsson et al. 'Engineering Genes for Predictable Protein Expression'. In: *Protein Exprssion and Purification* 83 (Mar. 2012), pp. 37–346. DOI: 10.1016/j.pep.2012.02.013 (cit. on p. 24).

[28] Mario dos Reis, Renos Savva and Lorenz Wernisch. 'Solving the Riddle of Codon Usage Preferences: a Test for Translational Selection'. In: *Nucleic Acids Research* 32.17 (2004), pp. 5036–5044. DOI: 10.1093/nar/gkh834 (cit. on pp. 24, 47, 48).

[29] Kurt Fredrick and Michael Ibba. 'How the Sequence of a Gene Can Tune Its Translation'. In: *Cell* 141 (Apr. 2010), pp. 227–229. DOI: 10.1016/j.cell.2010.03.033 (cit. on pp. 25, 53).

[30] Sivan Navon and Yitzhak Pilpel. 'The Role of Codon Selection in Regulation of Translation Efficiency Deduced from Synthetic Libraries'. In: *Genome Biology* 12.2 (Feb. 2011). DOI: 10.1186/gb-2011-12-2-r12 (cit. on pp. 25, 53).

[31] Alexander Roth, Maria Anisimova and Gina M Cannarozzi. 'Measuring codon usage bias'. In: *Codon Evolution. Mechanisms and Models.* Ed. by Gina M Cannarozzi and Adrian Schneider. Oxford University Press, 2012. Chap. 13. ISBN: 978–0–19–960116–5 (cit. on pp. 25, 28, 45, 47).

[32] Paul M Sharp and Wen-Hsiung Li. 'An Evolutionary Perspective on Synonymous Codon Usage in Unicellular Organisms'. In: *Journal of Molecular Evolution* 24 (1986), pp. 28–38 (cit. on pp. 26, 46).

[33] Paul M Sharp and Wen-Hsiung Li. 'The Codon Adaptation Index — A Measure of Directional Synonumous Codon Usage Bias, and Its Potential Applications'. In: *Nucleic Acids Research* 15 (1987), pp. 1281–1295 (cit. on pp. 26, 27, 46, 48, 49).

[34] Tshimichi Ikemura. 'Correlation Between the Abundance of Escherichia coli Transfer RNAs and the Occurence of the Respective Codons in its Protein Genes: A Proposal for a Synonymous Codon Choice that is Optimal for the E. coli Translational System'. In: *Journal of Molecular Biology* 151 (1981), pp. 389–409 (cit. on pp. 26, 27, 47).

[35] Bennetzen and Hall. 'Codon Selection in Yeast'. In: *Journal of Biological Chemistry* 257 (1982), pp. 3026–3031 (cit. on pp. 26, 47).

[36] Uttam Roymondal, Shibsankar Das and Satyabrata Sahoo. 'Predicting Gene Expression Level from Relative Codon Usage Bias: An Application to Escherichia coli Genome'. In: *DNA Research* 16 (2009), pp. 13–30. DOI: `10.1093/dnares/dsn029` (cit. on p. 26).

[37] D Charif and J R Lobry. 'SeqinR 1.0-2: a contributed package to the R project for statistical computing devoted to biological sequences retrieval and analysis'. In: *Structural approaches to sequence evolution: Molecules, networks, populations*. Ed. by U Bastolla et al. Biological and Medical Physics, Biomedical Engineering. New York: Springer Verlag, 2007, pp. 207–232. ISBN: 978-3-540-35305-8 (cit. on pp. 26, 48).

[38] Jeremy Schmutz et al. 'Qualist Assessment of the Human Genome Sequence'. In: *Nature* 429 (2004), pp. 365–368. DOI: `10.1038/nature02390` (cit. on p. 29).

[39] National Human Genome Research Institute. *An Overview of the Human Genome Project*. National Institute of Health. 2011. URL: `http://www.genome.gov/12011238` (visited on 27/09/2012) (cit. on p. 29).

[40] Department of Evolutionary Genetics. *The Neandertal Genome Project*. Max-Planck-Gesellschaft. URL: `http://www.eva.mpg.de/neandertal/` (cit. on p. 29).

[41] Ajit Varki. 'A Chimpanzee Genome Project Is a Biomedical Impreative'. In: *Genome Research* (2000), pp. 1065–1070. DOI: `10.1101/gr.10.8.1065` (cit. on p. 29).

[42] National Human Genome Research Institute. *The Human Connectome Project*. National Institute of Health. URL: `http://www.humanconnectomeproject.org/` (visited on 27/09/2012) (cit. on p. 29).

[43] Human Variome Project International Limited. *The Human Variome Project*. URL: `http://www.humanvariomeproject.org/` (visited on 27/09/2012) (cit. on p. 29).

[44] 1000 Genomes. *1000 Genomes: A Deep Catalog of Human Genetic Variation*. European Bioinformatics Institute. URL: `http://www.1000genomes.org/` (visited on 27/09/2012) (cit. on p. 29).

[45] Line Katrine Harder Clemmensen and Karl Sjöstrand. 'Computational Data Analysis'. DTU IMM Course 02582: lecture notes (cit. on pp. 30, 32).

[46] Trevor Hastie, Robert Tibshirani and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. Springer Science+Business Media, 2009. ISBN: 978-0-387-84857-0. DOI: `10.1007/b94608` (cit. on pp. 31, 37, 39–43, 60, 71).

[47] Frederick Mosteller et al. ‘Fitting Staight Lines By Eye’. In: *Exploring Data Tables, Trends, and Shapes.* Ed. by David C Hoaglin, Frederick Mosteller and John W Tukey. John Wiley & Sons, Inc., 1985. Chap. 6. ISBN: 0-471-09776-4 (cit. on p. 32).

[48] Bjarne Kjær Ersbøll and Knut Conradsen. ‘An Introduction to Statistics’. 2007 (cit. on p. 34).

[49] Arthur E Hoerl and Robert W Kennard. ‘Ridge Regression: Biased Estimation for Nonorthogonal Problems’. In: *Technometrics* 12 (1 Feb. 1970), pp. 55–67. DOI: `10.2307/1267351` (cit. on p. 36).

[50] Robert Tibshirani. ‘Regression Shrinkage and Selection via the Lasso’. In: *Journal of the Royal Statistical Society Series B (Methodological)* 58.1 (1996), pp. 267–288 (cit. on pp. 37, 39).

[51] Scott Shaobing Chen, David L Donoho and Michael A Saunders. ‘Atomic Decomposition by Basis Pursuit’. In: *Journal of the Royal Statistical Society Series B)* 58 (1996), pp. 267–288. DOI: `10.1137/S1064827596304010` (cit. on p. 38).

[52] Christopher M Bishop. *Pattern Recognition and Machine Learning.* Information Science and Statistics. Springer Science+Business Media, 2006. ISBN: 978-0387-31073-2 (cit. on pp. 38, 39, 41).

[53] Ildiko E Frank and Jerome H Friedman. ‘A Statistical View of Some Chemometrics Regression Tools’. In: *Technometrics* 35.2 (1993), pp. 109–135. DOI: `10.1080/00401706.1993.10485033` (cit. on p. 38).

[54] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective.* Massachusetts Institute of Technology, 2012. ISBN: 978-0-262-01802-9 (cit. on pp. 38–40).

[55] Wenjiang J. Fu. ‘Penalized Regressions: The Bridge Versus the Lasso’. In: *Journal of Computational and Graphical Studies* 7.3 (1998), pp. 397–416. DOI: `10.1080/10618600.1998.10474784` (cit. on p. 38).

[56] Leo Breiman. ‘Heuristics of Instability and Stabilization in Model Selection’. In: *Annals of Statistics* 24.6 (1996), pp. 2350–2383. DOI: `10.1214/aos/1032181158` (cit. on p. 38).

[57] Mark Schmidt. *Least Squares Optimization with L1–Norm Regularization.* 2005 (cit. on p. 39).

[58] Shirish Krishnaj Shevade and S Sathiya Keerthi. ‘A Simple and Efficient Algorithm For Gene Selection Using Sparse Logistic Regression’. In: *Bioinformatics* 19.17 (2003), pp. 2246–2253. DOI: `10.1093/bioinformatics/btg308` (cit. on p. 39).

[59] Hui Zou and Trevor Hastie. ‘Regularization and Variable Selection Via the Elestic Net’. In: *Journal of the Royal Statistical Society Series B* 67 (2005), pp. 301–320. DOI: `10.1111/j.1467-9868.2005.00503.x` (cit. on p. 39).

[60] Jerome H Friedman, Trevor Hastie and Robert Tibshirani. 'Regularization Paths for Generalized Linear Models via Coordinate Descent'. In: *Journal of Statistical Software* 33.1 (2010), pp. 1–22. URL: `http://www.jstatsoft.org/v33/i01/` (cit. on pp. 40, 71, 77).

[61] M dos Reis, L Wernisch and R Savva. 'Unexpected Correlations Between Gene Expression and Codon Usage Bias From Microarray Data For the Whole Escherichia coli K-12 Genome'. In: *Nucleic Acids Research* 31.23 (2003), pp. 6976–6985. DOI: `10.1093/nar/gkg897` (cit. on pp. 47, 48).

[62] S Kanaya et al. 'Studies Of Codon Usage and tRNA Genes Of 18 Unicellular Organisms and Quantification Of Bacillus subtilis tRNAs: Gene Expression Level and Species–Specific Diversity Of Codon Usage Based On Multivariate Analysis'. In: *Gene* 238.1 (1999). PMID: 10570992, pp. 143–155 (cit. on p. 47).

[63] Peter Bjarke Olsen. *Internal report (confidential).* Novozymes A/S, 2012 (cit. on pp. 48, 49, 84).

[64] Ronald Jansen, Harmen J Bussemaker and Mark Gerstein. 'Revisiting the Codon Adaptaion Indexn From a Whole–Genome Perspecive: Analyzing the Relationship Between Gene Expression and Codon Occurence In Yeast Using a Variety Of Models'. In: *Nucleic Acids Research* 631.8 (2003), pp. 2242–2251. DOI: `10.1093/nar/gkg306` (cit. on pp. 48, 49).

[65] NCBI. *The Nucleotide Database.* National Center for Biotechnology Information, U.S. National Library of Medicine. 2012. URL: `http://www.ncbi.nlm.nih.gov/nuccore` (cit. on pp. 49, 84).

[66] Paul M Sharp and Wen-Hsiung Li. 'Codon Usage In Regulatory Genes In Escherichia coli Does Not Reflect Selection For 'Rare' Codons'. In: *Nucleic Acids Research* 14.19 (1986). PMCID: PMC311793, pp. 7737–7749 (cit. on p. 53).

[67] Bjørn-Helge Mevik, Ron Wehrens and Kristian Hovde Liland. ***pls** Partial Least Squares and Principal Component Regression.* R package version 2.3-0. 2011. URL: `http://CRAN.R-project.org/package=pls` (cit. on p. 57).

[68] R Core Team. R*: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria, 2012. URL: `http://www.R-project.org/` (cit. on p. 71).

[69] Rahul Mazumder, Trevor Hastie and Jerome Friedman. ***sparsenet**: Fit Sparse Linear Regression Models Via Nonconvex Optimization.* R package version 1.0. 2012. URL: `http://CRAN.R-project.org/package=sparsenet` (cit. on p. 71).

[70] Rahul Mazumder, Jerome Friedman and Trevor Hastie. ***SparseNet**: Coordinate Descent with Non-Convex Penalties.* 2009. URL: `www.stanford.edu/~hastie/Papers/Sparsenet/paper_spnet_main.pdf` (cit. on p. 71).

[71] Trevor Hastie and Brad Efron. *`lars`: Least Angle Regression, Lasso and Forward Stagewise*. R package version 1.1. 2012. URL: `http://CRAN.R-project.org/package=lars` (cit. on p. 71).

[72] Ysushi Ishihama et al. 'Exponentially Modified Protein Abundance Index (emPAI) for Estimation of Absolute Protein Amount in Proteomics by the Number of Sequences Peptides per Protein'. In: *Molecular & Cellular Proteomics* 4 (9 2005), pp. 1265–1272 (cit. on p. 84).

[73] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN: 978-0-387-98140-6. URL: `http://had.co.nz/ggplot2/book` (cit. on p. 88).