Computational Analysis of T Cell Receptor Repertoire and Structure

Thomas P. Peacock

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Infection and Immunity

and

CoMPLEX

University College London

November, 2021

Abstract

The human adaptive immune system has evolved to provide a sophisticated response to a vast body of pathogenic microbes and toxic substances. The primary mediators of this response are T and B lymphocytes. Antigenic peptides presented at the surface of infected cells by major histocompatibility complex (MHC) molecules are recognised by T cell receptors (TCRs) with exceptional specificity. This specificity arises from the enormous diversity in TCR sequence and structure generated through an imprecise process of somatic gene recombination that takes place during T cell development.

Quantification of the TCR repertoire through the analysis of data produced by high-throughput RNA sequencing allows for a characterisation of the immune response to disease over time and between patients, and the development of methods for diagnosis and therapeutic design. The latest version of the software package Decombinator extracts and quantifies the TCR repertoire with improved accuracy and compatibility with complementary experimental protocols and external computational tools. The software has been extended for analysis of fragmented short-read data from single cells, comparing favourably with two alternative tools.

The development of cell-based therapeutics and vaccines is incomplete without an understanding of molecular level interactions. The breadth of TCR diversity and cross-reactivity presents a barrier for comprehensive structural resolution of the repertoire by traditional means. Computational modelling of TCR structures and TCR-pMHC complexes provides an efficient alternative. Four general-purpose protein-protein docking platforms were compared in their ability to accurately model TCR-pMHC complexes. Each platform was evaluated against an expanded benchmark of docking test cases and in the context of varying additional information about the binding interface.

Continual innovation in structural modelling techniques sets the stage for novel automated tools for TCR design. A prototype platform has been developed, integrating structural modelling and an optimisation routine, to engineer desirable features into TCR and TCR-pMHC complex models.

Impact Statement

This thesis provides a computational analysis of T cell receptor (TCR) sequence and structure. TCRs are responsible for the recognition of invasive antigens and for triggering downstream immune responses to fight off disease. A quantitative understanding of the TCR repertoire (set of TCRs) of individuals provides insight into the complexities of the immune system, allows us to develop methods for early diagnosis of disease, to design new therapeutics, and to predict patient outcomes.

The body of work that explores TCR structure in this thesis focuses on the prediction of binding modes between TCR structures and antigen structures in the form of peptides presented by major histocompatibility complex (pMHC) molecules. A benchmark of test cases has been extended and used to compare four popular protein-protein docking software platforms in the context of TCR-pMHC assembly. This work has been recently published, providing guidance for researchers about the most effective methods for modelling TCR-pMHC complexes in their own work.

A substantial section of this thesis is dedicated to describing the latest version of the Decombinator software for TCR sequence analysis. Decombinator has been designed to extract and quantify TCR repertoires from biological samples. This quantification allows us to compare the immune response between healthy individuals and patients with disease. Furthermore, we are able to track the immune response of individuals over time, assessing how they respond to treatment or how the disease progresses. The Decombinator software has been used in a wide variety of settings in numerous collaborative studies, including investigations into the immune response to tuberculosis, HIV, and a number of different cancers.

Over the course of this PhD project, the latest Decombinator release has been published in two academic journals. The first provides all the details necessary to carry out the experimental protocol for affordable TCR sequencing, and the complementary computational pipeline for TCR analysis. The second provides details on the latest features of the computational package. This newest version of the software has been used in collaborative studies to investigate the dynamics of the TCR repertoire in individuals being treated for alopecia areata, and in response to stem cell transplantation for multiple myeloma.

The latest version of Decombinator is the most accurate to date for TCR quantification. The software has been upgraded to support a more robust experimental protocol, while remaining backwards compatible, broadening its useability. The software has been brought in line with the International Adaptive Immune Receptor Repertoire Community guidelines, allowing it to be seemlessly integrated with a large body of other immunological tools. Decombinator is free to use, open source, and complemented by a large number of tools that have been designed and extended to support analyses. The software is provided with extensive documentation for new users. During the course of this project, the author of this thesis ran a workshop to guide interested users through installation and example usage of the software. The workshop was well attended by PhD students, postgraduate researchers, and group leaders, from a number of academic groups at University College London.

Acknowledgements

I will be forever thankful for the tutelage and supervision of Professor Benny Chain. I could not have wished for a supervisor of greater patience, encouragement and understanding. He has lived up to my every expectation of an exceptional scientist and teacher, and continues to inspire me.

I also express my gratitude to all the members of the Chain lab, past and present, with whom I have collaborated. I similarly thank Professor Mahdad Noursadeghi and his research group who have broadened my scientific understanding and critique.

My thanks to all those who work within the Division of Infection and Immunity to create a stimulating academic environment. In particular, I would like to thank Professor Hans Stauss for his support and assistance in organising funding via the Royal Free Charity. In addition, I am truly grateful to those at CoMPLEX who provided me with the platform to undertake my postgraduate research. In particular, my thanks to Guy Moss and Lewis Griffin for the confidence they have shown in me and their assistance with funding.

My sincere thanks to my parents, and to my brother and sister, for their belief, encouragement and patience. Thank you to Evren for his emotional support and who helps me to see the good in the world. Thank you to Kaliane, who has celebrated every small victory and instilled me with self-belief. Thank you to Joel, who has been by my side all along, and without whom I would never have embarked on this journey. Thank you to the rest of my family and friends for their love and support. And finally, thank you to those few unnamed yet cherished associates who have provided me with the seed of ideas and a vault of memories.

Contents

List of Figures

List of Tables

Glossary

β_2M	β_2-microglobulin
AIR	Ambiguous Interaction Restraint
AIRR	International Adaptive Immune Receptor Repertoire
ANARCI	Antigen receptor Numbering And Receptor ClassificatIon software
ANM	Anisotropic network model
APC	Antigen presenting cell
ATLAS	The Altered TCR Ligand Affinities and Structures database
BCR	B cell receptor
BLOSUM	Blocks Substitution Matrix
C_α	TCR α chain constant gene
C_β	TCR β chain constant gene
CAPRI	The Critical Assessment of PRediction of Interactions experiments
CASP	The Critical Assessment of protein Structure Prediction experiments
CDR	Complementarity Determining Region
CoM	Centre of Mass
CPU	Central Processing Unit
cryo-EM	Cryo-electron microscopy
D_β	TCR β chain diversity gene
DCR	Decombinator 5-part classifier
DFIRE	Distance-scale Finite Ideal-gas REference scoring function
DNA	Deoxyribonucleic acid
DOPE	Discrete optimized protein energy

EGI	European Grid Initiative	**LRMSD**	Ligand RMSD
F_{nat}	Fraction of native contacts	**LYRA**	The LYmphocyte Receptor Automated modelling web server
$F_{non\text{-}nat}$	Fraction of non-native contacts	**MD**	Molecular dynamics
FCC	Fraction of common contacts	**MHC**	Major Histocompatibility Complex
FFT	Fast Fourier Transform	**ML**	Machine learning
GA	Genetic algorithm	**MMM**	Materials and molecular modelling
GPU	Graphics processing unit	**mRNA**	Messenger ribonucleic acid
GSO	Glowworm Swarm Optimisation	**MSA**	Multiple sequence alignment
HLA	Human leukocyte antigen	**NART**	Neoantigen-reactive T cell
HPC	High performance computing	**NGS**	Next-generation sequencing
HTS	High-throughput sequencing	**NMR**	Nuclear magnetic resonance
HV	Hypervariable	**NSCLC**	Non-small-cell lung cancer
IDI	Internal dual index	**PCR**	Polymerase chain reaction
iL-RMSD	Interface ligand RMSD	**PDB**	Protein Data Bank
IMGT	The international ImMunoGeneTics information system	**PEARS**	Position dEpendent Antibody Rotamer Swapper software
IRMSD	Interface RMSD	**pMHC**	Complex of peptide presented by MHC
J_α	TCR α chain joining gene	**RMSD**	Root-mean-square deviation
J_β	TCR β chain joining gene	**RNA**	Ribonucleic acid

SARS-CoV-2	Severe acute respiratory syndrome coronavirus 2	**TCR3d**	The TCR3d database
SAXS	Small angle x-ray scattering	**UCL**	University College London
SP1	Illumina sequencing primer 1	**UDI**	Unique dual index
SP2	Illumina sequencing primer 2	**UMI**	Unique molecular identifier
STCRDab	The Structural TCR Database	**UTR**	Untranslated region
TCR	T cell receptor	**V$_\alpha$**	TCR α chain variable gene
TCR-pMHC	complex of TCR and pMHC	**V$_\beta$**	TCR β chain variable gene
TCRα	TCR α chain	**VDJ**	Junction formed by the rearrangement of TCRβ V, D and J genes
TCRβ	TCR β chain	**VJ**	Junction formed by the rearrangement of TCRα V and J genes

Chapter 1

Introduction

1.1 The Innate and Adaptive Immune Systems

The human immune system has evolved to protect the body from a rich world of pathogenic microbes and toxic substances. Pathogens that threaten to disrupt normal homeostasis replicate, spread and damage the host through a broad range of mechanisms. Tasked with detecting and eliminating these pathogens without excessively damaging self-tissue or commensal microbe populations, the immune system has developed into a highly complex network of biological processes (Chaplin, 2010).

The mechanisms of the immune response are generally categorised as pertaining to two immune subsystems. The first of these subsystems, the innate immune system, is assumed to have arisen over 600 million years ago, and elements of innate immunity can be found in all multicellular organisms (Smith *et al.*, 2019). The innate system responds rapidly to infection, and is triggered within minutes or hours of pathogenic invasion (Marshall *et al.*, 2018). The various components of the innate immune system are diverse and include physical barriers such as the skin, secreted mucus layers and epithelial cilia (as in the respiratory and gastrointenstinal tracts), humoral components (such as cytokines, antimicrobial peptides, complement serum proteins, and a number of other soluble proteins) and cell-mediated mechanisms (such as those of phagocytic and non-specific cytotoxic cells) (Romo *et al.*, 2016; Smith *et al.*, 2019; Chaplin, 2010).

The development of the second subsystem, the adaptive immune system, is a more recent event in the history of verterbrate evolution. Specific components of the adaptive immune system, including immunoglobulins, T cell receptors (TCRs) and major histocompatibility complex (MHC), are believed to have arisen approximately 450 million years ago in the first jawed vertebrates (Smith *et al.*, 2019; Flajnik and Kasahara, 2010; Brazeau and Friedman, 2015; Buchmann, 2014). Through an imprecise process of somatic gene recombination, the cells that comprise the adaptive immune response achieve extraordinary specificity for a broad array of pathogens, toxins and allergens. Upon encountering a foreign antigen, these cells proliferate to attain sufficient numbers to generate effective protection against the infection (Chaplin, 2010). Consequently, the adaptive response generally activates more slowly than the innate response. A subset of the expanded cell population persists after infection in a dormant state (Ratajczak *et al.*, 2018). This process provides the host with an immunological memory, whereby a rapid and effective response can be mounted against a specific pathogen upon reinfection, and provides the basis for immunisation (Marshall *et al.*, 2018).

Like the innate immune system, the adaptive immune system is composed of both humoral and cell-mediated components. Humoral mechanisms, including the production of high affinity immunoglobulins and cell surface presentation of antigen for T cell activation, are orchestrated by B cells (Smith *et al.*, 2019). Cell-mediated immune response activity is coordinated by T cells. TCRs, which sit on the surface of T cells and provide them with their exquisite specificity, are the primary mediators of T cell activation (Pennock *et al.*, 2013) and the main focus of this thesis.

1.2 Computational Approaches in Immunology

Over the last few decades, the scientific community has witnessed an explosion of biological data. Breakthroughs in sequencing technology and advancements in computing power and storage capabilities have seen the fields of bioinformatics and computational biology firmly cemented at the forefront of modern science. So much of the biological research conducted today relies on some form of quantitative analysis, that it has been argued that we now live in an era where all biology is computational biology (Markowetz, 2017).

The world of bioinformatics encompasses a broad array of computational approaches used to address biological questions across all scales. At the molecular level, modelling the behaviour, activity and interaction of proteins and small compounds are of crucial importance for drug discovery (Sliwoski *et al.*, 2014) and in rational protein design (Tiwari *et al.*, 2012). At the population level, computational methods are used to disentangle the underlying genetic basis of common diseases (Moore *et al.*, 2010).

Attempts to make sense of the accumulation of vast quantities of immunological data have seen the emergence of the immunoinformatics (or computational immunology) branch of bioinformatics. Integration of functional, clinical and epidemiological data generated using high-throughput experimental techniques are used to probe the dynamics of the immune system in the context of health and disease (Tomar and De, 2014). Research presented in this thesis includes the development of computational tools, approaches, and resources for the study of T cell receptors (TCRs) at different scales. The following sections describe TCRs in terms of their sequence and structure, and provides a background on the computational approaches used to investigate them for application in medicine.

1.3 The T Cell Receptor Repertoire

1.3.1 TCR Recognition and Function

T cell receptors (TCRs) are glycosolated heterodimers that are expressed on the surface of T lymphocytes (T cells). The majority of TCRs (approximately 95%) are composed of an α chain and a β chain, and are responsible for the recognition of antigenic peptide fragments presented at the surface of antigen presenting cells (APCs) by major histocompatibility complex (MHC) molecules. A small proportion of TCRs (approximately 5%) are composed of a γ chain and a δ chain, and may recognise free peptides (Glusman *et al.*, 2001).

The T cell mediated immune response is driven by two broad T cell subpopu-

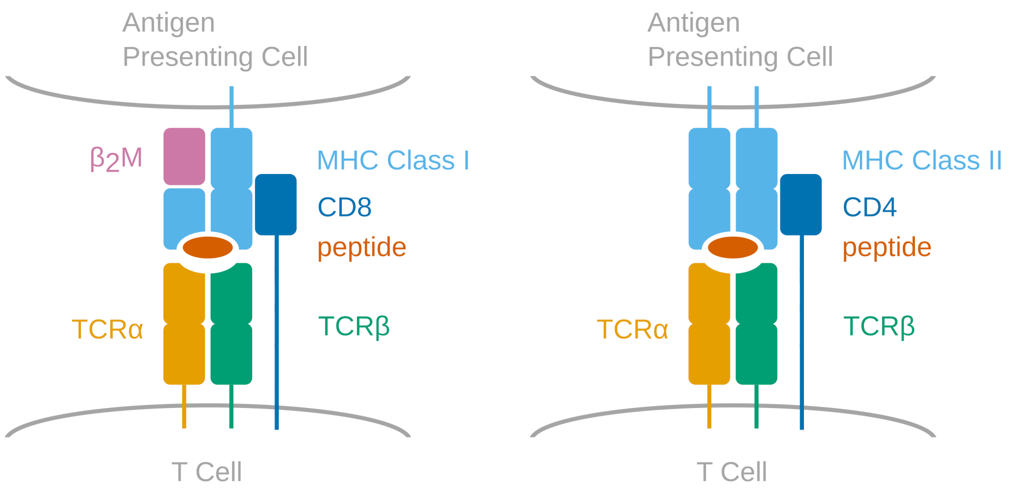


Figure 1.1: (*left*) An $\alpha\beta$ TCR expressed on the surface of a CD8 cytotoxic T cell, supported by a CD8 coreceptor, recognises an antigenic peptide presented by a MHC class I molecule at the surface of an antigen presenting cell. The MHC class I molecule is stabilised by the non-covalently bonded β_2-microglobulin (β_2M) subunit. (*right*) An $\alpha\beta$ TCR expressed on the surface of a CD4 helper T cell, supported by a CD4 coreceptor, recognises an antigenic peptide presented by a MHC class II molecule at the surface of an antigen presenting cell.

lations. CD8 “cytotoxic” (or “killer”) T cells express TCRs that recognise peptides presented by MHC class I molecules. A cytotoxic T cell also expresses the CD8 glycoprotein coreceptor at its surface, which binds the T cell to the MHC molecule for the duration of T cell activation. Upon recognition of antigen, CD8 cytotoxic T cells most commonly promote apoptosis through the delivery of cytotoxic granules into the cytosol of the infected or dysfunctional cell.

CD4 “helper” T cells, meanwhile, express TCRs that recognise peptides presented by MHC class II molecules. A helper T cell also expresses the CD4 glycoprotein coreceptor at its surface, which binds the T cell to the MHC molecule for the duration of T cell activation. Upon antigen recognition, CD4 helper T cells provide indirect mechanisms for controlling pathogens primarily though the generation of cytokines that activate neighbouring cells, or chemokines that recruit new immune cell subsets to the site of infection (Pennock *et al.*, 2013). Schematics of CD4 and CD8 T cell activation are illustrated in Figure 1.1.

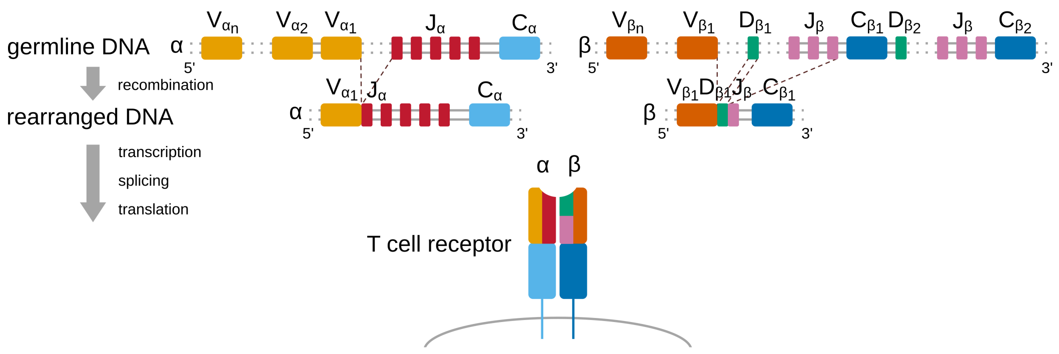


Figure 1.2: The somatic V(D)J recombination process is shown for the TCR α and β chains. (*left*) The top row shows a V_α gene segment rearranging to a J_α gene segment, to form the the VJ_α junction in the second row. Transcription and splicing of the VJ_α exon to the C_α gene segment, followed by translation of the generated mRNA, produces the TCR α chain displayed at the cell surface in the third row. (*right*) The top row shows the rearrangement of a V_β gene segment, a D_β gene segment and a J_β gene segment to form the VDJ_β junction in the second row. Transcription and splicing of the VDJ_β exon to a C_β gene segment, followed by translation of the generated mRNA, produces the TCR β chain displayed, paired with the α chain, at the cell surface in the third row. This figure is a redesign of that which features in Murphy *et al.* (2008).

1.3.2 Somatic Recombination and TCR Diversity

TCRs are formed through the somatic recombination of gene segments during T cell development in the thymus. V (variable) and J (joining) gene segments are found at non-adjacent locations in the TCRα locus. Similarly, V, J and additional D (diversity) gene segments are found at non-adjacent locations in the TCRβ locus. During the recombination process, extraneous DNA between gene segments is excised, such that junctions are formed between a V_α gene segment and a J_α gene segment for the α chain, and between a V_β gene segment, a D_β gene segment and a J_β gene segment for the β chain. The rearrangement is imperfect, and a random number of non-templated nucleotides are inserted and deleted at each of the VJ (TCRα) and the VDJ (TCRβ) junctions. Once transcribed, the rearranged VJ section of the α chain is spliced onto the C_α constant gene segment, and the rearranged VDJ section of the β chain is spliced onto one of two (homologous and functionally indistinct) C_β constant segments (Murphy *et al.*, 2008). The recombination process is illustrated in Figure 1.2.

The rearrangement of gene segments generates an extraordinarily diverse set of TCRs. The number of configurations made available by combining α and β chains, each of which is composed of one of a number of V, D and J genes, exceeds 2.8 million unique TCRs. Naïvely assuming that up to 10 insertions or deletions are permitted at each junction sees this number climb to in excess of 10^{16} TCRs. More sophisticated probabilistic estimates have placed the number as over 10^{20} (Zarnitsyna *et al.*, 2013), and even as high as 10^{61}, (Mora and Walczak, 2019) possible configurations. These numbers dwarf the total number of cells in the human body (Sender *et al.*, 2016). Consequently, this theoretical diversity is limited in a single individual by their T cell population size, which is thought to be on the order of 10^{11} in humans and 10^{8} in mice (Mandl and Germain, 2014). Nevertheless, the extraordinary breadth of the TCR repertoire allows the immune system to respond effectively to an equally broad array of invading pathogens.

1.3.3 Analysis of the TCR Repertoire

The diversity introduced by recombination has made the analysis of TCR repertoire a daunting challenge. Historically, TCR repertoire screening was predominantly performed through the application of flow cytometry and CDR3 spectratyping (Fozza *et al.*, 2017). Flow cytometry allows for the screening of millions of cells to provide quantitative information on the frequency of lymphocyte gene expression. Clonal expansion (proliferation) in the TCR repertoire can be used to identify specific T cell responses to disease. A fast and relatively inexpensive technology (Faint *et al.*, 1999), the application of flow cytometry to TCR analysis is limited by its inability to profile junction diversity, and by the availability of monoclonal antibodies specific for TCR V_β genes (Six *et al.*, 2013).

The recognition of antigen by TCRs is driven by the three complementarity determining region (CDR) loops of each chain. The majority of TCR diversity is found in the CDR3 loop which spans the V(D)J junction, and the insertion and deletion of nucleotides at the junctional sites results in CDR3 regions that vary in length. The CDR3 spectratyping method involves the amplification of CDR3 fragments followed by gel electrophoresis. The fragments are sieved through a sequencing gel via the application of an electric field. The shorter the fragment, the faster and further it migrates through the gel. Therefore, patterns can be identified in the CDR3 length distribution of the TCR repertoire. Comparison of CDR3 profiles can be used to establish the degree of clonal expansion in the repertoire at a greater resolution than flow cytometry (Fozza *et al.*, 2017).

Early attempts to capture TCR diversity at the nucleotide level were based on molecular cloning and Sanger sequencing (Sant'Angelo *et al.*, 1998; Correia-Neves *et al.*, 2001), but could only provide a limited description of the TCR repertoire (De Simone *et al.*, 2018). Since those pioneering studies, the scientific community has witnessed dramatic breakthroughs in DNA sequencing technology (Voelkerding *et al.*, 2009; McGinn and Gut, 2013; Heather and Chain, 2016). Coupled with major advances in computing power and storage, massively parallel sequencing of millions of DNA or RNA molecules provides means for the generation of the (increasingly) large datasets required for meaningful quantitative analysis of TCR diversity (De Simone *et al.*, 2018).

As the cost of next-generation sequencing (NGS) machines continues to decrease, an increasing need has arisen for bioinformatics pipelines (workflows) that are efficient, scalable, and simple to use (Fjukstad and Bongo, 2017). The development of such pipelines and the consolidation of wet lab protocols means that it is now a routine exercise in immunoinformatics to extract full α and β chain rearrangements of every clone in a sample for plates of tens of samples at a time. A comprehensive comparative review of twelve of the most widely used computational tools for TCR repertoire analysis has been undertaken by Zhang *et al.* (2020), and highlights the rapidity at which the field has grown.

One of the twelve compared pieces of software is the Decombinator platform — a suite of tools created by the Chain lab at University College London. The Decombinator software makes novel use of an efficient pattern-matching search algorithm to annotate TCR sequences efficiently in raw sequence data. The facilities of this computational pipeline have grown in response to the increasing flow of raw biological data produced through various protocols and encoded in various formats. The most up-to-date description of Decombinator has been recently published in Peacock *et al.* (2021), reflecting work achieved during the course of this PhD project.

Methods and applications of the Decombinator platform are described in Chapter 2. The core components and principles of Decombinator are discussed to provide a background for improvements made to the pipeline as part of this project. Additionally, a number of tools that have been developed and upgraded for use with Decombinator are outlined.

The Decombinator pipeline was designed with the bulk sequencing of pooled immune cell populations in mind. From an immunological point of view, bulk sequencing is limited by the inability to pair the sequenced α and β chain of the TCR, and consequently the inability to capture the full biological function of a T cell *in vivo* (De Simone *et al.*, 2018). The feasibility of solving this problem has recently garnered considerable interest from the research community, as single cell sequencing platforms have become more readily available (Hwang *et al.*, 2018).

The basis of the novel tag-based approach used by Decombinator relies on a minimum read length to annotate both the V and J genes of the TCR successfully. This makes the pipeline — and indeed, most other TCR sequencing pipelines — unsuitable for most single cell analyses, where reads are typically short and fragmented. Chapter 2 describes an adapted version of Decombinator for use with raw single cell data. Finally, a perspective is offered on the scope of future development for the platform.

1.4 T Cell Receptor Structure

1.4.1 TCR Domains and Geometry

TCR heterodimers are composed of variable and constant domains, a transmembrane domain and a short cytoplasmic tail (Rudolph *et al.*, 2006). The TCR variable domains can be divided into framework regions and complementarity determining regions (CDRs). The CDR regions are found as six hypervariable loops (three per TCR chain), and are understood to be the driving force behind TCR recognition of

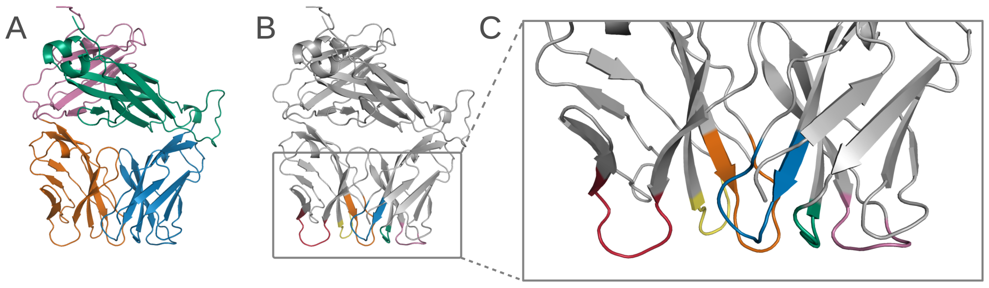


Figure 1.3: (A) The crystallised structure of the 2C TCR (PDB code 1TCR) is shown with the V_α domain highlighted in orange, V_β domain in blue, C_α domain in rose, and C_β domain in green. (B) The same TCR is shown with the α chain CDR1, CDR2, and CDR3 loops highlighted in yellow, red, and orange, respectively, and the β chain CDR1, CDR2, and CDR3 loops highlighted in green, rose, and blue, respectively. (C) A close up of the CDR loops is provided, with identical highlighting.

antigen, with the framework regions serving as structural support. An example TCR structure with highlighted domains and CDR loop regions is shown in Figure 1.3.

Analysis of antibody structures has shown that antigen binding is affected by the relative orientation of variable domains (Abhinandan and Martin, 2010; Kuroda *et al.*, 2012). Recent computational work has shown a similar phenomenon for TCR structures, with CDR3 loop conformation strongly influenced by the interface between the V_α and V_β domains (Fernández-Quintero *et al.*, 2020). The relative orientation between TCR variable domains has been previously characterised by six parameters (the "TRangles"), following an approach first used to describe antibody geometry (Dunbar *et al.*, 2014).

1.4.2 MHC Class I and Class II Architecture

MHC class I molecules are heterodimers composed of a heavy α chain covalently bonded to a light β_2-microglobulin (β_2M) chain. The groove in which the peptide is presented is constructed from only the heavy chain. MHC class II molecules are also heterodimers, but are composed of two heavy chains (α and β). The peptide groove is constructed from both heavy chains, such that, despite their differences in composition, the two MHC class molecules share a similar resultant architecture (Rudolph *et al.*, 2006). Examples of MHC class I and class II molecules are

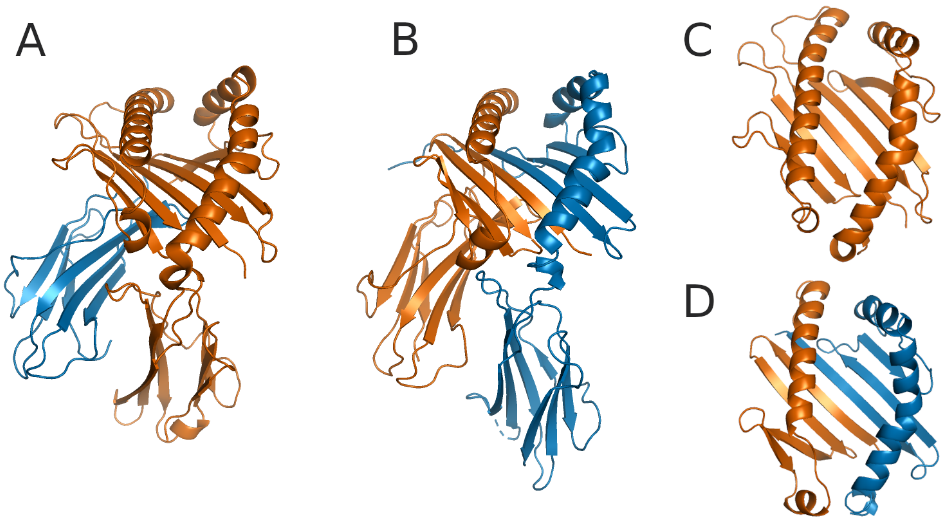


Figure 1.4: (A) The HLA-A2 MHC class I structure with heavy chain shown in orange and light β_2M chain in blue. (B) The HLA-DR1 MHC class II structure is shown with the two heavy chains in orange and blue. (C) View of the HLA-A2 MHC class I peptide groove from above, composed of only the heavy chain, shown in orange. (D) View of the HLA-DR1 MHC class II peptide groove from above, with the two heavy chains shown in orange and blue. The MHC class I and class II structures correspond to structures in the PDB under codes 1DUZ and 1KLG, respectively, with the peptides artificially removed for visualisation.

compared in Figure 1.4.

1.4.3 Presentation and the Peptide Groove

For both MHC classes, the presented peptide sits above a β-sheet and is bordered on each side by an α-helix wall. Peptides presented by MHC class I are anchored by their termini, and are therefore somewhat restricted in their length. The majority of MHC class I peptides are between 8 and 10 amino acids long. Nevertheless, MHC class I has been shown to bind longer peptides, either by extending at the C terminus or by bulging outwards from the binding groove (Rudolph *et al.*, 2006; Burrows *et al.*, 2006), thereby providing additional surface area for recognition. Their fixed termini and bulging from the groove mean that TCR binding usually occurs around the central residues of the peptide (Singh *et al.*, 2020). In contrast, the MHC class II peptide groove is open at both ends. Peptides presented by this groove therefore tend to be longer than those presented by MHC class I, typically between 13 and 25 amino acids in length (Wieczorek *et al.*, 2017). These peptides do not bulge away

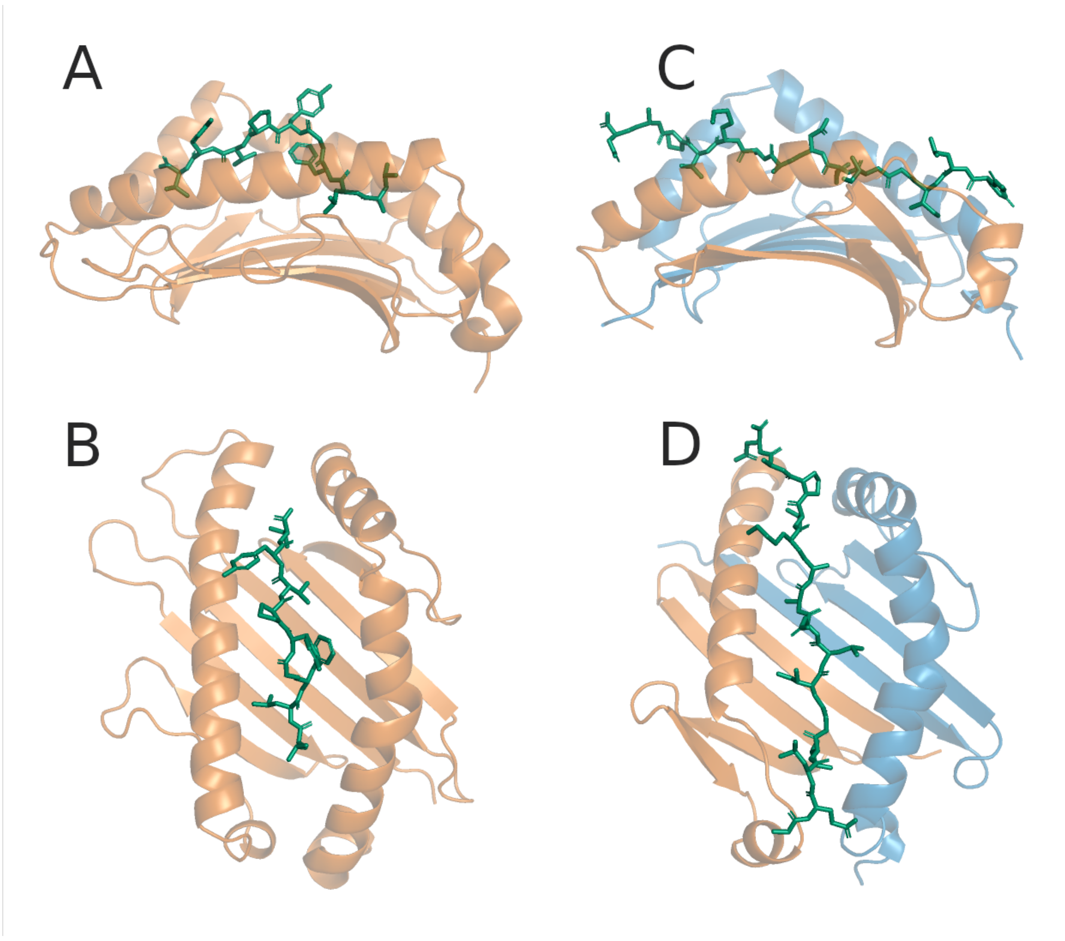


Figure 1.5: The tax nonamer peptide is shown (in green and in stick representation) presented in the MHC class I peptide groove (orange) from the side (A) and from above (B). The triosephosphate isomerase 15-mer peptide is shown (in green and in stick representation) presented in the MHC class II peptide groove (orange and blue) from the side (C) and from above (D). The two structures can be found in the PDB under codes 1DUZ and 1KLG, respectively.

from the groove and instead the residues show a pattern of alternating high and low exposure to TCRs (Singh *et al.*, 2020). Examples of peptides displayed in MHC class I and class II binding grooves are shown in Figure 1.5.

1.4.4 TCR-pMHC Complex Binding and Geometry

The majority of resolved structures of TCRs bound to pMHC show a relatively conserved binding mode, whereby the CDR3α and CDR3β loops are oriented over the center of the bound peptide, maximising interaction (Wucherpfennig *et al.*, 2009; Rudolph *et al.*, 2006). The CDR1 and CDR2 loops are generally responsible for binding the MHC α helices. Examples of bound TCR-pMHC complexes are shown

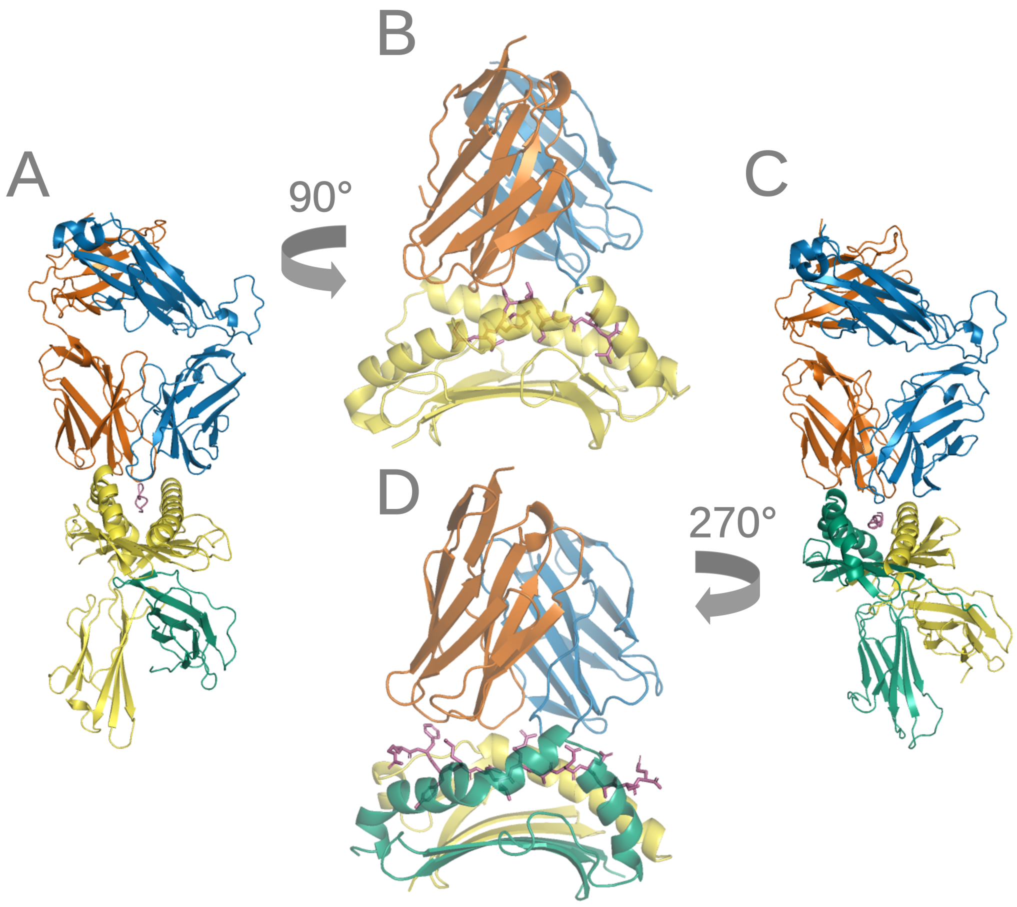


Figure 1.6: The crystallised bound complex of the 868 TCR and the HIV p17 Gag peptide presented by the HLA-A2 MHC class I molecule (found under PDB code 5NMF) is shown (A) with the TCR α chain highlighted in orange, β chain in blue, peptide in rose, HLA-A2 domain in yellow, and β_2M domain in green. The same complex is shown (B) rotated anticlockwise by 90° with only binding interface domains visible, the same highlighting, and the peptide shown in stick representation. The crystallised bound complex of the F24 TCR and the RQ13 peptide presented by the DR11 MHC class II molecule (found under PDB code 6CQL) is shown (C) with the TCR α chain highlighted in orange, β chain in blue, peptide in rose, DR11 α domain in yellow, and DR11 β domain in green. The same complex is shown (D) rotated clockwise by 270° with only binding interface domains visible, the same highlighting, and the peptide shown in stick representation.

for MHC class I and MHC class II in Figure 1.6. A TCR generally takes up a diagonal rotation above the pMHC such that the CDR1 and CDR2 loops may also make contact with the presented peptide and contribute to antigen specificity.

The diagonal binding mode of a specific TCR with respect to pMHC can be described by the crossing (or docking) angle and the incident angle. Traditionally, the optimal method for measuring the crossing angle has been to calculate the dot product of the vector that passes through the centroids of the conserved disulphide-forming sulfur atoms in the two TCR domains and the best-fit straight line that passes through the Cα atoms of the MHC helices (Rudolph *et al.*, 2006). This provides a measure of TCR twist relative to the pMHC. The incident angle is calculated as the angle between the axis of pseudo symmetry between the TCR variable domains and the vector normal to the MHC peptide groove plane. This provides a measure of TCR tilt relative to the pMHC (Pierce and Weng, 2013).

A large degree of variability in the crossing and (to a lesser extent) incident angles has been reported for resolved TCR-pMHC structures (Rudolph *et al.*, 2006; Pierce and Weng, 2013; Rossjohn *et al.*, 2015). TCR-pMHC complexes with unusual binding geometries have been implicated in limited or altered T cell responses (Adams *et al.*, 2011; Gras *et al.*, 2016) and autoimmune disease (Wucherpfennig *et al.*, 2009). Furthermore, visual inspection and centre of mass (CoM; defined in Appendix A) calculations have shown that these two measurements alone are incomplete descriptors of TCR binding modes (Blevins *et al.*, 2016; Singh *et al.*, 2020).

A new geometric description of TCR binding has recently been designed by Singh *et al.* (2020) that differentiates between TCR binding modes that were indistinguishable using crossing and incident angle calculations. In this novel coordinate system, the CoM of the MHC peptide binding domain is set as the origin. The y-axis is formed by the line of best fit through the Cα atoms of the two alpha helix walls of the binding groove; the x-axis is defined as normal to the y-axis, forming an xy plane; and the z-axis defined as normal to the xy plane. This system allows three new parameters to be defined: r, the distance between the MHC binding plane domain CoM and the TCR (V domain) CoM; ϕ, (tilt of TCR relative to MHC), the angle formed by the z-axis, the MHC CoM and the TCR (V domain) CoM; and θ, (rotation of TCR around MHC), the angle between the x-axis and the projection of the TCR (V domain) CoM onto the xy plane. These measurements are shown for

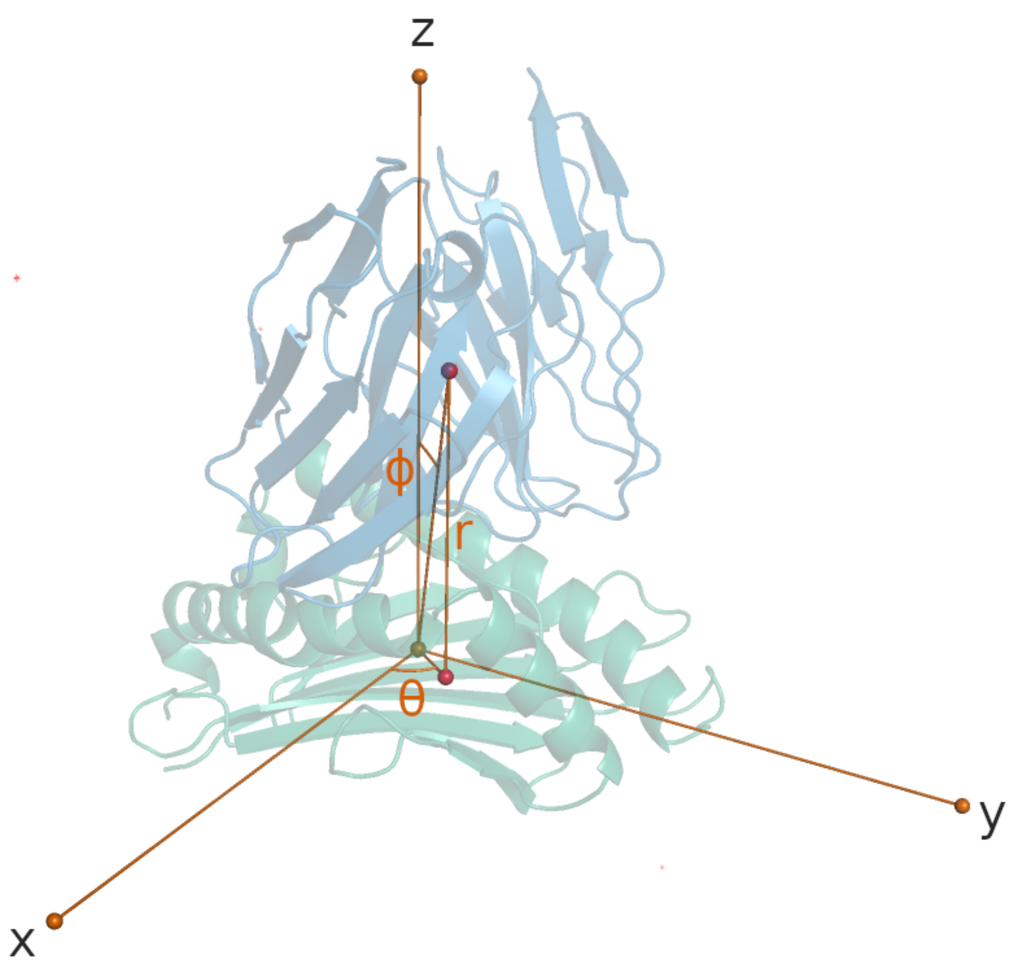


Figure 1.7: An influenza peptide presented by MHC Class I molecule HLA-A2 (green) bound by the JM22 TCR (blue) superimposed with the coordinate system devised by Singh *et al.* (2020). The geometry is described by three parameters: (1) r gives the distance between the TCR CoM and xy binding plane, (2) ϕ gives the angle which specifies the tilt of the TCR relative to the pMHC, and (3) θ gives the angle which specifies the rotation of the TCR around the pMHC. The values that correspond to the displayed structure are: $r = 29.10\text{Å}$, $\theta = 52.69°$ and $\phi = 12.64°$. These values were calculated using the Python script provided in Singh *et al.* (2020). The raw structure for this complex can be found under PDB code 1OGA.

an example TCR-pMHC structure in Figure 1.7.

Analysis using this coordinate system has revealed that class I and class II restricted TCRs orientate such that their CDR loops converge on peptide regions that are maximally solvent exposed, and bisect the high points of the MHC α helices (Singh *et al.*, 2020).

1.4.5 TCR Structural Databases

The growing number of available TCR and TCR-pMHC structures has encouraged efforts to curate TCR-specific structural databases that might permit the community to analyse TCR structures more easily and efficiently. The Structural TCR Database (STCRDab) automatically collects and annotates TCR and TCR-pMHC structural data from the Protein Data Bank (PDB) (Leem *et al.*, 2018). MHC type, $\alpha\beta$ and $\gamma\delta$ pairings, CDR loop canonical forms, and geometric measurements are all automatically determined. As of September 2021, STCRDab reports 518 PDB entries that contain at least one TCR structure. The majority of these contain $\alpha\beta$ TCRs and are the focus of most structural studies. Nevertheless, a small number of $\gamma\delta$ TCR structure have also been resolved.

The ATLAS (Altered TCR Ligand Affinities and Structures) database is a manually curated repository for TCR and TCR-pMHC structures which have associated binding affinity measurements recorded (Borrman *et al.*, 2017). ATLAS contains both structures collected from the PDB and homology modelled structures.

The TCR3d database is another repository that automatically identifies all TCR and TCR-pMHC structures in the PDB, and focuses on TCR targeting and antigen interactions (Gowthaman and Pierce, 2019). TCR3d reports MHC type, TCR docking angles, buried surface area and shape complementarity. Users can filter by virus or cancer targeting TCRs, and tools are provided for calculating the r, θ and ϕ parameters and the crossing and incident angles described in Section 1.4.4 for TCR-pMHC complexes. In addition, TCR3d provides a set of 30 TCR-pMHC complexes that have experimentally determined unbound TCR and pMHC structures. These structures are particularly useful for benchmarking TCR-pMHC structure prediction tools. Research conducted to expand this benchmark through the identification and analysis of other such structures is discussed in Chapter 3.

1.5 Computational Modelling of TCR and TCR-pMHC structures

1.5.1 Protein Structure Prediction

A complete understanding of the function of a protein is often impossible without knowledge of its three dimensional structure. Pioneering work in the 1950s produced the first experimentally resolved protein structures (Fersht, 2008). Over the subsequent decades, tremendous progress has been made in the determination of protein structures through innovations in x-ray crystallography, nuclear magnetic resonance (NMR) and, more recently, cryo-electron microscopy (cryo-EM) (Hameduh *et al.*, 2020). In response to the increasing quantity of crystallographic data and the advent of graphical hardware and software, the Protein Data Bank (PDB) was established in 1971 as a resource for the storage and search of protein structural data (Protein Data Bank, 1971), and remains the central repository for structural biologists today (wwPDB consortium, 2019). As of September 2021, the Protein Data Bank (Berman *et al.*, 2000) contains over 150,000 protein structures.

Despite these successes, breakthroughs in sequencing technology have created an ever-increasing gap between the enormous number of annotated protein sequences (over 100 million) and the number of resolved protein structures (Muhammed and Aki-Yalcin, 2019). Modern computing has allowed for cheaper, faster, and increasingly more accurate and automated approaches to the problem of structure prediction, and there is hope that these methods may reduce the divergence between available sequence and structural data (Muhammed and Aki-Yalcin, 2019; Hameduh *et al.*, 2020; Guo *et al.*, 2008).

Initial attempts to predict protein structure from sequence aimed to derive rules for protein folding using principles from thermodynamics (Levitt and Warshel, 1975). However, the enormous size and complexity of the search space that must be explored to locate the lowest free-energy structure into which an amino sequence should fold has proven beyond the reach of the scientific community (Guo *et al.*, 2008). Nevertheless, these early approaches have evolved into one of two branches of structural prediction methods. First, a branch that aims to address the protein folding problem directly with no prior structural information. These so-called "*ab initio*" methods include molecular dynamics (MD) simulations of protein folding, Monte Carlo optimisation routines for efficiently sampling conformational space, and contact prediction through evolutionary covariation (Bonneau and Baker, 2001). *Ab initio* methods tend to be computationally expensive.

The second branch of structural prediction methods overcomes the challenge of protein folding by ignoring it, and instead focuses on making use of existing protein structures as templates. Homology modelling is a "knowledge-based" technique that relies on the assumption that proteins with similar sequences will share structural features. The sequence of the target (the protein being modelled) is aligned against a set of template proteins with known sequence and resolved structure. Once templates have been identified, they are aligned with the target sequence and optimised. A backbone model is constructed from structural elements of the templates, and regions of the target with no template may be constructed through *de novo* loop modelling or by consulting loop databases. Side chain rotamers may be added, and a model is generally optimised through energy minimisation, before final validation (Hameduh *et al.*, 2020; Krieger *et al.*, 2003). Homology modelling has been a popular and high achieving strategy in the regularly held CASP (Critical Assessment of protein Structure Prediction) experiments over the last two decades (Moult *et al.*, 2018).

Protein threading (or fold recognition) is a modelling approach that has been used to predict structures for proteins that have the same fold as proteins with known structures, but for which no homologous structure is known. This method "threads" the target sequence, amino acid by amino acid, through that of a template protein. For a given local environment, the similarity between target and template is assigned a score. This process is repeated for a large number of templates, and the highest scoring template provides a prediction of the backbone for the target (Guo *et al.*, 2008). Threading relies on the assumption that the total number of protein folds in nature is small and that the majority have already been discovered. This technique has proved a useful tool, particularly when the sequence identity between the target and known structures is low.

Over the last few years, machine learning (ML) techniques have been incorporated into all aspects of protein structure prediction (AlQuraishi, 2021). The ability to lever enormous quantities of input data coupled with extensive deep learning architectures, and facilitated by considerable computational resources, has seen a dramatic improvement in protein structure prediction (Jumper *et al.*, 2021). The unrivaled success of Deepmind's AlphaFold2 software at the latest round of the CASP experiments has been a milestone for the field, and it has been argued that the folding problem has now been essentially solved for single domain proteins (AlQuraishi, 2021).

Preliminary studies that have extended this approach to predict protein-protein complexes have shown a number of successes, though it has been noted that attempts at antibody-antigen structural prediction remain largely unsuccessful (Evans *et al.*, 2021; Yin *et al.*, 2021). Nevertheless, improvements in antibody-specific epitope prediction have been reported when combining AlphaFold with rigid-docking approaches (Xu *et al.*, 2022). Recently, a study that compared AlphaFold and MODELLER for TCR-pMHC complex modelling resulted in divergent sets of structural models, however benchmarking against known TCR-pMHC structures was not undertaken (Rollins *et al.*, 2022).

The following sections describe homology modelling approaches that have been designed specifically for modelling TCR and TCR-pMHC complex structures.

1.5.2 Homology Modelling of TCRs

Provided that adequate templates can be identified, homology modelling approaches are not limited to only single domain proteins. Due to their central role in the adaptive immune response and their potential for cell-based and soluble therapeutics, a large number of TCR structures have been resolved, providing a broad set of templates for TCR structural modelling. While numerous homology modelling software platforms have been designed for general purpose modelling, over the last few years a small number of platforms have been published for TCR-specific modelling.

The LYRA (LYmphocyte Receptor Automated modeling) web server provides automatic B cell receptor (BCR) and TCR modelling (Klausen *et al.*, 2015). LYRA selects framework templates based on BLOSUM score, and CDR loops based on canonical structures. Users have the additional options of manually selecting templates or excluding certain templates from selection. The interface residues of the selected chains are subjected to packing (if the chains originate from different templates and exhibit atomic clashes), loops are grafted onto the framework regions, side chains are repacked, and the entire model subject to energy minimisation using the ENCAD program (Levitt *et al.*, 1995). The LYRA web server was used for modelling successive generations of TCR structures from sequence for research described in Chapter 5.

The TCRModel web server automatically models TCRs using the Rosetta modelling framework (Leaver-Fay *et al.*, 2011), and has been shown to compare favourably to LYRA (Gowthaman and Pierce, 2018). TCRModel also selects independent templates for framework and CDR regions, assembles models by grafting the loops onto the framework regions, and then refines the backbone and side chains using Rosetta energy minimisation protocols. Users can choose to further refine the CDR3 loop of their models through an implemented kinematic closure loop modelling algorithm (although this feature significantly increases the length of the task). Like LYRA, TCRModel also allows the exclusion of undesirable templates.

The Repertoire Builder web server uses multiple sequence alignment (MSA) for bulk generation of BCR or TCR structures (Schritt *et al.*, 2019). Separate MSAs are constructed for CDR and framework templates and then extended to include the target sequence. Each alignment between target and template is scored per residue to select an appropriate template. The backbone structure is assembled using conserved anchor residues between the CDR, framework and orientation templates, and side chains are re-modelled using SCWRL4 (Krivov *et al.*, 2009) where appropriate. Repertoire Builder has been reported to compare favourably with LYRA and TCRModel, and is capable of generating 10,000 structures in approximately 30 minutes. The web server is backed by considerable computational resources, and the designers have emphasised that other existing TCR modelling platforms would be capable of similar throughput if distributed over a large number of processors.

The TCRBuilder web server uses structural templates to generate an ensemble of TCR models predicted to capture the dynamic nature of the TCR binding region (Wong *et al.*, 2020). Framework and orientation templates are selected by sequence identity, CDR templates are predicted using FREAD (Choi and Deane, 2010) or modelled using Sphinx (Marks *et al.*, 2017), and side chains are modelled by PEARS (Leem *et al.*, 2018). TCRBuilder has shown comparable performance to LYRA, TCRModel and Repertoire Builder.

1.5.3 Homology Modelling of TCR-pMHC Complex Structures

Traditional methods for modelling protein-protein complexes reside in the realm of computational docking. Where either the two (or more) independent protein structures are already known, or can be modelled accurately, protein-protein docking approaches attempt to predict the relative position and orientation of the components accurately through efficient sampling and effective scoring to build a complete model of the complex. A more detailed overview of the docking field is provided in Section 1.5.4. However, there is no reason that homology modelling should be only limited to unbound proteins, provided adequate templates of complexes exist.

Two recently published software platforms have demonstrated that full TCR-pMHC complex models can be constructed using the growing number of experimentally determined complexes as templates. The TCRpMHCModels platform (Jensen *et al.*, 2019) uses LYRA to model the TCR component and the homology modelling tool MODELLER (Šali and Blundell, 1993) to model the pMHC component from sequence independently. MODELLER is then used to construct the TCR-pMHC complex using the modelled TCR and pMHC components and one or more TCR-pMHC templates. The platform has been built to model only complexes containing MHC class I, is supported by a free-to-use web server, and has been shown to outperform the bespoke TCRFlexDock docking platform for TCRs for a set of test cases. The TCRpMHCModels web server was used for modelling successive generations of TCR-pMHC structures from sequence for research described in Chapter 5.

The ImmuneScape (Li *et al.*, 2019) platform builds TCR-pMHC complex models through similar approaches to TCRpMHCModels. Models for the TCR and pMHC are constructed independently and then oriented using TCR-pMHC templates to build the complex. Unlike TCRpMHCModels, ImmuneScape models the CDR regions after the TCR and pMHC templates have been combined (Teraguchi *et al.*, 2020). Furthermore, it permits modelling of complexes containing both MHC class I and MHC class II molecules. At present there is little in the way of benchmarking reported for ImmuneScape, and it is hard to judge how the two platforms compare.

Advantages of homology modelling methods include the rapid speed at which models can be produced and the accuracy of predictions if high sequence identity can be found with template structures. Where adequate templates cannot be found, or a more rules-based approach is desired, computational protein-protein docking is a well-established field.

1.5.4 Computational Docking

Previous sections in this chapter have discussed methods for modelling (most frequently) individual proteins from their amino acid sequence using computational methods. In general, resolving the structure of a protein-protein complex experimentally is more challenging than for an individual protein (Vakser, 2014). While comparative techniques have been explored for modelling proteins that have a large number of templates available, such as BCRs and TCRs, computational docking offers an alternate approach for protein-protein structural modelling, and has long been a focal point of structural biology.

The scope of computational docking is very broad, encompassing methods for the structural prediction of protein-protein complexes (Porter *et al.*, 2019), protein-ligand (small molecule) complexes (Pagadala *et al.*, 2017), protein-nucleic acid complexes (Tessaro and Scapozza, 2020), protein-peptide complexes (Agrawal *et al.*, 2019), and protein-carbohydrate complexes (Pérez and Tvaroška, 2014). Molecular docking approaches are regularly used throughout the virtual screening and lead optimisation processes for drug discovery (Meng *et al.*, 2011). Macromolecular (protein-protein) docking approaches provide insight into protein interactions at the binding interface and provide a basis for rational protein design (Vakser, 2014). The CAPRI (Critical Assessment of PRediction of Interactions) community-wide experiment has served as a means for researchers to regularly test and discuss their algorithms with others, and has seen the field flourish over the last two decades (Janin *et al.*, 2003).

In general, the computational docking process involves efficiently sampling thousands of possible positions and orientations of the binding partners relative to one another, with the aim of producing a model that captures the true structure (native solution) of the complex. Each pose is ranked according to a scoring function that is designed to predict how similar a generated model is to the native solution.

The size of the search space presents a challenge for sampling procedures. Many algorithms (Chen and Weng, 2003; Tovchigrechko and Vakser, 2006; Cheng *et al.*, 2007; Macindoe *et al.*, 2010) are based on the rigid-body fast Fourier Transform (FFT) shape complementarity approach (Katchalski-Katzir *et al.*, 1992) developed in the 1990s for rapid, yet exhaustive, searching. Others have made use of geometric hashing techniques (Fischer *et al.*, 1992; Schneidman-Duhovny *et al.*, 2005).

A wide variety of techniques have been explored since in attempts to improve upon the speed and accuracy of docking. Geometric search has been extended to include electrostatic (Gabb *et al.*, 1997) and desolvation contributions (Chen and Weng, 2003). Speed improvements have been gained through implementing the algorithm in spherical polar (rather than Cartesian) coordinates (Ritchie and Kemp, 2000), and by using graphics processing units (GPUs) (Ritchie and Venkatraman, 2010). External scoring functions that include knowledge-based terms that take advantage of information from existing structures as well as physics-based terms have proven successful. Consensus-based scoring and scoring functions that make use of existing evolutionary information have also performed well (Oliva *et al.*, 2013).

An alternate approach to FFT docking has been to make use of Monte Carlo algorithms. Binding partners arranged in some initial position and orientation are translated and rotated over a large number of successive steps. At each step, the pose is scored, and then either accepted or rejected based on the score, with the aim of converging upon the native solution. This is the basis for the popular and successful RosettaDock protocol (Chaudhury *et al.*, 2011). Other algorithms have attempted similar convergence on solutions using optimisation routines, such as swarming (Moal and Bates, 2010; Jiménez-García *et al.*, 2018), simulated annealing (Dominguez *et al.*, 2003) and genetic algorithms (Gardiner *et al.*, 2001). Clustering of docked models has also proved a powerful tool for improving the accuracy of results (Comeau *et al.*, 2004; Mashiach *et al.*, 2010; Dominguez *et al.*, 2003).

While the occurrence of conformational change upon protein-protein binding is well established, accounting for the additional degrees of freedom introduced by flexibility proves very difficult for traditional approaches such as FFT docking. Modelling conformational change is often performed as an additional routine after an initial FFT search, such as by short MD simulation, normal mode analysis, or rigidity and hinge detection algorithms (Andrusier *et al.*, 2008). Alternatively, algorithms that reduce the search space through the use of optimisation routines are more readily capable of exploring flexibility during the initial docking process (Moal and Bates, 2010; Jiménez-García *et al.*, 2018; May and Zacharias, 2008; Zacharias, 2003).

A number of docking platforms have introduced features to allow users to specify additional information alongside the protein binding partners to guide the docking. These "data-driven", "information-driven" or "integrative modelling", approaches have been highly successful in recent years (Rodrigues and Bonvin, 2014). Permitting the specification of binding residues or regions has been popularly implemented into algorithms to bias the scoring step (Kozakov *et al.*, 2017; Tovchigrechko and Vakser, 2006; Schneidman-Duhovny *et al.*, 2005; Pierce *et al.*, 2014b; Jiménez-García *et al.*, 2018), the sampling step (Dominguez *et al.*, 2003; Moal and Bates, 2010; Macindoe *et al.*, 2010; Chaudhury *et al.*, 2011), or a post-modelling filtering step (Cheng *et al.*, 2007). Docking platforms have also made use of information from small angle x-ray scattering (SAXS) (Karaca and Bonvin, 2013; Pons *et al.*, 2010) and Cryo-EM density maps (de Vries and Zacharias, 2012) to improve results. The HADDOCK platform, which is very much designed with integrative modelling principles in mind, accepts mutagenesis, mass spectrometry and NMR analysis data as ambiguous restraints to drive the docking (Dominguez *et al.*, 2003).

The construction of TCR-pMHC model structures has revealed biological insight in a range of contexts. The ClusPro rigid-body platform has been used in conjunction with MD refinement in the search for cross-reactive epitopes for vaccines in response to the 2009-H1N1 influenza pandemic ("swine flu"), revealing effects of TCR residue mutations on binding affinity (Su *et al.*, 2013). ClusPro has also been used to study promiscuous T cell epitope prediction for vaccine design against Streptococcus pyogenes (Ebrahimi *et al.*, 2019), in the generation of cytotoxic T cells with enhanced anti-tumor activity (Ouchi *et al.*, 2018), and for the investigation of possible superantigenic fragments in the spike protein of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) (Cheng *et al.*, 2020). The RosettaDock platform has been used to help reveal the influence of CDR loops on the V_α/V_β interdomain angle and the impact this can have on antigen binding (McBeth *et al.*, 2008). Although unfortunately no longer available, FTDOCK (Gabb *et al.*, 1997) has been used to dock the V_β domain of TCR to pMHC to provide insight into the mechanisms of self reactivity in the context of rheumatoid arthritis (Rosa *et al.*, 2010), and AutoDock has been used to build TCR-pMHC models to reveal details of the TCR binding mode in the context of multiple sclerosis (Kato *et al.*, 2010).

Despite the broad interest in constructing TCR-pMHC complexes using computational docking, there has been little research conducted into the accuracy and reliability of these algorithms for the purposes of TCR modelling. Most new docking algorithms are tested using the protein-protein docking benchmark (Vreven *et al.*, 2015), which contains a diverse set of bound and unbound proteins of varying degrees of modelling difficulty. However, no TCR-pMHC structures feature in this benchmark, and consequently few platforms are challenged in this context. The TCRFlexDock platform (Pierce and Weng, 2013), an extension of RosettaDock, is the only bespoke platform for TCR-pMHC modelling. To address the specific challenges in TCR-pMHC complex docking, a TCR-pMHC specific benchmark of 20 cases was assembled to test the TCRFlexDock software. This has been expanded to 30 cases in the TCR3d database. Efforts to identify additional benchmark cases comprise a portion of the work presented in this thesis, and are reported in Chapter 3.

To provide the community with guidance on the accuracy of general-purpose docking platforms for TCR-pMHC modelling, Chapter 4 of this thesis reports the performance of four popular docking algorithms — ClusPro, HADDOCK, LightDock and ZDOCK — against the expanded TCR benchmark. Each platform was chosen for its ability to incorporate additional information into the docking process. Varying levels of detail about the TCR-pMHC binding interface were provided to each platform to assess how final models might be improved given *a priori* information. Conformational changes that take place upon binding have been reported as being higher on average for TCR-pMHC structures than for unbound antibody-antigen structures (Pierce and Weng, 2013). Chapter 4 also explores the ability of the two flexible docking platforms — HADDOCK and LightDock — to accurately model conformational change in the TCR CDR loops upon binding.

1.5.5 Alternative Methods for TCR-pMHC Modelling

Although this thesis is largely concerned with the use of computational docking to model TCR-pMHC complexes, the problem has been approached using alternative methods. The bespoke DynaDom method (Hoffmann *et al.*, 2017) for TCR-pMHC assembly, implemented within the DynaCell molecular modelling suite (Antes, 2010), is something of a hybrid approach of template-based, docking and molecular dynamics techniques. It places an emphasis on the prediction of TCR V_α/V_β inter-domain and TCR-pMHC orientations, performing particularly well in the former case. It is unclear how long each modelling task takes, and the software does not appear to be currently available to the public.

Molecular dynamics (MD) simulations have been briefly discussed in the context of performing short energy minimisation routines on modelled protein or protein complex structures. An MD protocol numerically solves Newton's equations of motion for a system of particles over time and under certain thermodynamic constraints, and this field is another well-established branch of molecular modelling. A major advantage of MD simulations is the insight that can be obtained by observing the movement of particles over time, rather than the static snapshots of crystallised proteins. However, these simulations are computationally expensive, with run times often extending into several weeks for the analysis of a single complex (Knapp *et al.*, 2015).

Generally, MD simulations of TCR-pMHC complexes are more concerned with investigating structural mechanisms for TCR cross-reactivity and binding than TCR-pMHC complex assembly. Furthermore, computationally modelled structures are often required prior to a simulation study, and therefore constructed using homology approaches (Leimgruber *et al.*, 2011) or side-chain substitution tools (Knapp *et al.*, 2008, 2011). MD simulation for TCR-pMHC structural analysis has been reviewed by Flower *et al.* (2010) and, more recently, by Knapp *et al.* (2015). Application of MD simulations to the design of novel TCRs is further discussed in Section 1.6.

1.6 Rational Design and TCR Engineering

The breadth of TCR diversity and their critical role in fighting disease has made them attractive targets for protein engineering projects. Introducing enhanced binding properties through mutation has been explored for vaccine design (Chen *et al.*, 2005; Koup and Douek, 2011), and adoptive cell transfer therapy (Rosenberg *et al.*, 2008). A number of studies have successfully engineered novel high-affinity TCRs through directed evolution (Holler *et al.*, 2000; Li *et al.*, 2005; Dunn *et al.*, 2006), whereby protein function is modified through successive rounds of random mutation and artificial selection (Romero and Arnold, 2009).

In addition to these wet lab approaches, a number of rational design studies have engineered high-affinity TCRs using computational analysis (Haidar *et al.*, 2009; Alli *et al.*, 2011; Zoete *et al.*, 2013; Pierce *et al.*, 2014a). In contrast to the random mutation of directed evolution, rational design studies focus on identifying specific protein sites that are predicted to enhance function upon mutation. The advantages and disadvantages of directed evolution and rational design have been reviewed by Chica *et al.* (2005), in the context of enzymes, and a synthesis of the two approaches has been predicted to pave the way for new areas of protein design.

Lead optimisation techniques are common place in the realm of virtual drug discovery (Hughes *et al.*, 2011), whereby the various pharmaceutical properties of a compound identified as a leading drug candidate are enhanced through successive rounds of modification and validation. A number of automatic approaches have been developed for lead optimisation using fragment growing and replacement techniques (Lamoree and Hubbard, 2017; Li *et al.*, 2015). In attempts to automate replacement and iteration *in silico*, optimisation routines such as genetic algorithms have been implemented to converge on high quality solutions (Singh *et al.*, 1996; Spiegel and Durrant, 2020). Genetic algorithms have been similarly applied to the optimisation of peptides for validation *in vitro*, applying mutation and crossover to amino acid residues, rather than compound fragments (Fjell *et al.*, 2011; Röckendorf *et al.*, 2012; Burnside *et al.*, 2019). Additionally, genetic algorithms have been explored in combination with structural modelling methods to optimise peptides for MHC binding (Knapp *et al.*, 2011). Chapter 5 of this thesis describes a prototype platform for the design of TCRs with novel properties using a genetic algorithm and structural modelling methods.

1.7 Scope of Thesis

This thesis explores computational methods and tools for the analysis of TCR sequence and structure. A short summary of the work presented in the following chapters is provided below.

- Chapter 2 provides a detailed overview of the entire Decombinator pipeline and the new features that have been introduced for the latest official release, Decombinator V4. Additional tools that have been developed to aid in repertoire analysis or pipeline testing are also described. An adaption of Decombinator for the analysis of short-read single cell data is also described, and compared to two alternative platforms using independent datasets.

- Chapter 3 expands the existing benchmark of TCR-pMHC bound and unbound structures to aid the development and testing of tools for TCR-pMHC complex structural modelling. Methods are described for the cleaning, relabelling and repairing of the publicly available data.

- Chapter 4 compares and contrasts four general-purpose computational docking platforms in the context of TCR-pMHC modelling. A background is provided for each platform along with useability notes for future researchers. The success rate of each platform is evaluated against the expanded TCR benchmark and with varying degrees of additional information about the TCR-pMHC binding interface. The platforms are compared to the bespoke TCRFlexDock modelling platform, and CDR loop modelling is assessed for the flexible docking approaches. It is hoped that this study will provide researchers with guidance and rationale when choosing appropriate software for TCR-pMHC complex modelling in future projects.

- Chapter 5 describes a novel platform for the computational engineering of TCRs. The software centres around a genetic algorithm that evolves the amino acid sequence of a population of TCRs over successive generations according to a customisable fitness function. For every iteration, a structural model of each TCR (or TCR-pMHC complex) is built and evaluated according to the fitness function, such that desired features are engineered into the models. It is hoped that this platform demonstrates a proof of concept of the scope of automated approaches for biological design that will only improve as structural modelling methods become more accurate.

Chapter 2

The Decombinator package for TCR repertoire analysis

2.1 Introduction to the Decombinator Package

Decombinator was one of the earliest software packages designed to detect TCR signatures in the expanding realm of bulk immunological sequence data (Thomas *et al.*, 2013). While this computational pipeline operates as a stand-alone tool for sequence analysis, it is ideally complemented by an optimised experimental protocol designed by the Chain lab at University College London (UCL) (Uddin *et al.*, 2019).

The Decombinator platform has been used to annotate TCR repertoire data in a range of biological settings. Changes in the TCR repertoire have been analysed in the context of immunization against Mycobacterium tuberculosis (Thomas *et al.*, 2014), in chronic HIV infection following antiretroviral therapy (Heather *et al.*, 2016), and in response to cord blood transplantation (Gkazi *et al.*, 2018). Decombinator has also been used to explore the immune response to neuroblastoma (Fisher *et al.*, 2014) and to clear cell renal cell carcinoma (Au *et al.*, 2021) (preprint — manuscript accepted). Furthermore, the pipeline has been used to investigate TCR repertoire diversity in naïve and memory subsets from healthy donors (Oakes *et al.*, 2017), and in the context of non-small-cell lung cancer (NSCLC) (Joshi *et al.*, 2019).

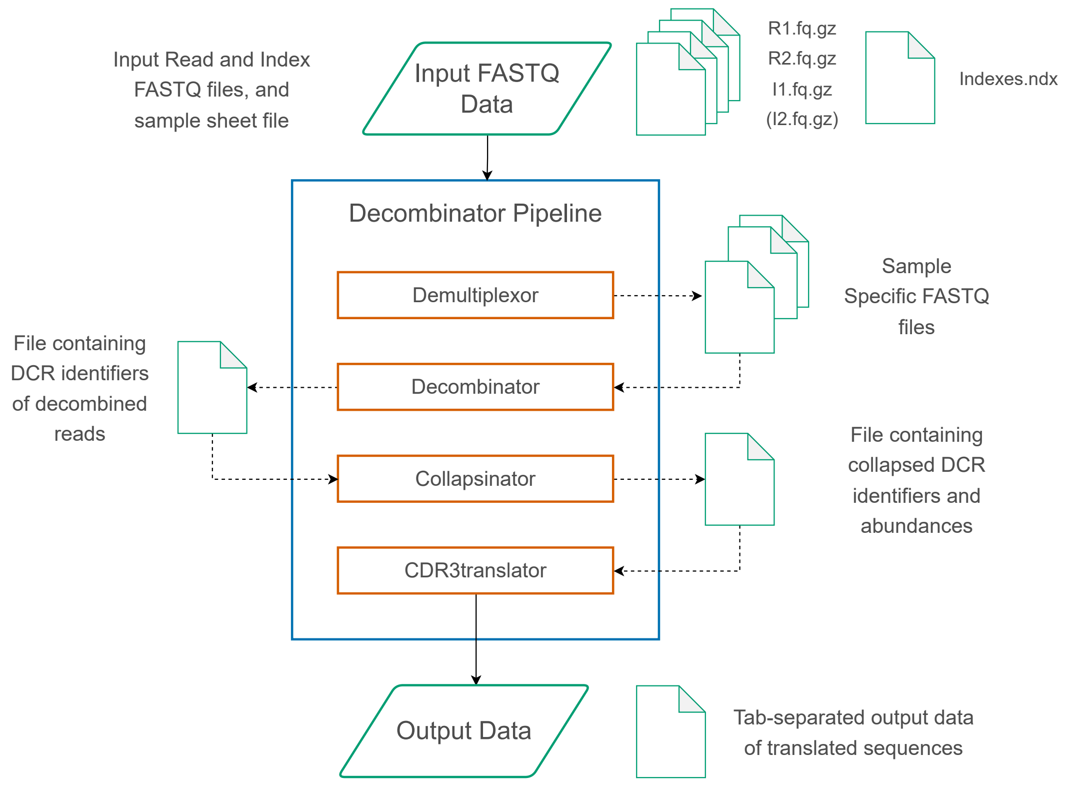

Input Read and Index FASTQ files, and sample sheet file
Input FASTQ Data
R1.fq.gz
R2.fq.gz
I1.fq.gz
(I2.fq.gz)
Indexes.ndx
Decombinator Pipeline
Demultiplexor
Decombinator
Collapsinator
CDR3translator
Sample Specific FASTQ files
File containing DCR identifiers of decombined reads
File containing collapsed DCR identifiers and abundances
Output Data
Tab-separated output data of translated sequences

Figure 2.1: An overview of the main components and the flow of data through the Decombinator pipeline.A DCR identifier is a shorthand annotation that unambiguously describes a TCR sequence, and is described further in Section 2.1.4.

The name "Decombinator" originally referred to a single software module (Thomas *et al.*, 2013) that was designed to search for TCR signatures by inferring the recombination events that produced a given read (Peacock *et al.*, 2021). Over subsequent years, the name has been extended to refer to a suite of complementary tools that extend beyond the original module. In general, Decombinator offers tools for: sorting data produced by massively parallel sequencing runs into separate sample files; V and J gene read annotation; sequence error correction; full DNA and protein translations; and CDR3 extraction. This functionality is divided among four primary modules: (1) Demultiplexor, (2) Decombinator, (3) Collapsinator, and (4) CDR3translator.

An overview of the computational pipeline is provided in Figure 2.1. Section 2.1 of this chapter provides an introduction to the main components of the original Decombinator platform prior to work undertaken over the course of this PhD project (except where explicitly referred to as Decombinator V4). The original pipeline was written collaboratively by a number of researchers, and some of the extensions that have been made to the pipeline have been formally published over the years (Thomas *et al.*, 2013; Best *et al.*, 2015; Oakes *et al.*, 2017).

Section 2.2 describes the latest version of the pipeline, Decombinator V4 (Peacock *et al.*, 2021), which was developed over the course of this PhD project. The extensions presented in this section can be assumed to have been made by the author of this thesis, except where collaborators are explicitly acknowledged.

Section 2.3 describes a suite of additional tools that may be used in conjunction with Decombinator to aid in TCR repertoire analysis. These scripts were written over several years by multiple collaborators. Where an existing tool has been extended for compliance with Decombinator V4 (such as being upgraded from Python 2 to Python 3), the author of this thesis can be assumed to have been the sole contributor. Where new tools have been written by others or in collaboration with the author of this thesis, these collaborators are explicitly acknowledged.

Section 2.4 outlines an adaption of Decombinator for use with short-read fragmented data. This work was undertaken solely by the author of this thesis. The performance of this adaption is compared to two alternative TCR repertoire analysis packages for two single cell data sets. Except where explicitly stated otherwise, this comparative study was undertaken by the author of this thesis.

Finally, Section 2.5 offers some perspective on what the future might hold for the Decombinator pipeline. Decombinator is open source and made freely available online at `https://github.com/innate2adaptive/Decombinator`.

2.1.1 Experimental Protocol

It is recommended that the Decombinator computational pipeline is used in conjunction with an experimental protocol designed by the Chain lab at University College London. The technicalities of the experimental design are not detailed here, but the interested reader is directed to a recent publication by Uddin *et al.* (2019), where each step of the protocol is described in full. However, it is worth drawing attention to three key features that make this pipeline valuable to the immunological community.

Firstly, unlike many commercial TCR sequencing services, this experimental pipeline is open-source — full details of the protocol are freely available, and required primers, enzymes and materials are easy to source (Uddin *et al.*, 2019).

Secondly, the pipeline is cheap to run. The total cost of a typical reaction is less than $50. Sequencing samples in parallel as part of the same run also reduces costs. The Chain lab typically analyses 80 samples per run using an Illumina NextSeq machine, without significant loss in read depth or repertoire coverage (Uddin *et al.*, 2019).

Finally, the experimental protocol is designed to incorporate unique molecular identifiers (UMIs) (Weinstein *et al.*, 2009; Mamedov *et al.*, 2013; Shugay *et al.*, 2014) with high efficiency. UMIs are used in the Decombinator pipeline to correct for sequencing error introduced by the incorrect assignment of base pairs by the sequencing machine, as well as error introduced through several rounds of polymerase chain reaction (PCR) amplification of the recombined TCR α and β chains (Oakes *et al.*, 2017). Crucially, these error-correction procedures allow for quantitative estimates of TCR gene abundance (Best *et al.*, 2015).

The construct produced by the original experimental protocol is shown in Figure 2.2. The UMI is composed of two random hexamers separated and bordered by two known spacer sequences (identical 8 base pair I8 oligonucleotides). This barcoding scheme is typically referred to in shorthand as the “I8 oligo” in Decombinator documentation. More recently, a new barcoding schema has been developed and incorporated into the pipeline, and is discussed further in Section 2.2.2.

When sequenced, the raw data generated by the machine must typically be converted to FASTQ format to produce the output reverse, forward and index files. The structure of the reads in these files is additionally illustrated in Figure 2.2, and the file conversion process is described in the following section. The original experimental protocol makes use of two index sequences, shown here as `x1` and `x2`, for demultiplexing purposes. This protocol is referred to throughout this chapter as the internal dual index (IDI) protocol, named for the `x1` index contained within the R1 sequenced read. The latest version of Decombinator is also compatible with a

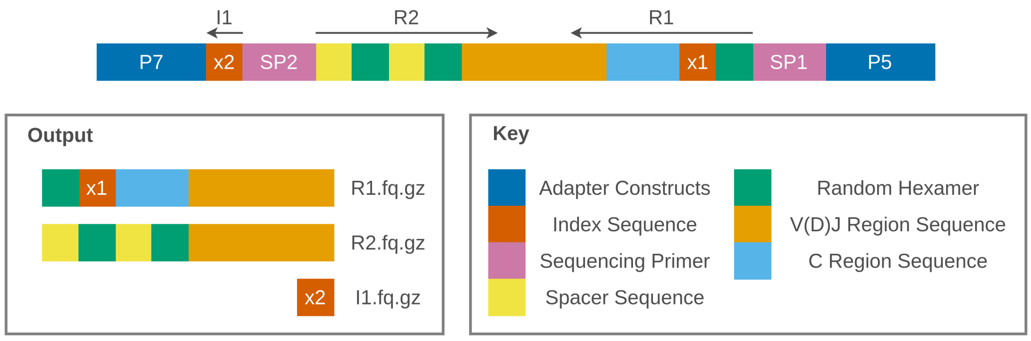


Figure 2.2: Schematic for the (internal dual index) sequencing of the TCR construct assembled through the recommended experimental protocol for use with the Decombinator software package. The `x1` and `x2` indexes are used by Demultiplexor to separate reads into sample specific files, while the random hexamers make up the molecular barcode.

new unique dual index (UDI) protocol, which is outlined in Section 2.2.1.

2.1.2 Preprocessing of Raw Sequence Data

Sequencing performed by the Chain lab is typically accomplished using Illumina MiSeq or NextSeq HTS machines. The raw data produced by these machines are stored in binary base call (`.bcl`) format. On some machines, such as the Illumina MiSeq, these files are automatically demultiplexed by default to the commonly used FASTQ format using the Illumina SP2 index (index `x2` in Figure 2.2).

FASTQ input for the Decombinator pipeline, however, must be demultiplexed using both the SP2 index and an additional SP1 index introduced during the experimental protocol (`x2` and `x1` in Figure 2.2, respectively). Therefore, the raw data should be converted to FASTQ format without automatically demultiplexing, and instead processed using the custom Demultiplexor script of the Decombinator platform.

File conversion is typically performed using the Illumina `bcl2fastq` conversion software (`http://emea.support.illumina.com/downloads/bcl2fastq-conversion-software-v2-20.html`). It should be noted that separate index FASTQ files are not always generated as default by the sequencing machine. Under these circumstances, it is recommended that an experienced Illumina user should be consulted to alter the configuration files of the machine appropriately.

If using an IDI protocol, as illustrated in Figure 2.2, the `bcl2fastq` conversion should be performed as:

```
bcl2fastq --runfolder-dir $RUN_DIR -o $OUT_DIR --use-bases-mask
   y*,y*,y* --minimum-trimmed-read-length 6
   --mask-short-adapter-reads 0 --no-lane-splitting
```

If using the Illumina MiSeq machine, `R1.fastq.gz`, `R2.fastq.gz` and `I1.fastq.gz` files are generated, which can be used by the Demultiplexor script. If using the Illumina NextSeq machine, the output files have different default labelling, and `R1.fastq.gz`, `R2.fastq.gz` and `R3.fastq.gz` files (R1, I1, and R2 in Figure 2.2, respectively) are generated.

If using the Decombinator V4 UDI protocol, described in Section 2.2.1 and illustrated in Figure 2.9, the conversion should be performed as:

```
bcl2fastq --runfolder-dir $RUN_DIR -o $OUT_DIR --use-bases-mask
   y*,y*,y*,y* --minimum-trimmed-read-length 6
   --mask-short-adapter-reads 0 --no-lane-splitting
```

When using the Illumina NextSeq and Novoseq machines, the `bcl2fastq` output files are generated as: `R1.fastq.gz`, `R2.fastq.gz`, `R3.fastq.gz` and `R4.fastq.gz` (R1, I1, I2, and R2 in Figure 2.9, respectively).

2.1.3 Demultiplexor Module

For the sake of efficiency, a typical TCR sequencing experiment involves the sequencing of many biological samples pooled together (multiplexed) as part of the same run on a sequencing machine. The Demultiplexor script uses index sequences to identify the specific sample to which a given sequence belongs, and to then save that sequence into a data file specific to that sample. Output sample files are written in FASTQ format and compressed by default. Demultiplexor has been written specifically for the TCR library preparation wet lab protocol used by the Chain lab (Uddin *et al.*, 2019), and some modification may be required if using different experimental setups.

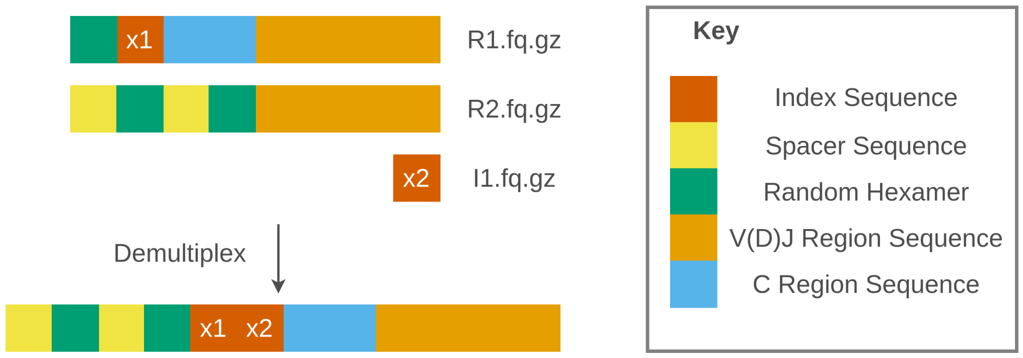


Figure 2.3: Sequences in the R1, R2 and I1 files pertaining to a single read are rearranged by the Demultiplexor script and saved to a sample-specific file defined by the `x1` and `x2` indices. This schematic describes the demultiplexing process for IDI experimental protocols.

As described in Section 2.1.2, Illumina sequencing machines produce three or four output data files. In the context of the Chain lab IDI protocol, these files contain: (1) the first sequencing read (R1) containing the V(D)J sequence and the SP1 demultiplexing index (`x1` in Figure 2.2 and Figure 2.3); (2) the second sequencing read (R2) which contains the UMI barcode sequences, and reads into the start of the 5′ untranslated region (UTR) of the TCR; and (3) the 8 base pair index read (I1) containing the second demultiplexing SP2 index (`x2` in Figure 2.2 and Figure 2.3). In the more recent UDI protocol, the SP1 index (`x1`) is removed from its position in R1 to a position between the SP1 sequencing primer and P5, and the read (I2) is written to a fourth file.

Iterating through the three files, Demultiplexor extracts relevant subsequences from each read and combines them into a new single sequence. This output sequence is composed of: the first 45 base pairs of R2 (which covers the I8 oligo barcoding schema, or the longer M13 oligo barcoding schema described in Section 2.2.2, along with a few additional buffer base pairs); the 6 base pair SP2 index from I1; the 6 base pair SP1 index from R1; and the remainder of the R1 read, which includes the end of the constant region and the majority of the variable region of the TCR sequence. These reconstructed sequences are saved to individual FASTQ files specific to the SP1 and SP2 indices.

User-friendly file naming can be achieved by supplying an additional file con-

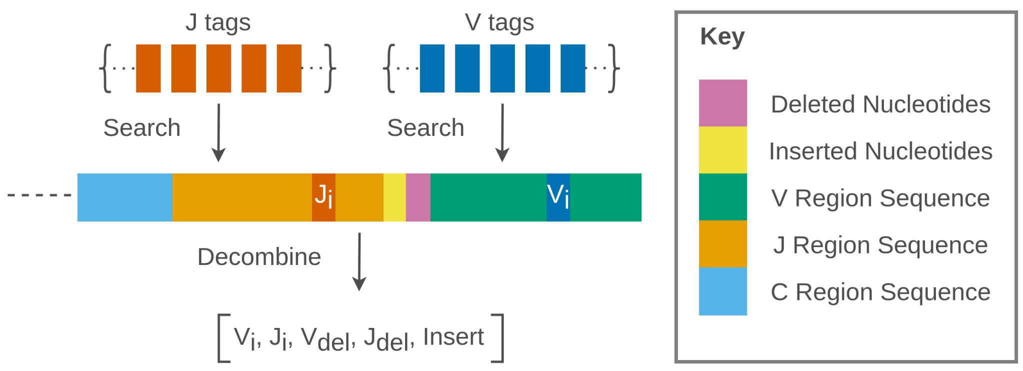


Figure 2.4: Each sequence provided to Decombinator is searched for short tag sequences that are each unique to a given V or J gene. If both a V and J tag are identified in the sequence, a 5-part identifier (DCR) is assembled by Decombinator as a descriptor of the TCR sequence.

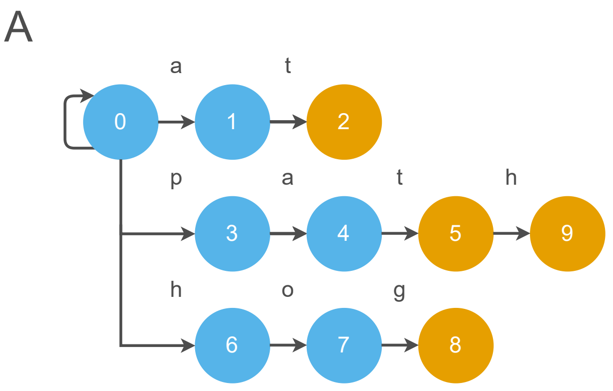
taining SP1 and SP2 index combinations alongside desired sample names. Demultiplexor also writes the accompanying FASTQ identifier and sequence quality scores to the output files. The demultiplexing procedure is illustrated in Figure 2.3.

The latest version of Demultiplexor has been upgraded to support UDI protocols, and is discussed further in Section 2.2.1.

2.1.4 Decombinator Module

The Decombinator script operates as the core of the whole pipeline, efficiently identifying TCR sequences in FASTQ reads. This is achieved through the novel application of the finite-state pattern-matching Aho-Corasick algorithm (Aho and Corasick, 1975), which searches reads for V and J genes. This algorithm is outlined in the following section. Following V and J gene assignment, junctional insertions and deletions (produced by the imperfect somatic recombination during TCR formation) are identified relative to the germline gene sequences.

For each successful TCR identification, a key five-part identifier, referred to as a "DCR", is written to an output file. The identifier unambiguously represents a given TCR rearrangement, and consists of: (1) the V index (referencing which V gene was found); (2) the J index (referencing which J gene was found); (3) the number of $3'$ deletions of the V region; (4) the number of $5'$ deletions of the J region; and (5) the "insert" sequence — the sequence found between the ends of the deleted



B

i	1	2	3	4	5	6	7	8	9
f(i)	0	0	0	1	2	0	0	0	6

C

i	output(i)
2	{at}
5	{pat, at}
8	{hog}
9	{path}

Figure 2.5: The three primary functions of the Aho-Corasick algorithm. (A) A directed graph of states built from the set of keywords {*at,pat,hog,path*}, and which defines the goto function. (B) The result of the failure function for each state. (C) The set of words returned by the output function is shown for each of the terminating states. This diagram is based on a figure in the original paper (Aho and Corasick, 1975), but uses an updated example set of keywords.

V and J regions (Thomas *et al.*, 2013). The decombination process is illustrated in Figure 2.4. The DCR identifier is typically accompanied by additional, and non-essential, read-specific information: (6) the FASTQ identifier; (7) the "inter-tag" sequence — the full nucleotide sequence from the start of the V tag to the end of the J tag; (8) the inter-tag sequence quality scores; (9) the barcode sequence; and (10) the barcode sequence quality scores.

2.1.4.1 The Aho-Corasick Algorithm

The Aho-Corasick algorithm performs the construction of a simple, but highly, efficient finite-state machine that searches a given string of text for a defined set of keywords in one pass (Aho and Corasick, 1975). The directed graph structure of the machine is built only once and in advance of the search, and consequently the construction time is proportional to the sum of the lengths of the keywords. All keywords are searched for simultaneously, such that run time is proportional to the sum of the lengths of the keywords, the length of the queried text string, and the number of keywords that are found in the queried string (Thomas *et al.*, 2013). The algorithm can be described in terms of three main functions, and is illustrated with a simple example below.

Consider a finite set of keywords, $K = y_1, y_2, ..., y_k$, where keywords $y_1, ..., y_k$ are text strings. Let x denote an arbitrary text string that we wish to search for keywords. For the sake of this example, let us consider a simple set of four keywords: $K = \{at, pat, hog, path\}$.

We first construct a directed graph of states, s, and associated symbols, σ, by moving in order through K. Therefore, starting from an initial state 0, that maps to itself, we build the path $0 \xrightarrow{a} 1 \xrightarrow{t} 2$ by stepping through the characters of the first keyword, at. This is shown in the first row of Figure 2.5a. For the second and third keyword, pat and hog, we begin construction again from the initial state, 0, and create new routes, $0 \xrightarrow{p} 3 \xrightarrow{a} 4 \xrightarrow{t} 5$ and $0 \xrightarrow{h} 6 \xrightarrow{o} 7 \xrightarrow{g} 8$, illustrated on the second and third rows of Figure 2.5a. For the final keyword, $path$, we note that there is an already constructed route that takes us some way through the word. Consequently, we are left only to define one final state, 9, and complete the route, $0 \xrightarrow{p} 3 \xrightarrow{a} 4 \xrightarrow{t} 5 \xrightarrow{h} 9$.

Goto Function

The goto function, $G(s, \sigma)$, aims to map a state, s, and an associated input symbol, σ, into another state, s'. Following the example in Figure 2.5a, we can see, for instance that $G(0, a) = 1$ or $G(4, t) = 5$. If no mapping is available for a given state and symbol, then the goto function fails, and we consult the failure function. In the provided example, for instance, $G(0, \sigma) = fail \ \forall \ \sigma \notin \{a, p, h\}$.

Failure Function

The failure function, $F(s)$, is consulted upon a failed mapping of the goto function. We first define the depth, d, of a given state s as the length of the shortest path from the initial state to s. In the provided example, states 1, 3 and 6 have depth 1. States 2, 4, and 7 have depth 2, and so on. To construct the failure function, we first set $F(s) = 0$ for all states of depth 1. We next compute the failure function for all states of depth 2, then depth 3, and so on, until all states have been considered. To compute the failure function for state s with $d \geq 1$, we perform the following actions:

1. Let r be the state with $d-1$ along the path from the initial state to s. Let ρ be the symbol such that $G(r,\rho) = s$. First compute $s^* = F(r)$.

2. Iteratively compute $s^* = F(s^*)$ until a state is reached such that $G(s^*,\rho) \neq fail$.

3. Set $F(s) = G(s^*,\rho)$.

Performing this routine for all state results in the function illustrated in Figure 2.5b. As an example, consider calculating the failure function for state $s = 9$. Examining the graph, we determine $r = 5$ and $\rho = h$. In the first iteration, We first compute $s^* = F(r) = F(5) = 2$. We note that $G(s^*,\rho) = G(2,h) = fail$. Consequently, we continue to iterate, and compute $s^* = F(2) = 0$. Now, $G(s^*,\rho) = G(0,h) = 6$. As the goto function has not failed, we finally set $F(9) = s^* = 6$.

Output Function

The output function, H, is constructed alongside the goto and failure functions. We initially consider the output function to return an empty set for any given state. Upon construction of the graph, a keyword is added to the set returned by the output function of the state at which the path terminates. For the given example, the output function returns non-empty sets when $H(2) = \{at\}$, $H(5) = \{pat\}$, $H(8) = \{hog\}$ and $H(9) = \{path\}$. These terminating states are highlighted in orange in Figure 2.5a.

As we construct the failure function, we merge the set $H(s)$ into the set $H(s')$ for $F(s') = s$. For instance, in the given example, the set $H(2) = \{at\}$ is merged with $H(5) = \{pat\}$ when considering the failure function of state 5, $F(5) = 2$. Then the output function for state 5 becomes $H(5) = \{pat,at\}$. The output function for non-empty sets is shown in Figure 2.5c.

Operating Cycle Example

As a full example of an operating cycle, whereby all keywords are located in a given string in one pass, consider the aforementioned keywords $K = \{at, pat, hog, path\}$,

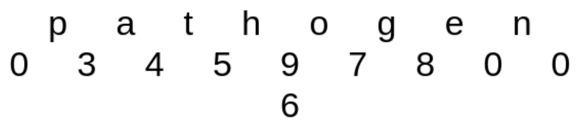


Figure 2.6: Finite state machine transitions for the text string "pathogen" and the keyword trie shown in Figure 2.5.

and the text string $x = pathogen$. The sequence of state transitions is shown in Figure 2.6. The machine begins the search at state $s = 0$ and $\sigma = p$. Consulting the goto function, $G(0, p) = 3$. The output function $H(3)$ is consulted and returns an empty set, indicating no keywords have been found. The machine then considers the transition $G(3, a) = 4$, and so on. Upon reaching state 5, the output function returns $H(5) = \{pat, at\}$. Upon reaching state 9, the output function returns $H(9) = \{path\}$. The goto function then attempts $G(9, o) = fail$, and so the failure function is consulted. $F(9) = 6$, and so the machine proceeds from state 6. The final keyword is returned upon the reaching state 8, as $H(8) = \{hog\}$. Consequently, all keywords within the string *pathogen* have been determined without repeated looping through the graph.

2.1.4.2 Assignment of Sequences with Mismatches

Rather than searching for entire V and J sequences in target read sequences, Decombinator instead uses a set of short V and J tag sequences as its vector of keywords. Each tag is unique to a known V or J sequence, and were chosen via a simple exhaustive search (Thomas *et al.*, 2013). Not only do these shorter keywords improve the efficiency of the algorithm, they also navigate the problem of mismatch between read and V/J sequence due to junctional deletion of nucleotides.

The Aho-Corasick algorithm is a method for solving exact pattern-matching problems. However, error is commonly introduced into DNA or RNA sequence data, both from misassignment of the correct base by the sequencing machine, and through successive rounds of PCR amplification. As a result, the algorithm is modified in Decombinator to account for mismatches. Each V and J tag is subdivided into two half tags, which are not necessarily unique to a given V or J sequence. For target sequences where a tag is not found, the search is repeated using half tags. If

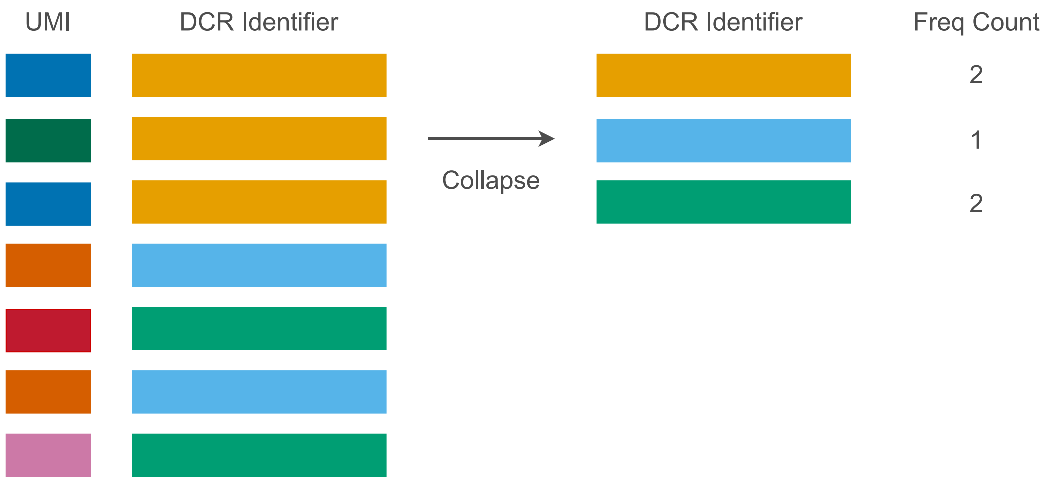


Figure 2.7: The introduction of UMIs prior to PCR are used to account for the uneven amplification of raw sequences. Identical DCR identifiers with identical UMIs are considered to originate from the same TCR, and therefore should not be counted more than once. The Collapsinator script uses the UMI correction process to provide a frequency count of each TCR in the sample. This schematic shows a simplification of the collapsing procedure. In reality, both UMIs and TCR sequences may feature errors introduced during amplification or sequencing. Solutions to these challenges are described further in Section 2.2.3.

a half tag is found, the corresponding full tag sequence is aligned with the target, and the Hamming distance (defined in Appendix A) is computed. If the Hamming distance is equal to a maximum of 1 for, at most, one of the full V and J tags, then an assignment is made using these tags to annotate the target as containing a TCR sequence. This approach maintains a high degree of speed compared to alternative approaches, and achieves an accurate assignment rate. The method is outlined fully in the original Decombinator manuscript by Thomas *et al.* (2013).

2.1.5 Collapsinator Module

The Collapsinator script is used to correct for the effects of sequencing error and PCR amplification heterogeneity in the data format produced by Decombinator (Best *et al.*, 2015). Upon input streaming, each line in the input file (which contains the DCR identifier and barcode information) is quality-checked. Firstly, the barcode is searched for the fixed spacer oligonucleotide sequences that border the UMI hexamers using regular expression patterns. A small number of substitutions, insertions and deletions are allowed when locating the spacers. Lines for which no spacers are suitably identified are skipped. Additionally, barcode sequences containing ambiguous bases (assigned as "N" by the sequencing machine) or containing low average quality (determined from the machine-assigned sequence quality scores) are skipped by default.

After quality control, input DCRs are grouped and clustered by their UMI sequences. This process has undergone much development since. the original publication (Best *et al.*, 2015), and has been outlined in a recent manuscript describing the latest version of the Decombinator pipeline (Peacock *et al.*, 2021). The clustering routine has since been optimised for speed, and a complete overview, including the differences between the old and new methods, is provided in Section 2.2.3.

Once the data has been clustered by UMI sequence, each cluster is "collapsed" to give a UMI–TCR pair that is believed to represent the sequence output from a single initial TCR molecule. Quantitative estimate of TCR clone size can then be estimated by counting the number of identical TCR sequences (or more constructively, identical DCRs) that have different associated UMIs. This principle is illustrated in Figure 2.7 for an ideal case featuring no errors in TCR or UMI sequences.

2.1.6 CDR3translator Module

The CDR3translator script has been primarily designed to read in data produced by Collapsinator or Decombinator, and extract the CDR3 protein sequence of each DCR identifier using predefined sets of sequences for each V and J region.

The CDR3translator script first assembles the full TCR nucleotide sequence from the DCR identifier. This sequence is then translated to protein sequence. The algorithm identifies both "productive" TCR sequences and "non-productive" TCR sequences (sequences containing stop codons, that are out of frame, or do not contain CDR3 motifs) (Oakes *et al.*, 2017). Finally, the CDR3 sequence is extracted by locating gene-specific conserved motifs. A schematic and example translation are illustrated in Figure 2.8.

The CDR3 region is conventionally defined as the region from the position of the second conserved cysteine in the V gene to the phenylalanine in the conserved "FG(X)G" motif in the J gene (where the X amino acid varies across J genes). For

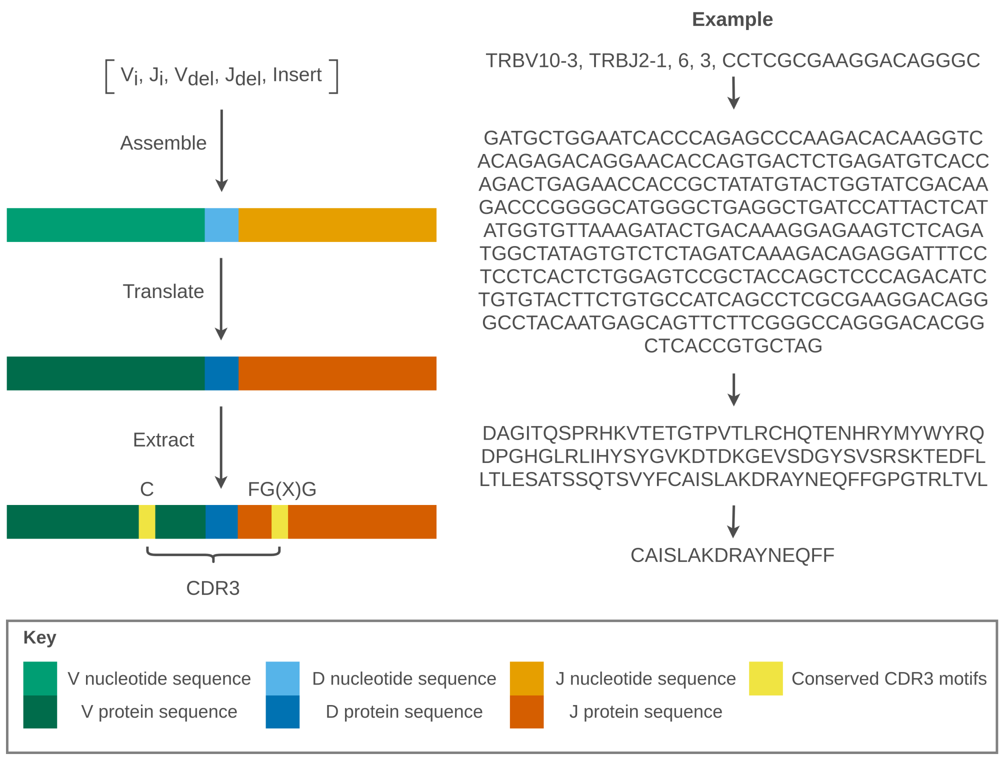

Example
[V_i, J_i, V_del, J_del, Insert]
Assemble
Translate
Extract
C
FG(X)G
CDR3
TRBV10-3, TRBJ2-1, 6, 3, CCTCGCGAAGGACAGGGC
GATGCTGGAATCACCCAGAGCCCAAGACACAAGGTC
ACAGAGACAGGAACACCAGTGACTCTGAGATGTCACC
AGACTGAGAACCACCGCTATATGTACTGGTATCGACAA
GACCCGGGGCATGGGCTGAGGCTGATCCATTACTCAT
ATGGTGTTAAAGATACTGACAAAGGAGAAGTCTCAGA
TGGCTATAGTGTCTCTAGATCAAAGACAGAGGATTTCC
TCCTCACTCTGGAGTCCGCTACCAGCTCCCAGACATC
TGTGTACTTCTGTGCCATCAGCCTCGCGAAGGACAGG
GCCTACAATGAGCAGTTCTTCGGGCCAGGGACACGG
CTCACCGTGCTAG
DAGITQSPRHKVTETGTPVTLRCHQTENHRYMYWYRQ
DPGHGLRLIHYSYGVKDTDKGEVSDGYSVSRSKTEDFL
LTLESATSSQTSVYFCAISLAKDRAYNEQFFGPGTRLTVL
CAISLAKDRAYNEQFF
Key
V nucleotide sequence
V protein sequence
D nucleotide sequence
D protein sequence
J nucleotide sequence
J protein sequence
Conserved CDR3 motifs

Figure 2.8: The CDR3translator script can be used to convert DCR identifiers produced by Decombinator or Collapsinator to CDR3 sequence. The TCR nucleotide sequence is assembled from the DCR and translated to protein sequence. The CDR3 region is then identified by locating the conserved C and FG(X)G motifs. This procedure is shown schematically (*left*), and for an example DCR identifier (*right*). CDR3translator has been recently upgraded to produce additional information, including CDR1 and CDR2 sequences, and is described further in Section 2.2.4.

some genes, however, non-canonical motifs are used, the positions of which vary in the sequence. The CDR3translator script performs a look up to identify the position and type of motif used for a given sequence based on the V and J genes in the DCR identifier. The predefined lookup motif sets that are imported into CDR3translator can be found as the `.translate` files in the Decombinator-Tags-FASTAs repository (`https://github.com/innate2adaptive/Decombinator-Tags-FASTAs`), assembled from the V and J gene germline sequences recorded in the IMGT/GENE-DB database (Giudicelli *et al.*, 2005).

The CDR3translator module has recently been expanded to output data compliant with the standards set by the International Adaptive Immune Receptor Repertoire (AIRR) Community guidelines (Vander Heiden *et al.*, 2018). For the sake of consistency and compatibility, CDR3translator has retained its original name, but now provides a richer range of output data, including CDR1, CDR2 and full TCR sequences. These changes have been incorporated into the latest Decombinator release (Peacock *et al.*, 2021) and are discussed further in Section 2.2.4.

2.2 Decombinator V4

In response to the rapid growth of the TCR sequencing field, the Decombinator pipeline has undergone substantial development since its first publication. The latest major release, Decombinator Version 4 (V4), has been recently described in Peacock *et al.* (2021). This version has been subsequently used to investigate the dynamics of the TCR repertoire in individuals exposed to an organic potent skin sensitizer that is used to treat warts, melanoma, and alopecia areata (Ronel *et al.*, 2021), and to investigate the TCR response to autologous stem cell transplantation in multiple myeloma patients (Lee *et al.*, 2021).

The suite is actively maintained and updated, and represents the collaborative effort of a number of researchers. This section outlines some of the major and minor changes that have been made to improve the pipeline as part of, and since, the Decombinator V4 release. The specific modification of tools and features that have been implemented over the course of this PhD by collaborators are attributed where relevant.

2.2.1 Demultiplexor Compatibility with Unique Dual Index Protocols

The original experimental protocol used in conjunction with the Decombinator pipeline introduced an additional index for demultiplexing samples in combination with the Illumina SP2 index. This protocol is referred to in this thesis as the "internal dual index" (IDI) protocol. The arrangement of the sequencing construct for the IDI protocol has previously been illustrated in Figure 2.2.

The multiplexing method that is used for efficient high-throughput sequencing has been recently shown to produce incorrect sample assignment for a significant

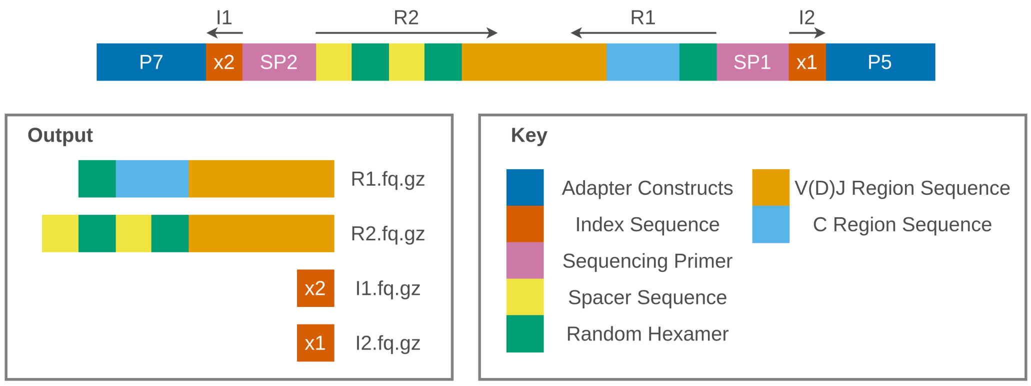


Figure 2.9: Schematic for the sequencing of the TCR construct assembled through the new UDI experimental protocol for use with Decombinator V4. Both `x1` and `x2` index are used by Demultiplexor to separate reads into sample specific files, while the spacer sequences and random hexamers make up the molecular barcode.

number of subsequently demultiplexed reads (Sinha *et al.*, 2017; Costello *et al.*, 2018). While a number of mechanisms can produce this effect, sample "index hopping" has been shown to be the primary cause (Farouni *et al.*, 2020).

During multiplexing, sequence constructs are pooled together. A percentage of index primers that were used to create these constructs do not bind to any other fragments, and so continue to float freely in the library pool. These free index primers may anneal to existing library molecules in the pool. Active DNA polymerase extends the annealed fragments to form a molecule with a switched index. This process has been termed "index hopping" (or "index swapping" or "index switching"). The synthesised strand separates and is amplified through rounds of PCR amplification (Sinha *et al.*, 2017).

The effects of index hopping have been shown to be effectively eliminated when using unique dual index (UDI) protocols (MacConaill *et al.*, 2018). When using two unique indices per sample, misassignment of samples will only occur in the unlikely event that both indices hop (Farouni *et al.*, 2020).

In accordance with this research, index hopping ambiguity leads to Demultiplexor assigning a given read to the wrong sample file for IDI protocols. The downstream effects of this are noticeable in the apparent sharing of TCR clones between unrelated samples. Given the relative rarity of public TCRs, the biological likelihood of finding such repertoire overlap between samples is very low.

To address the artefacts of artificial overlap, the Demultiplexor module has since been upgraded to support data produced using UDI protocols. The internal SP1 index is removed from the protocol, and two unique Illumina indices are introduced for each sample. The arrangement of the sequencing construct for the UDI protocol is illustrated in Figure 2.9.

Examples of the level of artificial overlap between the two protocols are shown in Figure 2.10. Figure 2.10A shows the overlap effect on a sequencing run using an IDI protocol, and Figure 2.10B the overlap effect on a similar sequencing run using a UDI protocol. Samples names are shown along each axis, with the colour bar denoting the amount of overlap between a given pair of samples. The sample names in these plots are composed of: (1) an experiment ID, (2) a patient ID, (3) a time point or sample type ID, and, for certain samples, (4) a label identifying the sample as a repeat of a previous run, all separated by underscore characters. The first two identifiers are most important when examining the overlap. Figure 2.10A shows apparent TCR sharing between samples from disparate experiments and patients. Figure 2.10B also shows TCR sharing, but only between samples from the same patient at different time points. The dual index protocol, therefore, largely eliminates the effects of artefactual sample mixing overlap. These plots show the overlap for the TCR α chain of each run. The results for the β chain are similar, and can be found in Appendix B.

The overlap measure shown in these plots is calculated in the following way: for an overlap matrix, A, the overlap between two samples, i and j, with respect to i, is calculated as,

$$A_{ij} = \frac{\text{Number of distinct TCRs found in both } i \text{ and } j}{\text{number of unique TCRs in } i} \tag{2.1}$$

where the 5-part DCR identifier is used to represent each TCR. It should be noted that A is an asymmetric matrix, and in general A_{ij} is not equivalent to A_{ji}.

The overlap calculation above was devised and implemented as an R script by Dr Tahel Ronel. The script, which was used to generate the plots in Figure 2.10, is

Spacer	Nucleotide sequence
I8	ATCACGAC
M13	CCAGGGTTTTCCCAGTCACGAC

Table 2.1: I8 and M13 spacer nucleotide sequences.

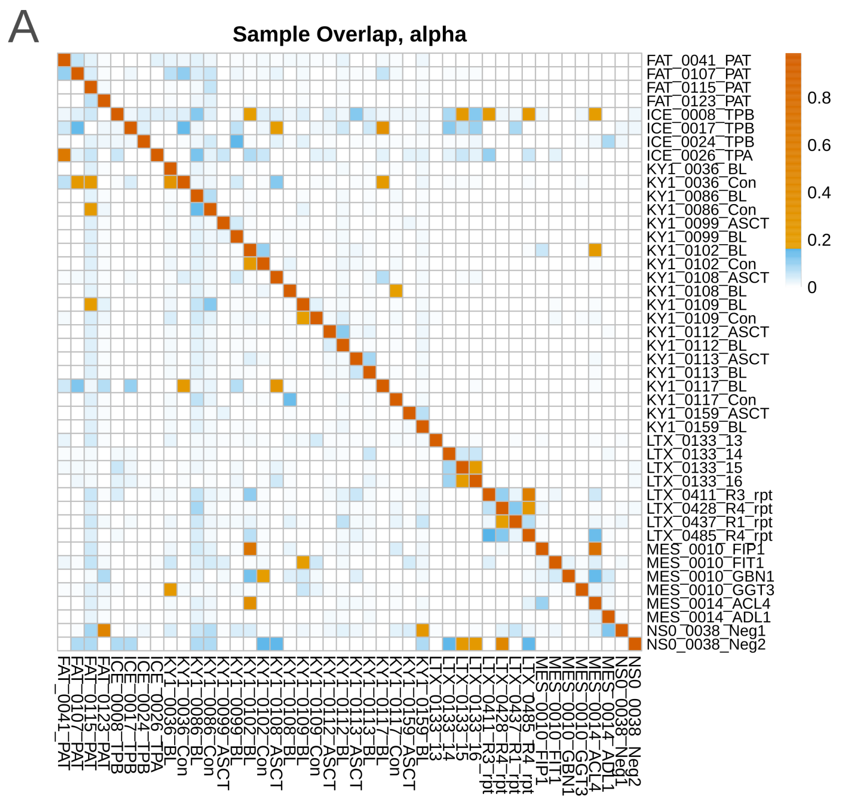
available in the Decombinator-Tools repository (`https://github.com/innate2adaptive/Decombinator-Tools`).

2.2.2 Support for Multiple Oligonucleotide Barcoding Protocols

Molecular barcoding is critical for accurate quantification of TCR clones in PCR amplified data. The original experimental protocol for use with the Decombinator pipeline used a barcode schema composed of: (1) an I8 spacer, (2) a random hexamer, (3) a second I8 spacer, and (4) a second random hexamer. The Collapsinator script searches for the two spacer sequences in order to locate the random hexamers, which, when extracted and joined together, form the 12 base pair UMI.

A more robust barcode schema has been designed since, and is composed of: (1) an M13 spacer, (2) a random hexamer, (3) an I8 spacer, and (4) a second random hexamer. This schema is now assumed as default by Collapsinator, however the module has been refactored to also retain compatibility with the original barcoding protocol. Running Collapsinator for the either I8 or M13 data can be specified through an input argument. The task of implementing a new oligo is now fairly trivial, and while modification is required in code, only minimal changes are required to the `getOligo` function. Extraction to an input configuration file could be implemented easily into a future iteration.

The sequences for the I8 and M13 spacers are shown in Table 2.1 (and their reverse complements are implemented in Collapsinator). Collapsinator searches for these sequences, using regular expressions, to identify the location of the random hexamers that make up the UMI. A small number of substitutions, insertions and deletions are allowed in each spacer. Previous iterations of the Collapsinator fuzzy matching method would occasionally filter out reads with insertions or deletions detected in the spacers more frequently than necessary. This has been improved in



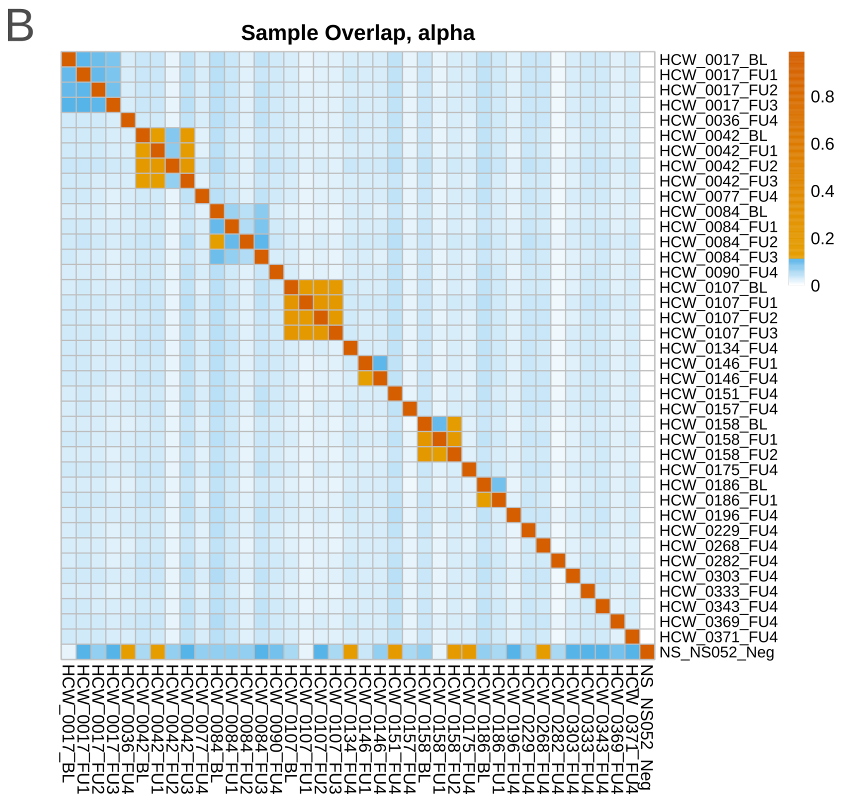


Figure 2.10: A) TCR overlap between patients and experiments for α chain data from an IDI protocol NextSeq run. (B) TCR overlap for α chain data from a UDI protocol NextSeq run. Rows and columns with suffixes "Neg", "Neg1" and"Neg2" provide negative controls.

the V4 release, increasing the number of usable reads.

To support the fuzzy matching method, a large number of unit tests have been written for both the M13 and I8 protocols. These tests cover a range of cases to ensure the correct analysis of spacers with minor errors. Details for running the test suite are provided online as part of the Decombinator pipeline README (`https://github.com/innate2adaptive/Decombinator#collapsinator-test-suite`).

2.2.3 Improved Accuracy of UMI Clustering

The Collapsinator script is used to correct errors introduced to the data through PCR amplification and through misassignment of bases by the sequencing machine. The algorithm has been re-designed for the Decombinator V4 release, and has been shown to significantly improve the accuracy of the pipeline (Peacock *et al.*, 2021).

The original Collapsinator algorithm is outlined by example in Figure 2.11. (1) The first step in the diagram shows three TCRs, each with a unique UMI. The TCRs are identical, such that the clone size equals 3. (2) The second step shows the effects of PCR amplification and sequencing, whereby errors are introduced into both the TCR sequences and the UMIs. (3) The third step demonstrates how the algorithm reads in the barcoded data. Input data is rapidly streamed into a Python dictionary (hash table), with reads (values) indexed by unique UMIs (keys). The data is, therefore, initially grouped by unique UMI. (4) The fourth step shows the results of collapsing and regrouping the data. Levenshtein distances (defined in Appendix A) are computed between TCR sequences in each UMI group. Those sequences with Levenshtein distance below a set threshold are considered to be identical, and are consequently collapsed, with the most populous TCR sequence taken as the representative sequence. The data is subsequently re-grouped, such that it is indexed by TCR (or, more accurately, by DCR identifier). (5) In the fifth step, the Levenshtein distance is computed for all UMIs in each group. Those within a set threshold are considered to be identical UMIs and are clustered together, to complete the error correction process. (6) Finally, the sixth step provides a count of how many UMIs are associated with each TCR. A clone size of 3 is recovered for the

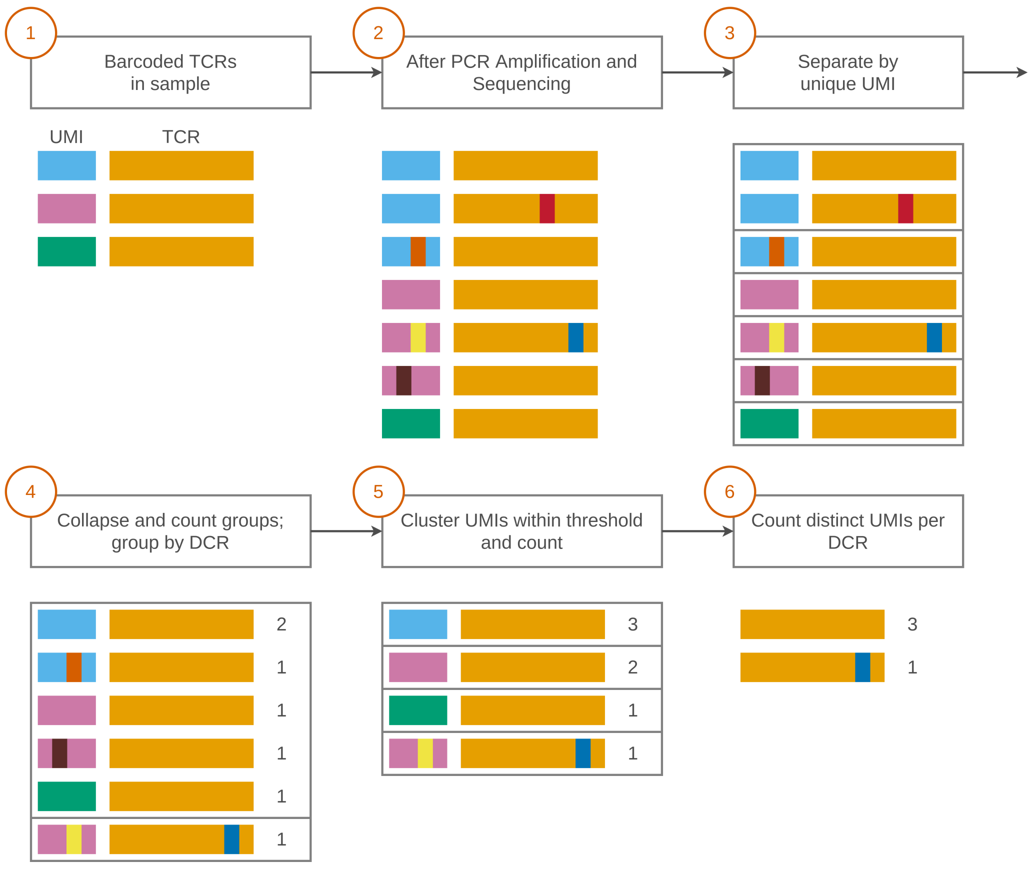

1
Barcoded TCRs in sample
UMI
TCR
2
After PCR Amplification and Sequencing
3
Separate by unique UMI
4
Collapse and count groups; group by DCR
2
1
1
1
1
1
5
Cluster UMIs within threshold and count
3
2
1
1
6
Count distinct UMIs per DCR
3
1

Figure 2.11: The outdated algorithm for clustering and collapsing TCR data in legacy Collapsinator versions. (1) Three identical TCRs with unique UMIs, representing a clone size of 3. (2) Errors are introduced into the UMI and TCR sequences through PCR amplification and sequencing. (3) Reads are initially grouped by unique UMI. (4) The UMI groups are collapsed, and the data is re-organised to group by identical TCR — or, more specifically, by DCR identifier. (5) Within each DCR group, UMIs are clustered by their similarity and collapsed. (6) A count is provided of the number of unique UMIs that are associated with a specific TCR (DCR identifier). The TCR with clone size of 3 is recovered, but the algorithm introduces an additional erroneous sequence.

original TCR present in the first step. However, the same TCR sequence but containing sequencing error is incorrectly annotated as a separate TCR. This occurs when errors are present in both UMI and sequence, and through the independent handling of UMI and sequence by the algorithm. This phenomenon was the main contributor to Collapsinator inaccuracy in legacy versions of the Decombinator pipeline.

Figure 2.12 shows the same example but processed by the re-designed Collapsinator algorithm of Decombinator V4. (1-2) The first and second steps show the

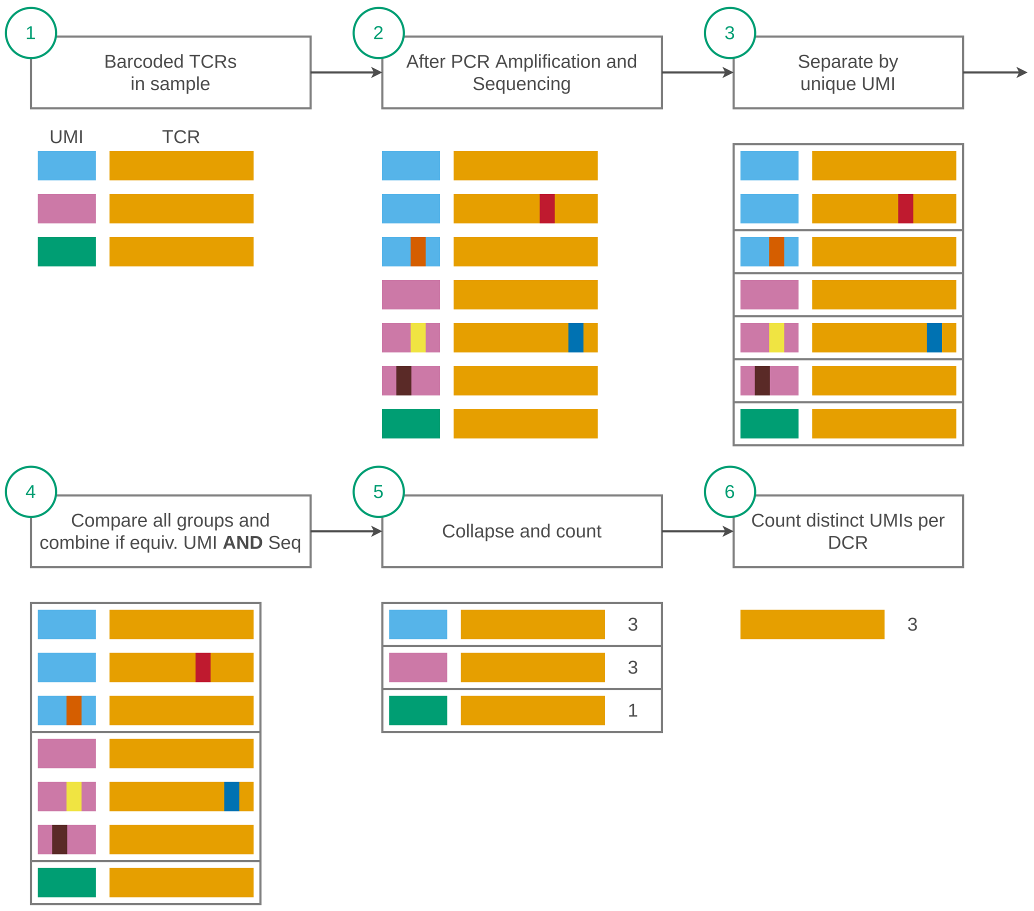

1
Barcoded TCRs in sample
UMI
TCR
2
After PCR Amplification and Sequencing
3
Separate by unique UMI
4
Compare all groups and combine if equiv. UMI **AND** Seq
5
Collapse and count
3
3
1
6
Count distinct UMIs per DCR
3

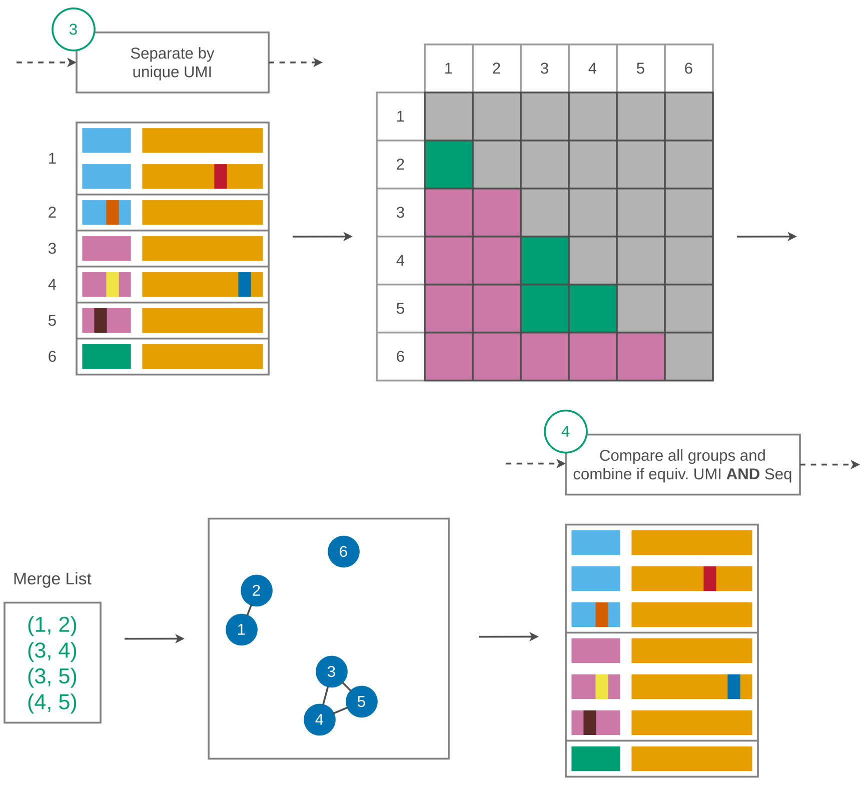
Figure 2.12: Decombinator V4 features a re-designed algorithm for clustering and collapsing TCR data. (1) Three identical TCRs with unique UMIs, representing a clone size of 3. (2) Errors are introduced into the UMI and TCR sequences through PCR amplification and sequencing. (3) Reads are initially grouped by unique UMI. (4) Initial groups are merged if both UMI and sequence meet threshold criteria. (5) Each cluster is collapsed, with the cluster size providing a measure of PCR heterogeneity. (6) A count is provided of the number of unique UMIs that are associated with a specific TCR (DCR identifier). The TCR with clone size of 3 is recovered, and no erroneous sequences are introduced.

same barcoded TCRs before and after PCR amplification and sequencing. (3) The third step shows the initial grouping of data by unique UMI, as before. (4) In the fourth step, both the UMI and sequences of each group are compared to that of every other group. Groups are combined if both the Levenshtein distance between UMIs and the Levenshtein distance between sequences are below set thresholds. (5) In the fifth step, each group is collapsed, providing a count of how many sequences contribute to each cluster. (6) Finally, the sixth step provides a count of how many

UMIs are associated with each TCR, and the TCR clone size of 3 is recovered. The consideration of both UMI and sequence simultaneously by the V4 algorithm avoids the artefacts produced by the old algorithm.

A simulated data set was constructed to test the improvement in accuracy of the new algorithm. An initial set of artificial TCR sequences was created using the probabilistic repertoire generation software package, IGoR (Marcou *et al.*, 2018). Individual TCRs were given different abundances based on the observed long-tailed distributions observed in experimental data sets (Oakes *et al.*, 2017). A 12 base pair UMI was stochastically generated for each TCR sequence to produce an initial artificial set of 10,000 TCR-UMI combinations. PCR amplification was simulated by duplicating sequences over a number of iterations, introducing sequence errors at a rate of 5×10^{-6} (somewhat higher than that which has been estimated for the polymerase used in the experimental protocol), to produce a data set of approximately 2.8 million sequences. Finally, NextSeq sequencing errors and base pair quality scores were simulated using the ART Illumina simulator (Huang *et al.*, 2012). Across 10 simulated runs, all but 2 ± 1 (mean $\pm$ standard deviation) sequences were recovered by the new Collapsinator algorithm. 109 ± 15 sequences introduced by PCR amplification or sequencing error were incorrectly retained by the new algorithm. The correlation coefficient between the sequence abundances in the initial simulated set and the Collapsinator output was 0.97 ± 0.7. In contrast, the old Collapsinator algorithm introduced 5855 ± 335 erroneous sequences, and the correlation coefficient was measured as 0.86 ± 0.006. Generation of the simulated data and Collapsinator benchmarking was carried out by Professor Benny Chain.

The main drawback of the new algorithm is the increased run time associated with comparing each UMI and TCR sequence simultaneously before collapsing. This requires a pairwise comparison of the initial unique UMI groups of sequences, rather than the rapid hash mapping technique of the old algorithm. Figure 2.13A illustrates the pairwise comparison step and efforts designed to optimise for speed and memory. The comparison of groups can be visualised as a 2D symmetric matrix. It is sufficient to compute only the off-diagonal elements, A



B

File	dcr_Test-beta2.n12.gz
No. Input Reads	81149
No. Passing Quality Filters	44782
No. Initial Groups	25238
No. Clusters	24203
No. Clusters with 1 input read	14338

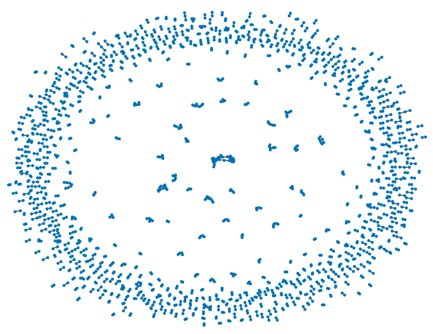


Figure 2.13: (A) The most expensive part of the new Collapsinator algorithm is the pair-wise comparison of UMIs and sequences of all initial UMI groups. Run time can be reduced by considering only off-diagonal pairs. Memory usage can be reduced by storing a merge list of IDs pertaining to only those groups that should be merged. The merge list can be visualised as a network of disconnected subgraphs. Extraction of the subgraphs amounts to clustering the data. (B) The number of input reads, initial groups and clusters are shown for one of the Decombinator test data samples. Clusters of size greater than 1 are visualised for reference.

reducing the number of calculations from N^2 to $\frac{N(N-1)}{2}$, with N the number of initial groups. A merge list is created, storing pairs of IDs of each group that meet the similarity criteria. Storing only IDs — and, moreover, only the IDs of groups that require merging — avoids overloading the memory with vast arrays of data that would typically crash the software for large datasets. If the initial UMI groups are abstracted as nodes in a network, tuples in the merge list describe the edges between them. Clusters can then be extracted as disconnected subgraphs from the main network. The `networkx` Python package has been introduced as a Collapsinator dependency to quickly and effectively perform this routine. Figure 2.13B provides some values for the number of reads, initial groups and clusters involved in collapsing data from the Decombinator test data (which is considerably smaller than production runs), which can be found in the test data repository (`https://github.com/innate2adaptive/Decombinator-Test-Data`). The network of clusters of size greater than 1 is also visualised.

Finally, estimates of the effect of heterogeneous PCR amplification can be made by measuring the size of a given cluster. This information is now included in the default output by Collapsinator, and carried through the rest of the pipeline to be reported in the final CDR3translator output. Furthermore, an optional input argument can be used to output an additional `.csv` file of cluster size data. If used, tailored commands are provided to the user with instructions of how to generate histograms of the data automatically using the `UMIHistogram` script in the Decombinator-Tools repository.

2.2.4 AIRR Community Compliance

The Decombinator pipeline is now compliant with the standards outlined by the Adaptive Immune Receptor Repertoire (AIRR) Community of The Antibody Society for the acquisition, storage, annotation, or sharing of associated AIRR-seq data sets (Rubelt *et al.*, 2017).

The final output data of the pipeline, produced by the CDR3translator module, is written as a tab-separated values (`.tsv`) formatted file with the required schema for compatibility with other compliant immune-repertoire tools (Vander Heiden

Field	Description
sequence_id	A unique identifier for the TCR rearrangement
v_call	TCR V gene
d_call	Blank required field
j_call	TCR J gene
junction_aa	CDR3 junction amino acid sequence
duplicate_count	TCR abundance
sequence	Inferred full-length variable domain nucleotide sequence
junction	CDR3 junction nucleotide sequence
decombinator_id	Five-field Decombinator identifier
rev_comp	True/False (T/F) flag for whether rearrangement is reverse complemented
productive	True/False (T/F) flag for whether rearrangement is productive
sequence_aa	Inferred full-length variable domain amino acid sequence
cdr1_aa	CDR1 amino acid sequence of the used V gene
cdr2_aa	CDR2 amino acid sequence of the used V gene
vj_in_frame	True/False (T/F) flag for whether rearrangement is in frame
stop_codon	True/False (T/F) flag for whether rearrangement contains a stop codon
conserved_c	True/False (T/F) flag for whether rearrangement contains a conserved cysteine
conserved_f	True/False (T/F) flag for whether rearrangement contains a conserved phenylalanine (or equivalent)
legacy_v_call	V gene as referred to by older versions of Decombinator (i.e. $\leq$ v3)
legacy_j_call	J gene as referred to by older versions of Decombinator (i.e. $\leq$ v3)
v_alleles	List of V gene alleles covered by used V tag
j_alleles	List of J gene alleles covered by used J tag
v_gene_functionality	IMGT predicted functionality of V gene (or genes) used in this rearrangement (F/ORF/P, comma delimited)
j_gene_functionality	IMGT predicted functionality of J gene (or genes) used in this rearrangement (F/ORF/P, comma delimited)
sequence_alignment	Blank required field
germline_alignment	Blank required field
v_cigar	Blank required field
d_cigar	Blank required field
j_cigar	Blank required field
av_UMI_cluster_size	Number of identical sequences associated with a single UMI

Table 2.2: Fields that feature in the AIRR Community compliant output format produced by CDR3translator, as displayed in the Decombinator repository README.

et al., 2018). All fields present in CDR3translator output data are listed in Table 2.2 alongside complementary descriptions. Output data now includes full nucleotide and amino acid sequences, and CDR1, CDR2 and CDR3 subsequences. TCR rearrangements are labelled as either productive or non-productive. Mandatory fields that are irrelevant to Decombinator are left blank. A number of Decombinator-specific fields, including the DCR identifier and the average UMI cluster size, have

Script name	Description	Status	Author	Editor
`collapsed_sample_overlap.R`	Measure collapsed sample overlap	new	TR	
`DCRtoGeneName.py`	Convert V/J IDs to gene names	existing	JH	TP
`ExactSearch.py`	Exact subsequence search	new	TP	
`ExactSearchLogSummary.py`	Summarise exact search results	new	TP	
`LogSummary.py`	Collates log data	existing	MI	TP
`RandomlySample.py`	Sub-sample Decombinator data	existing	JH	TP
`RunTestData.py`	Run Decombinator test data	new	TP	
`SortSummary.py`	Sort log summary file	new	TP	
`TestDataGenerator.py`	Generate Decombinator test data	new	TP	
`UMIHistogram.py`	Plot UMI distribution	new	TP	
`Recipes`	Templates for cluster jobs	new	BC, TP	

Table 2.3: A summary of the available tools in the Decombinator Tools repository. Tools with status "existing" are old tools that were upgraded by the "Editor" for compatability with the Decombinator V4 release. Tools with status "new" are tools first released alongside Decombinator V4. "Author" and "Editor" initials represent collaborators: Dr Tahel Ronel (TR), Dr James Heather (JH), Thomas Peacock (TP; thesis author), Dr Mazlina Ismail (MI) and Professor Benny Chain (BC).

been added as permitted by the AIRR Community schema. These adaptions to CDR3translator were conducted in collaboration with Dr James Heather.

2.2.5 Python 3

The Decombinator pipeline was originally written in Python 2.7, support for which ended on January 1st 2020. Consequently, all main Decombinator scripts, as well as the majority of support scripts in the Decombinator-Tools repository, have been upgraded to Python 3.7.

It is recommended that the Decombinator pipeline is run within a virtual environment to avoid potential package dependency clash with other installed Python projects. In particular, instructions for setting up and installing Decombinator through a Conda environment either locally or on one of the UCL computing clusters are provided in the Decombinator README.

2.3 Decombinator Tools

The Decombinator-Tools repository (`https://github.com/innate2adaptive/Decombinator-Tools`) has been created to host a number of scripts that may be of use when working with the Decombinator pipeline or the data it produces. These scripts are summarised in Table 2.3, and an overview of each script is provided in Appendix D.

2.4 Single Tag Decombinator for Single Cell Analysis

A weakness of bulk sequencing pooled immune cell populations has always been the lack of information about which α chain pairs with which β chain, limiting descriptions of individual T cells and their interaction with antigen. Recently, solutions to this problem have emerged through the increased availability of single cell sequencing technology (Hwang *et al.*, 2018). It might be expected that an individual T cell expresses a single TCR on its cell surface. However, as has been reviewed by Schuldt and Binstadt (2019), studies have revealed that approximately 10% and 1% of $\alpha\beta$ T cells overcome allelic exclusion to express dual surface α chains, and dual surface β chains, respectively. Nevertheless, in many cases the sequencing of a single T cell offers the opportunity to pair α and β chains to resolve the full composition of the TCR, offering much greater insight into immune function than traditional methods.

The sequencing of single T cells typically produces fragmented, short-read data (De Simone *et al.*, 2018). These fragmented reads most frequently feature only part of the TCR V region sequence or part of the TCR J region sequence. This presents a problem for computational TCR analysis methods, which generally rely on the alignment of reads to a reference sequence and the subsequent inference of the CDR3 junction sequence. The Decombinator pipeline is no exception — we might expect to find a 20 base pair V tag or 20 base pair J tag in a single read, but we are unlikely to find both. Consequently, a certain amount of *de novo* assembly must be performed to reconstruct full TCR sequences.

An adaptation of the Decominator module has been designed to analyse short-read fragmented data for use with single cell data. This approach allows reads featuring only a single V or J tag to persist through to the output data, rather than requiring both. Reads featuring V tags are aligned with reads featuring J tags (using the Biopython `pairwise2` dynamic programming algorithm) in an attempt to locate overlap between the ends of the reads, and so reconstruct the TCR sequence.

Single Tag Decombinator began as a Masters research project prior to the start of this PhD. During this time, the single tag search was completed and a rudimentary overlap identifier was designed, with some manual reassembly required. Since the original project, the overlap identifer and reconstruction steps have been improved and automated, and the software has been tested with larger datasets. Single Tag Decombinator has been used to determine TCRs expressed by neoantigen-reactive T cells (NARTs) in the context of non-small-cell lung cancer (NSCLC) (Joshi *et al.*, 2019). The following sections detail the various steps of the Single Tag Decombinator pipeline and provide a comparison between Single Tag Decombinator and the popular TraCeR (Stubbington *et al.*, 2016) and MiXCR (Bolotin *et al.*, 2015) TCR sequence analysis programs for two independent single cell datasets.

2.4.1 Pipeline

The original Decombinator pipeline was not designed with a single tag approach in mind. Consequently, Single Tag Decombinator sits as a separate module outside the main pipeline, although a large proportion of code is reused. Figure 2.14 shows the various steps of the Single Tag Pipeline that performs the end-to-end annotation of TCRs from short-read data. This procedure can be run as a single script (`SingleTagPipeline.py`) from the Single-Tag-Decombinator repository (`https://github.com/innate2adaptive/Single-Tag-Decombinator`). Each step in this pipeline is outlined below.

Single Tag Decombinator

The Single Tag Decombinator module is an adaption of the Decombinator V3 module, and identifies reads with a single V or single J tag. Rather than producing the traditional five-part DCR identifier, the Single Tag Decombinator approach produces a list of three key elements:

$$(V_i, J_i, S_{extratag}) \tag{2.2}$$

where V_i and J_i are the V and J indices (if found, otherwise “N/A”), and $S_{extratag}$

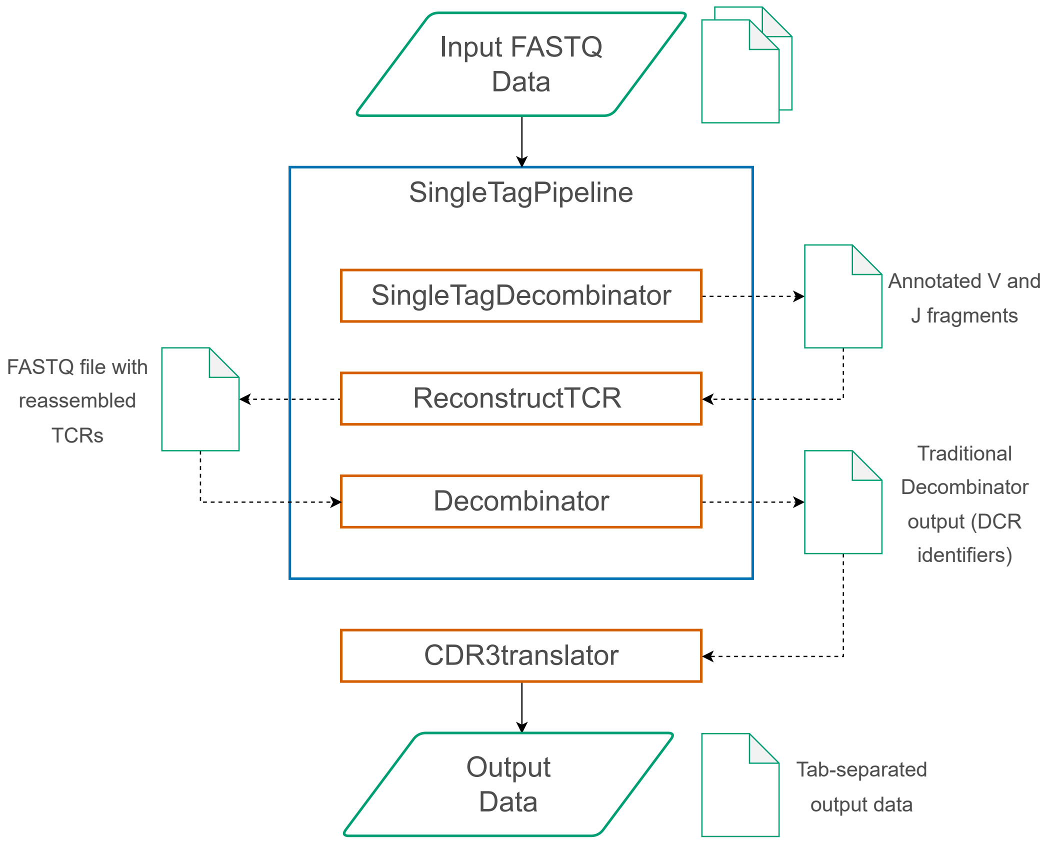

Input FASTQ Data
SingleTagPipeline
SingleTagDecombinator
ReconstructTCR
Decombinator
Annotated V and J fragments
FASTQ file with reassembled TCRs
Traditional Decombinator output (DCR identifiers)
CDR3translator
Output Data
Tab-separated output data

Figure 2.14: Overview of the Single Tag Decombinator Pipeline. Up to two FASTQ files can be provided to the SingleTagPipeline script, along with input arguments to specify search orientation (forward, reverse or both) and chains (α, β, γ, δ, or a combination). The SingleTagPipeline script will automatically run (1) SingleTagDecombinator to search for V and J fragments, (2) ReconstructTCR to pair V and J fragments and reassemble TCR reads into new FASTQ files, and (3) Decombinator to produce the usual DCR identifier output. This output can then be used with CDR3translator to extract the full information about the identified TCR sequences.

is the subsequence from the end of the V tag to the end of the read if a V tag is identified, or the subsequence from the start of the read to the start of the J tag if a J tag is identified.

For single cell data, it is not always clear whether TCR fragment matches will be found in forward or reverse order in the files produced by the machine. Additionally, α and β chain data are unlikely to be separated. Single Tag Decombinator includes functionality to search reads for both α and β chain tags, in both forward and reverse directions, simultaneously. Users may also provide two FASTQ files (for example, the forward and reverse read files from the sequencing machine), rather than one, to be searched in succession. These modifications allow users to search for as many TCR fragment matches as possible by running Single Tag Decombinator only once per cell, while also noticeably improving run time.

Reconstruct TCR Reads

The output TCR fragments identified by Single Tag Decombinator are aligned and by the `reconstructTCR` script to locate overlaps. The aim of this algorithm is to pair as many fragments as possible containing a V tag (V fragment) to fragments containing a J tag (J fragment). V and J fragments are exclusive — that is to say, a V fragment can be paired with one, and only one, J fragment, and *vice versa*. These pairings are prioritised based on the length of overlap and the number of mismatches the overlap contains. The process is illustrated in Figure 2.15 and described below.

Each V fragment is compared with each J fragment to identify overlapping regions between the ends of the fragments. An overlap must meet a set minimum length to be considered. The end of the V fragment of the minimum length is searched for in each J fragment. If detected, a pairwise sequence alignment is performed to identify the full length of the overlap and to calculate a quality score based on the number of mismatches and gaps.

For each V fragment, those J fragments that meet the overlap criteria are sorted into a ranked list. J fragments are first sorted by the longest overlap that they share with the V fragment. The longer the overlap, the higher the ranking. J fragments that have the same length overlap are then sub-ranked by their "purity". The purity is here defined as the score produced by the pairwise sequence alignment of the fragments subtracted from the length of the overlap. The closer the purity is to a value of zero, the better the ranking.

Once each V fragment has a ranked list of J fragments, the V fragments are sorted by the length of the overlap they share with their top ranking J fragment. The highest ranking V fragment gets its "first choice" of J fragments. The two are paired and then excluded from the pool. Next, the second highest ranking V fragment gets its first choice of J fragments, so long as that J fragment has not already been

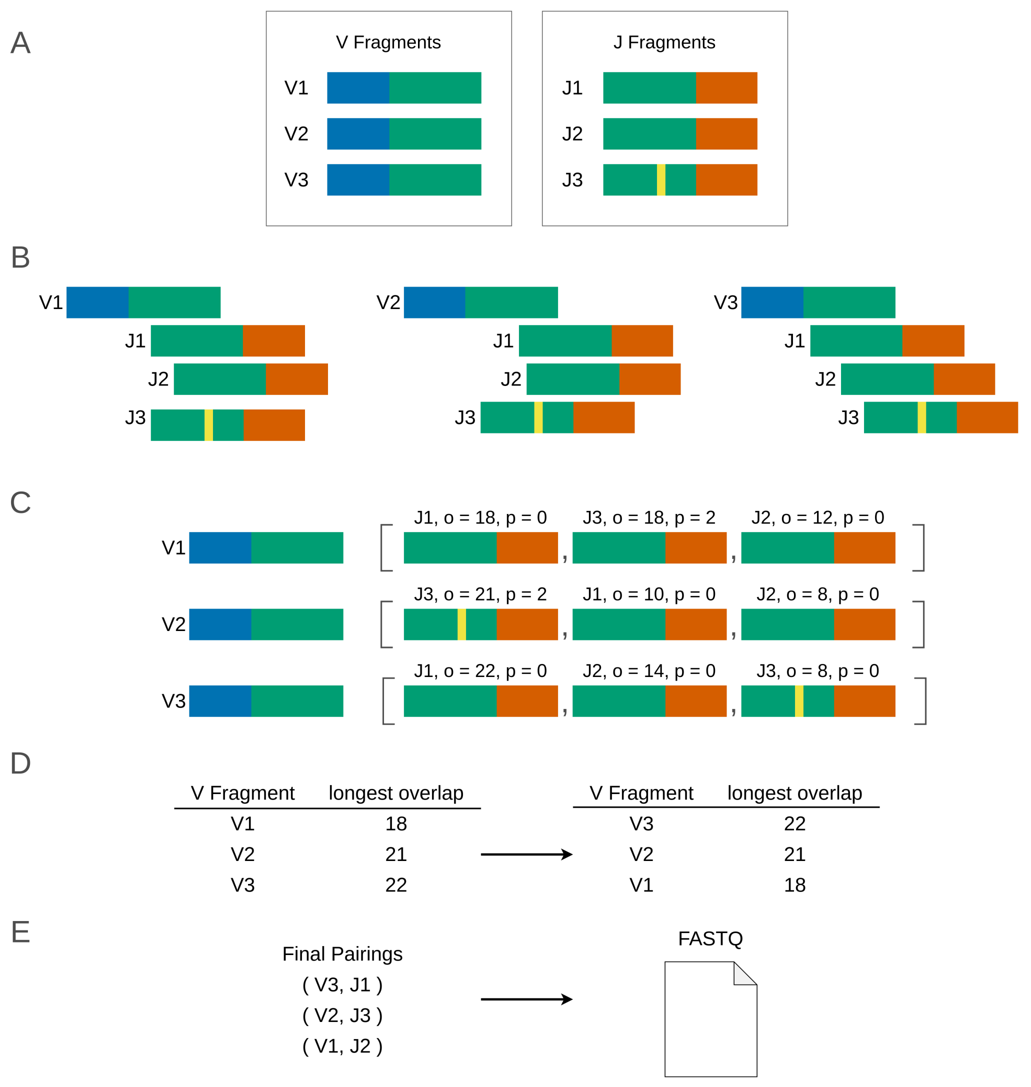


Figure 2.15: (A) Fragments with identified V tags (blue) or J tags (orange) are considered for TCR reassembly. (B) For each V fragment, J fragments are aligned to search for overlap between the ends of the reads. (C) A ranked list of J fragments is produced for each V fragments. J fragments are sorted by the maximum overlap ("o") with the V fragment. J fragments that have the same length overlap are sorted by "purity" ("p"), where the closer the value to zero, the higher the ranking. (D) V fragments are sorted by which has the longest overlap with a J fragment. The top V fragment pairs with its top J fragment. The second top V fragment pairs with its top J fragment, provided that that J fragment does not already have a pair — otherwise, the second top J fragment is chosen, and so on. (E) The final pairings are reassembled to construct sequences containing the V tag, the J tag, and the insert (overlap) sequence, and are written to FASTQ format.

excluded. If its top candidate has already been excluded, the V fragment receives its second choice, and so on. This process continues until all V fragments have been matched.

Finally, reassembled sequences of V tag, J tag, and the overlapping insert sequence are written to FASTQ file, along with the relevant sequence quality scores produced by the HTS machine.

The `TestDataGenerator.py` script, described in Figure 2.3, was written to produce test data for Single Tag Decombinator in order to evaluate the performance of the algorithm. No formal benchmarking was recorded using the generated test data, so high accuracy can be only reported anecdotally. However, the performance of Single Tag Decombinator is shown to be competitive with two other analysis platforms in sections Section 2.4.2 and Section 2.4.3.

Decombinator

As FASTQ reads produced by the fragment reassembly script contain both a V and J tag by construction, they can be easily analysed using the usual Decombinator pipeline. The usually low number of reassembled reads makes this process trivial in terms of computational run time. Five-part DCR identifiers are produced for each read, which can be analysed by the rest of the Decombinator pipeline in the traditional way.

CDR3translator

Finally, the identifiers for the reconstructed TCRs can be provided to the CDR3translator script to extract full information about their sequences.

2.4.2 Application in Non-small-cell Lung Cancer and Benchmarking against TraCeR

One of the earliest tools developed to identify paired α and β chains was the TraCeR software package (Stubbington *et al.*, 2016). TraCeR and Single Tag Decombinator were compared in their abilities to reassemble single cell fragments and identify TCRs in response to neoantigen in a non-small-cell lung cancer dataset. RNA data from 139 cells were analysed using both platforms. A crash on the supercomputer being used to analyse data with Single Tag Decombinator meant that results for one cell became corrupted, and the analysis was not re-run. Consequently, the analysis presented here features a total of 138 cells, 43 of which were labelled as neoantigen reactive T cell (NART) positive, and 95 of which were labelled as NART negative. This dataset forms part of the TRACERx project (not to be confused with the unrelated TraCeR software) for studying cancer evolution (Jamal-Hanjani *et al.*, 2017). The analysis of cells using TraCeR, and the mapping of TraCeR output to a Decombinator-style output (DCR identifiers), was performed by Dr Mazlina Ismail.

Figure 2.16 shows a comparison of TCR α chains identified by Single Tag Decombinator and TraCeR across the 138 cells. A TCR chain is here represented by the Decombinator DCR identifier. Hits in green represent DCRs identified by both TraCeR and Single Tag Decombinator. Hits in blue represent DCRs identified by TraCeR but not by Single Decombinator, while hits in orange represent DCRs identified by Single Tag Decombinator but not by TraCeR. Figure 2.17 shows a similar plot of identified β chains. To aid readability, the axes are labelled by an ID for each cell and for each DCR. Full mapping of IDs to DCRs and IDs to cells are provided in Appendix Table C.1, Appendix Table C.2 and Appendix Table C.3.

At least one TCR α chain was identified by either of the two platforms in 47.8% of cells. Single Tag Decombinator found at least one TCR α chain in 47.8% of cells, and TraCeR found at least one TCR α chain in 44.2% of cells. For no cell was an α chain identified by TraCeR but not by Single Tag Decombinator. However, for 3.62% of cells, an α chain was identified by Single Tag Decombinator but not by TraCeR.

At least one TCR β chain was identified by either of the two platforms in 59.4% of cells. Single Tag Decombinator found at least one TCR β chain in 59.4% of cells, and TraCeR found at least one TCR β chain in 49.3% of cells. Again, for no cell was a β chain identified by TraCeR but not by Single Tag Decombinator. However, for 10.1% of cells, a β chain was identified by Single Tag Decombinator but not by TraCeR.

For cells where at least one TCR α chain was identified, the same TCR was identified by Single Tag Decombinator and TraCeR in 91.0% of cases. For cells where at least one TCR β chain was identified, the same TCR was identified by Single Tag Decombinator and TraCeR in 81.7% of cases.

The Jaccard index (or Tanimoto index) provides a measure of similarity and diversity between sets, and for two sets A and B, is calculated as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.3}$$

The TCRs identified by the two approaches are more similar for the α chain data, which has a Jaccard index of 0.83. The β chain data has a Jaccard index of 0.68, perhaps implying that β chains (which notably contain an additional D gene) are more difficult to identify unambiguously than α chains. Looking closely at the DCR indexes in Table C.1 and Table C.2 in Appendix C, it is likely that these values provide a lower bound of TCR similarity. In several cases, very similar but distinct DCRs were identified by the two platforms.

For the α chain, three cases feature the same V and J annotation, different numbers of deletions, and either the same, or a short and extended version of the same, insert sequence (cell 24 for DCRs 12, 14, and 15; cell 47 for DCRs 12, 14 and 22; and cell 60 for DCRs 26 and 27). For the β chain, the same pattern is seen in an additional three cases (cell 25 for DCRs 21 and 23; cell 56 for DCRs 36 and 37; and cell 67 for DCRs 40 and 41).

Furthermore, five β chain cases feature the same J gene annotation, identical insert sequences, but different V gene annotations (cell 12 for DCRs 8 and 9, where the insert sequence features one additional prefixing residue; cell 15 for DCRs 11 and 12; cell 21 for DCRs 16 and 17; cell 82 for DCRs 48 and 51; and cell 138 for DCRs 56 and 59). One β chain case features the same V gene annotation, identical insert sequence, but different J gene annotations (cell 44 for DCRs 32 and 33).

These features reflect different decisions made by each platform when searching for overlaps, and to some extend may reflect issues in the mapping of TraCeR output to the Decombinator format. Furthermore, Single Tag Decombinator identifies multiple rearrangements in a single cell more frequently than TraCeR. It is likely that only one or two of these are real TCRs, and other rearrangements (that feature overly long overlaps or are found in low numbers) can be discarded. This would likely see an increase in the Jaccard index between the two platforms.

Despite some difference in annotation, the two platforms show a generally good agreement in identifying TCRs in single cell data, and this study has benefited from the complementary use of the two approaches. This is most evident in the clear discrimination between the NART negative cells (cells 1 to 95) and NART positive cells (cells 96 to 138) in Figure 2.16 and Figure 2.17. Nearly all cells labelled as NART negative were identified as expressing the same TCR α chain (ID 38) by both Single Tag Decombinator and TraCeR. Similarly, these cells were identified as expressing the same two TCR β chains (IDs 56 and 57) by the two platforms. The agreement between the approaches provides a measure of confidence that the TCR responsible for neoantigen recognition has been identified. The use of both platforms is particularly helpful for deducing that the positive cells express two β chains, rather than one chain being an artefact of the reassembly software. The two platforms identify a much more diverse set of TCR α and β chains for the NART negative cells, as one might expect. The identified TCRs for NART positive cells have been reported in (Joshi *et al.*, 2019).

2.4.3 Application in SARS-CoV-2 and Benchmarking against MiXCR

Single Tag Decombinator has also been more recently compared to the popular MiXCR TCR analysis platform (Bolotin *et al.*, 2015) in the context of SARS-CoV-2. Figure 2.18A and Figure 2.18B compare TCR α chains and TCR β chains identified by the two platforms for 48 CD8 T cells labelled as positive against the ORF3a-28 peptide in a single patient, respectively. A TCR chain is here represented by the identified V gene, J gene and CDR3 sequence. Hits in green represent TCRs identified by both MiXCR and Single Tag Decombinator. Hits in blue represent TCRs identified by MiXCR but not by Single Decombinator, while hits in orange represent TCRs identified by Single Tag Decombinator but not by MiXCR. To aid readability, the axes are labelled by an ID for each cell and for each V gene, J gene, and CDR3 sequence. Full mapping of IDs to TCRs and IDs to cells are provided

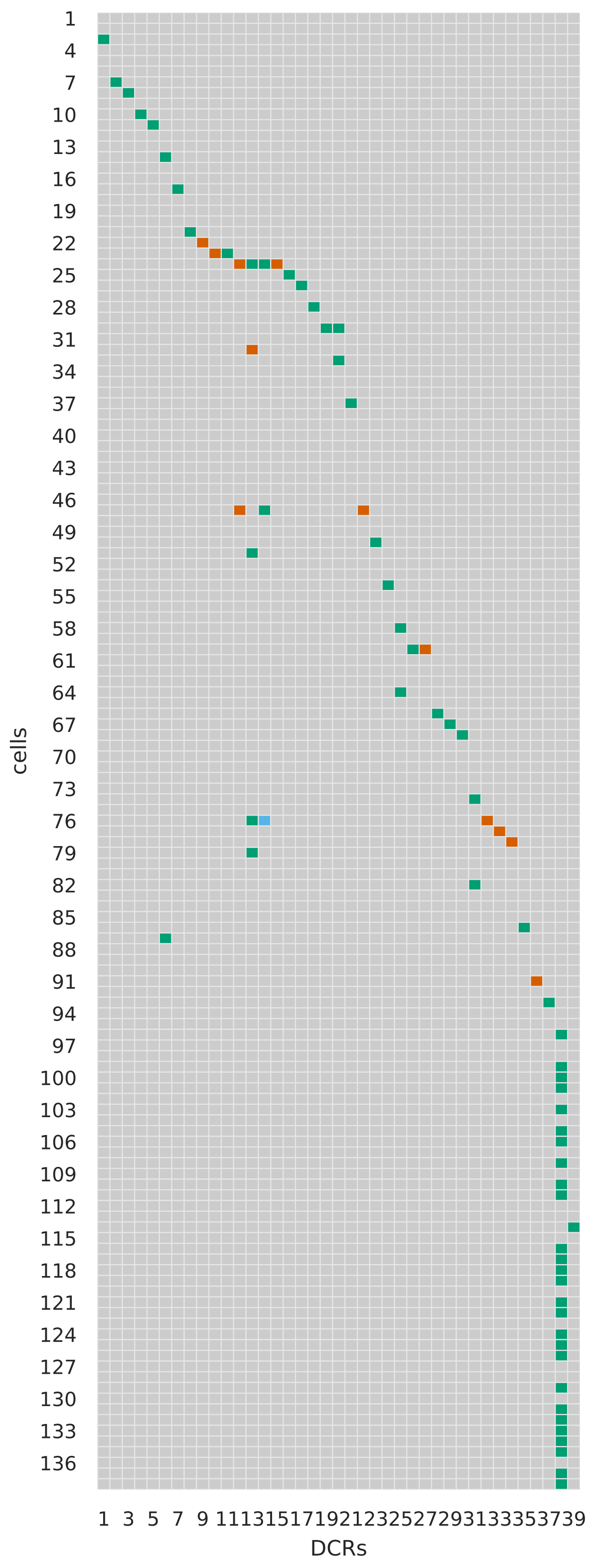


Figure 2.16: TCR α chains identified by only Single-Tag Decombinator, only by TraCeR, or by both platforms, are shown in orange, blue and green, respectively, in the NSCLC dataset.

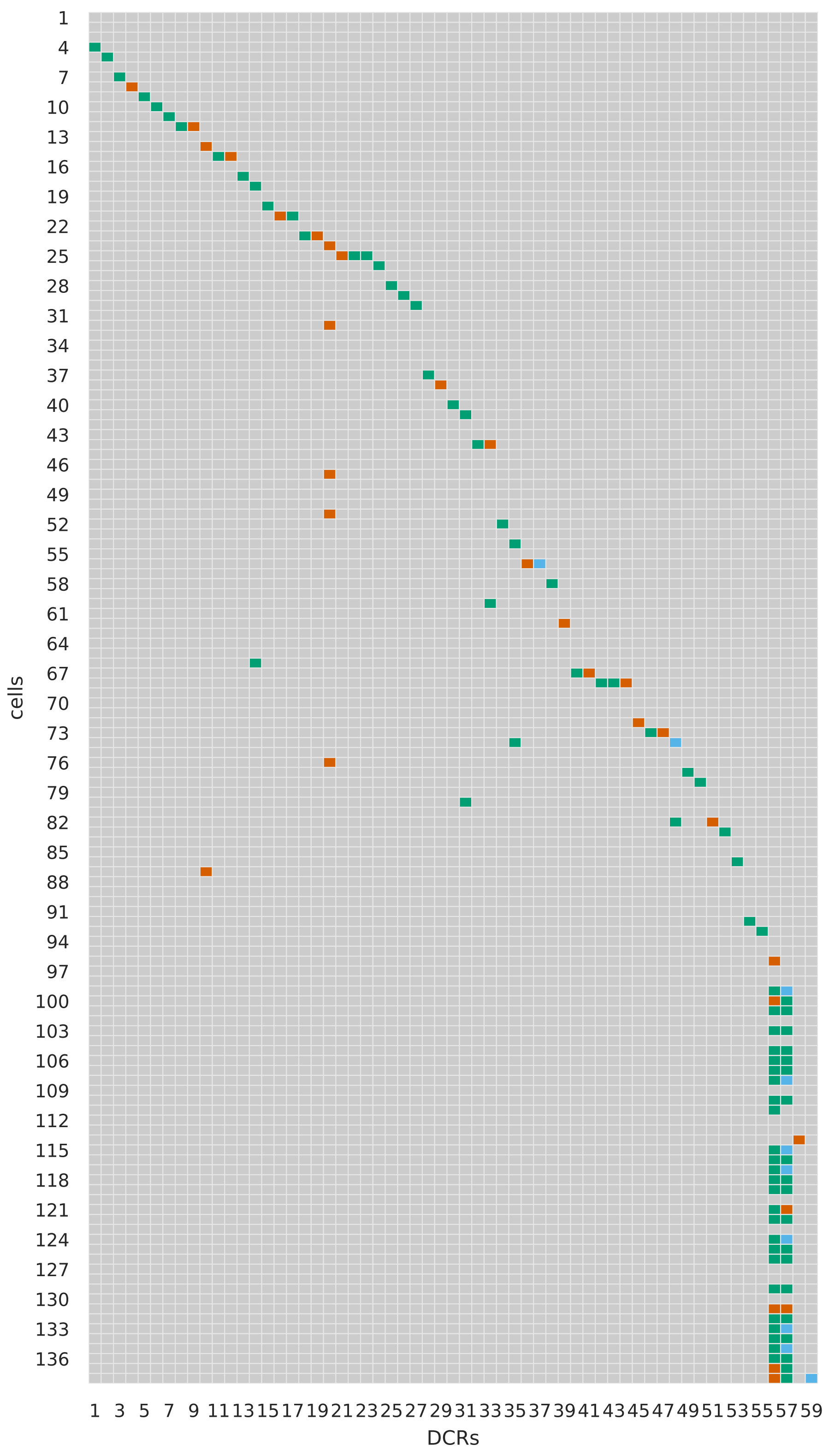


Figure 2.17: TCR β chains identified by only Single-Tag Decombinator, only by TraCeR, or by both platforms, are shown in orange, blue and green, respectively, in the NSCLC dataset.

in Table C.4, Table C.5 and Table C.6 in Appendix C. Data analysis using MiXCR was conducted by Dr Ling Felce.

In this study, rearrangements identified by Single Tag Decombinator were sorted according to how many times they were found for a single cell. A threshold was set equal to the number of rearrangements found by MiXCR for that cell or equal to 1 if MiXCR found no rearrangements. The top TCRs identified by Single Tag Decombinator within this threshold were kept for analysis. For instance, if MiXCR found 3 TCRs for a given cell, and Single Tag Decombinator found 5 TCRs, only the top 3 most frequently occurring TCRs would be kept for Single Tag Decombinator. This strategy was employed to counteract the exhaustiveness of Single Tag Decombinator in searching for overlaps, avoiding unlikely TCR rearrangements found in low numbers with very long overlap regions.

Rearrangements for the α chain and β chain were identified by at least one of the platforms in 75% and 87.5% of cells respectively. For cells where at least one α chain was identified, both platforms identified the same rearrangement in 83.3% of cases. For cells where at least one β chain was identified, both platforms identified the same rearrangement in 85.7%. The Jaccard index between the sets of TCRs identified by Single Tag Decombinator and MiXCR is 0.66 for the α chain, and 0.59 for the β chain.

In some circumstances where Single Tag Decombinator and MiXCR identify different TCRs for the same cell, the rearrangements are very similar. For example, the three β chains identified by MiXCR for cell 11 all feature the same V gene, the same J gene, and differ by one amino acid in the CDR3 sequence. Those identified by Single Tag Decombinator feature the same J gene as those identified by MiXCR, the same CDR3 sequence as one of the MiXCR rearrangements, but three differently annotated V genes. A second example can be seen for cells 3, 10, 12, 31 and 39. These cells each feature a TCR β chain identified by both platforms (with ID 2), and an additional β chain identified by MiXCR only (with ID 4). These two β chains share a J gene and CDR3 sequence, differing only in V gene (TRBV13 for ID 2, and TRBV7-3 for ID 4). The 20 bp sequence tags used by Single Tag

Decombinator are very similar for these genes, with a Levenshtein difference of 3.

It is difficult to assess whether MiXCR is correct in assigning two β chains, or Single Tag Decombinator is correct in assigning only one. With these characteristics of the data in mind, the analysis was repeated using only the CDR3 sequences annotated by both platforms. Respective plots for the α and β chain are shown in Figure C.1A and Figure C.1B in Appendix C. Separate tables are also provided mapping IDs to CDR3 sequences as Table C.7 and Table C.8, respectively. The overlap of hits predicted by the two platforms increases for this analysis, with a Jaccard index of 0.73 and 0.80 for the α and β chain, respectively.

The general agreement between the two approaches is a promising outcome of the analysis, suggesting that, for many cells, a correct TCR rearrangement has been identified. As in the previous comparison between Single Tag Decombinator and TraCeR, the complementary use of Single Tag Decombinator and MiXCR proves more informative than the use of a single platform. As this dataset comprises only CD8 positive cells from a single patient, it is limited in terms of the biological or clinical insight it can offer. Nevertheless, some interesting features of the analysis stand out.

Figure 2.18A shows the most commonly identified α chains across the cells are those with ID 1 (for cells 3, 8, 10, 12, 14, and 39) and ID 9 (for cells 18, 23, 27, 29, 30, 35, 43, and 45). 12 of these are identified by both platforms, and 2 by MiXCR only. Figure 2.18B shows that the same β chains are commonly identified for these cells. Rearrangements with β ID 2 are found for all cells with α ID 1 (cells 3, 8, 10, 12, 14, and 39), as well as for three additional cells (16, 24, and 31). For these additional cells, no α chain was identified at all by either platform. Rearrangements with β IDs 17 and 18 are both found for nearly all cells with α ID 9 (18, 23, 27, 29, 30, 43, and 45). For one cell (35) with α ID 9, no β chain was identified by either platform. All identified β chains were predicted by both platforms. The most prevalent $\alpha\beta$ chain pairings across all 48 cells are therefore those with α rearrangement 1 (TRAV8-6, TRAJ22, CAVSGPQGSARQLTF) and β rearrangement 2 (TRBV13, TRBJ1-2, CASSLIGQGGYTF), and those with

α rearrangement 9 (TRAV17, TRAJ7, CATGGNNRLAF) and dual β rearrangements 17 (TRBV16, TRBJ2-1, CASSQDLYNEQFF) and 18 (TRBV20-1, TRBJ1-2, CSVTRTHPRCYTF). These TCR clonotypes are the most prolific responders, at least in this limited dataset, to the ORF3a-28 peptide.

2.5 Future Perspectives

This chapter has described in detail the main components of the Decombinator pipeline for TCR repertoire analysis. The platform is actively maintained and updated in response to adjustments and improvements made in the complementary wet lab protocol. In this regard, this chapter has explored some of the new features that have been implemented into the computational pipeline over the course of this PhD and which have been released as Decombinator V4. An extensive set of additional scripts to aid with Decombinator analysis has been developed and improved, collected in the form of the Decombinator-Tools repository.

Additionally, an adaption of Decombinator for use with short-read fragmented (primarily single cell) data has been described. The performance of Single Tag Decombinator has been compared with the TraCeR and MiXCR repertoire analysis software in two independent studies of non-small-cell lung cancer and SARS-CoV-2. In general, Single Tag Decombinator shows a good agreement with these approaches, and its complementary use alongside the other platforms has proved biologically illuminating.

There are a number of routes that can be explored to enhance the performance of the Decombinator pipeline in the future. As Decombinator was not originally designed with single cell analysis in mind, Single Tag Decombinator has been developed outside of the main pipeline, to avoid considerable refactoring of the codebase. However, with the steady increase in the production of single cell data and the promising results of Single Tag Decombinator, an overhaul of the code to unify the projects would likely be worthwhile, expanding the ease of use and approachability of the software. Furthermore, a refactoring of the codebase into a more modular workflow would ease future software development and integration of new features.

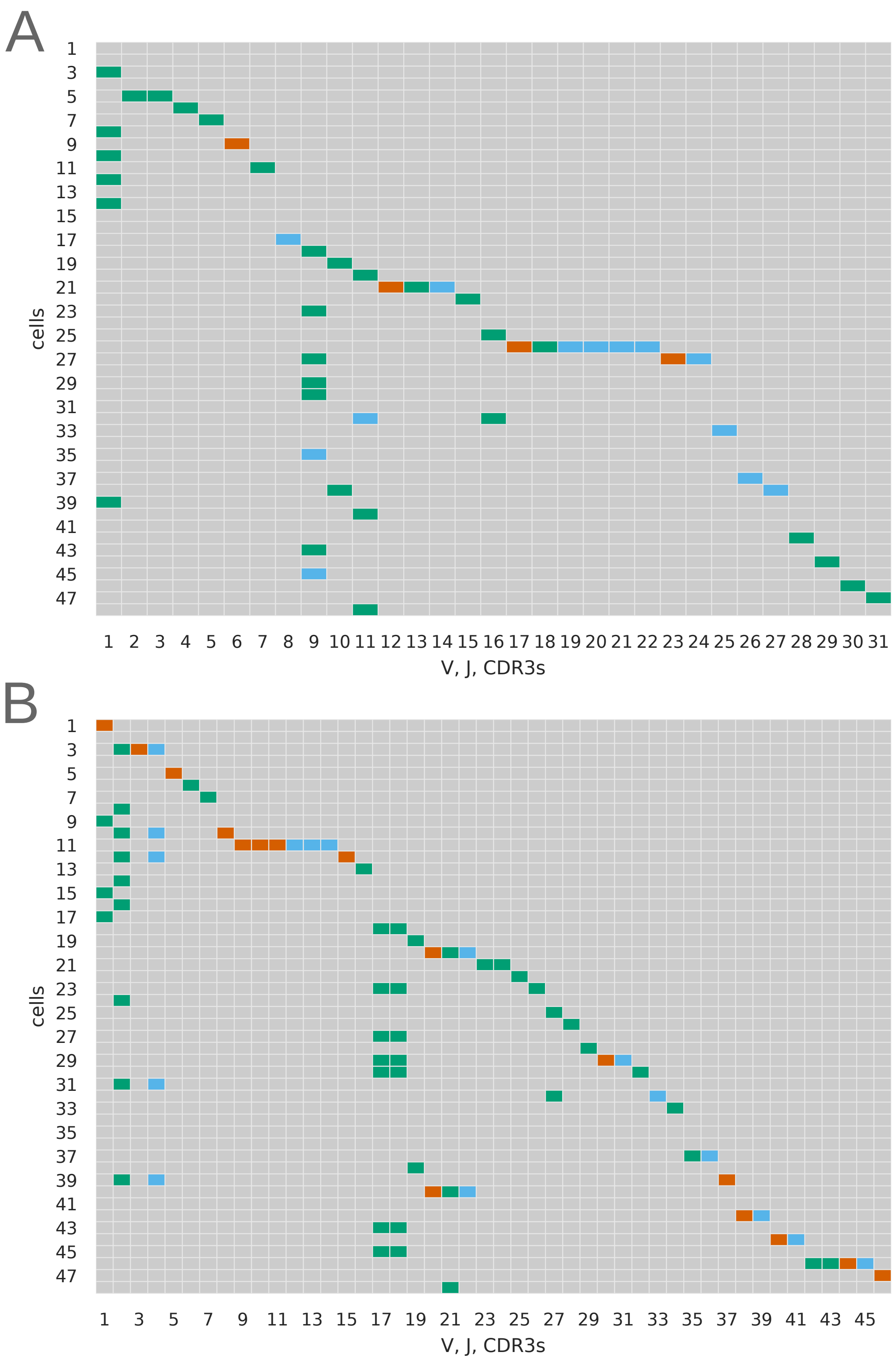


Figure 2.18: TCR (A) α chains and (B) β chains identified by only Single Tag Decombinator, only by MiXCR, or by both platforms, are shown in orange, blue and green, respectively, in the SARS-CoV-2 dataset.

Decombinator has largely been developed with an input data format consistent with the complementary Chain lab experimental protocol in mind. However, when working with external collaborators, initial input data often varies in format. A more generalised approach to input data parsing would streamline the process of working with new datasets. This could be accomplished through the use of configuration files or templates, where researchers could specify relevant details of their protocol, such as the locations of spacers, demultiplexing indices, or any other elements of the sequencing construct. Furthemore, the Decombinator pipeline has been designed to analyse FASTQ data, but could be extended to additionally parse the commonly used SAM/BAM format (Li *et al.*, 2009).

In terms of computational run time, Decombinator is one of the fastest tools currently available (Zhang *et al.*, 2020). An adaption of Decombinator parallelised to run over multiple cores was experimented with over the course of this PhD project, and offers the platform the potential to become significantly faster. Formal implementation of these features into the pipeline may prove worthwhile in the context of the increasingly large datasets being produced by HTS machines. This work would be aided by benchmarking Decombinator for speed against other TCR repertoire analysis pipelines. Furthermore, The Decombinator module currently searches input data for only α chain rearrangements or β chain rearrangements, and must be run twice to search for both. Adaptions in the Single Tag Decombinator pipeline to allow simultaneous α and β chain searching has drastically improved run time, and could be introduced in the main Decombinator workflow.

Decombinator V4 was benchmarked for accuracy against an artificial dataset based on probabilistic sequence generation and simulated PCR amplification and sequencing. A separate tool for generating test data sets based on the Decombinator α and β tags has also been described. Sampled experimental test data is also available in the Decombinator-Test-Data repository. Each of these sets serve a somewhat different purpose. The simulated data has been useful in assessing the end to end accuracy of the Decombinator pipeline. Customisable tag-constructed data is useful for testing specific functionalities within the Decombinator and Single

Tag Decombinator pipelines, such as error correction and overlap reconstruction. The Decombinator-Test-Data repository is generally used for rapid assessment that the pipeline is installed and working as intended, and is useful for onboarding new users with the software. Further work could be done to expand the Decombinator testing framework. While the input data is recorded for these datasets, the output is not. Test output data would be useful as a means to ensure that Decombinator is working as intended when implementing new features. This process could be automated by integrating the test output data with unit testing, which has proven an effective strategy in overhauling the Collapsinator fuzzy matching algorithm (using the Python `unittest` module). A wider range of unit tests would offer particular support for larger pieces of Decombinator refactoring, such as the unification of the Decombinator and Single Tag Decombinator pipelines.

Finally, a number of improvements could be made in the Single Tag pipeline. This could involve experimentation with more sophisticated overlap detection algorithms, such as the construction of de Bruijn graphs, which has been implemented in other platforms (Mose *et al.*, 2016; Chen *et al.*, 2020). The current version of Single Tag Decombinator exhaustively pairs as many V fragments and J fragments as possible. The resultant data often comprises a large number of some reconstructed sequences and very few of others. Automatic filtering of these sequences according to, for example, a minimum number of hits for a given reconstructed sequence prior to output, would likely improve performance in benchmarking tests. This threshold could be set as a hard cut-off, or, for example, as a fraction of the sequence with the largest number of reconstructed sequences. Additional automated filtering steps might include the prediction of unlikely rearrangements in the reconstructed data, such as those with over-long insert sequences. Finally, it would be illuminating to test the software in a wider range of contexts, against simulated datasets with "ground truth" information as with Decombinator V4, and in comparison to other high-performing software platforms such as CATT which reconstructs overlaps using a greedy-feasible-flow algorithm (Chen *et al.*, 2020) and TRUST4 which focuses on full-length TCR and BCR sequence reconstruction (Song *et al.*, 2021), neither of which had been published at the time this work began.

Chapter 3

Expanding the TCR Benchmark

3.1 Benchmark Assembly

To assess the accuracy of a given computational docking platform, the platform must be tested with known information. Accordingly, a docking platform may be tasked with generating a bound model of two unbound proteins of known structure, for a case where the structure of a complex of the two protein components is also already known. The modelled complex can then be compared to this bound reference complex so that an assessment can be made about the quality of the docking procedure. A successful model will closely resemble the reference structure. The regularly updated Protein-Protein Docking Benchmark (Vreven *et al.*, 2015) contains a large number of such docking cases and has been made use of extensively for this purpose.

Due to their generally conserved binding mode, no TCR-pMHC structures feature in the Protein-Protein Docking Benchmark. However, an independently curated set of 20 TCR-pMHC docking cases was assembled as the TCR Docking Benchmark for the development and testing of the TCRFlexDock software (Pierce and Weng, 2013). This benchmark has since been updated as part of the establishment of the TCR3d database (Gowthaman and Pierce, 2019), and totals 30 unique cases. Each case consists of three independently crystallised structures: (1) an unbound TCR structure, (2) an unbound pMHC structure, and (3) a bound complex structure composed of the aforementioned TCR and pMHC proteins.

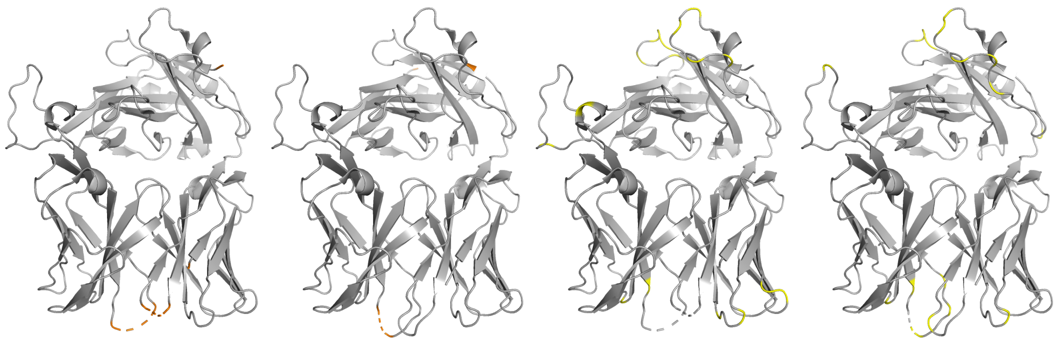


Figure 3.1: Structures of two identical TCRs crystallised together and found in PDB under code 4GRM. Despite matching a TCR-pMHC structure (4FTV) along with an unbound pMHC structure (1DUZ), these structures were not retained as a benchmark case due to regions of missing residues and missing atoms around the binding interface. Residues adjacent to missing residues are here shown in orange, and residues with missing atoms are shown in yellow.

As part of this study, an attempt was made to identify additional docking cases that might be added to the benchmark for the assessment of the TCR-pMHC modelling capabilities of docking software. Potential docking cases were identified by comparing unbound TCR and unbound pMHC structures with bound TCR-pMHC structures collected from online repositories. Unbound TCR and bound TCR-pMHC structures were sourced from the automatically curated set of structures recorded in the STCRDab database (Leem *et al.*, 2018). Unbound "pMHC-like" structures were sourced from the Protein Data Bank (PDB) (Berman *et al.*, 2000) using a simple keyword search ("MHC" and "HLA"). Structures with a resolution worse than 3.5Å were omitted. This cut off value was chosen such that the structure with PDB code 3DXA, which features in the most recent update to the TCR Benchmark (Gowthaman and Pierce, 2019) and has a resolution of 3.5Å, could be retained. All other structures presented in this chapter have resolutions of no worse than 3.25Å, which is the resolution cut off used in previous iterations of the TCR and protein-protein benchmarks (Pierce and Weng, 2013; Hwang *et al.*, 2010).

Sequence alignment was performed between the chains of the unbound TCR structures and the TCR chains of the bound TCR-pMHC structures. Similarly, sequence alignment was performed between the chains of the unbound pMHC structures and the peptide and MHC chains of the bound TCR-pMHC structures. A bound TCR-pMHC structure with a high degree of sequence similarity to both an unbound TCR structure and to an unbound pMHC structure was considered to be a “candidate” docking case. Each candidate case was then manually validated to ensure that it featured matching TCRs, peptides and MHC molecules.

Following the approach of the original TCR Benchmark (Pierce and Weng, 2013), candidate cases that featured additional proteins complexed with the identified structures were discarded. The cases 2IAM and 2IAN are notable exceptions. Both feature the staphylococcal enterotoxin superantigen bound to their pMHC structures (1KLG and 1KLU). The superantigen has been found not to interact significantly with the peptide or the TCR binding site, and these cases are therefore retained in the benchmark in this and previous assemblies (Pierce and Weng, 2013; Gowthaman and Pierce, 2019). The superantigen was removed from both structures.

Additionally, most candidates with structures that featured missing atoms or residues around the binding site were omitted from the benchmark assembly. Missing residues were identified by comparing the protein sequences provided in the header information in the raw PDB files downloaded from the Protein Data Bank and sequences constructed through the concatenation of the amino acid residues provided in the PDB file coordinates information. PDB header parsing and PDB coordinate parsing were performed using the Bio.SeqIO and Bio.PDB Biopython modules, respectively. Missing atoms were identified by comparing the atoms provided for each residue in the PDB coordinate information to an expected set of atoms for each amino acid. Locations of missing residues and atoms were visualised using PyMOL to judge whether they were far enough away from the binding interface to be included in the benchmark, or minimal enough in missing information to be repaired before docking. Some examples of highlighted missing residue and atom locations are shown in Figure 3.1. The candidate case displayed, 4FTV, was ultimately rejected for featuring too many missing residues and atoms in binding interface regions. Three cases were retained despite missing features around the binding interface — 2NX5, 2OI9 and 2PXY. These cases have been used previously in the benchmarking of the TCRFlexDock platform (Pierce and Weng, 2013).

Methods for repairing the structures prior to docking is detailed in Section 3.2.2.

In total, a resulting 14 additional TCR-pMHC docking cases were identified for use in the docking platform comparison. The original PDB codes for the structures of each of the 44 docking cases is provided in Table 3.1. Throughout this study, an individual docking case is referred to by the PDB code of its original bound TCR-pMHC complex.

3.2 Preprocessing Structures

Each of the benchmark structures was passed through a number of preprocessing (or "cleaning") steps before submission to the docking algorithms and future analysis. These steps are detailed below.

3.2.1 Isolation of a Single Biological Assembly

PDB structural data contains the unit cell, which contains one or more copies of a macromolecular assembly. These may represent the biologically active molecule or a part of the biological assembly (or biounit) that may be reconstructed by applying symmetry operations. Differences in experimental conditions and in local packing may result in these assembly copies having identical or slightly different conformations. The PDB file with code 3SKN from benchmark case 3SJV features four copies of the same TCR biological assembly, and is illustrated as an example in Figure 3.2. For an $\alpha\beta$ TCR, the biological assembly is composed of an α chain and a β chain. The chain IDs for each α and β chain pair are A and B, C and D, E and F, and G and H, and are coloured orange, blue, green and rose, respectively.

Generally, only one of each of the unbound components should be used when modelling a bound protein-protein complex using computational docking software. In preprocessing each TCR benchmark case, multiple biological asssemblies present in a single PDB file were automatically isolated using a bespoke pairing script that associates each α chain with the appropriate β chain. The RMSD (defined in Appendix A) between each isolated TCR and each isolated TCR-pMHC structure was computed, and the pair with the lowest value was retained to form the benchmark case. The RMSD between the chosen TCR-pMHC structure and

Bound Complex	Unbound TCR	Unbound pMHC	MHC Class	IRMSD	$F_{\text{non-nat}}$	Difficulty
1AO7 *	3QH3	1DUZ	I	1.25	0.33	rigid
1MI5 *	1KGC	1M05	I	1.25	0.48	medium
1MWA *	1TCR	1LEK	I	1.14	0.3	rigid
1OGA *	2VLM	2VLL	I	1.36	0.43	medium
2BNR *	2BNU	1S9W	I	0.72	0.23	rigid
2CKB * ‡	1TCR	1LEG	I	1.17	0.45	medium
2IAM *	2IAL	1KLG	II	0.87	0.24	rigid
2IAN *	2IAL	1KLU	II	0.82	0.3	rigid
2NX5 * †	2NW2	1ZSD	I	1.19	0.37	rigid
2OI9 *	1TCR	3ERY	I	1.1	0.41	medium
2PXY *	2Z35	1K2D	II	1.18	0.55	medium
2PYE *	2PYF	1S9W	I	0.88	0.3	rigid
3DXA * †	3DX9	3DX8	I	1.48	0.39	rigid
3H9S *	3QH3	3H7B	I	1.31	0.42	medium
3KPR *	1KGC	3KPQ	I	1.37	0.55	medium
3KPS *	1KGC	3KPP	I	1.31	0.48	medium
3PWP *	3QH3	3PWL	I	1.24	0.36	rigid
3QDG * † ‡	3QEU	1JF1	I	0.91	0.31	rigid
3QDJ * ‡	3QEU	2GUO	I	0.94	0.28	rigid
3SJV * ‡	3SKN	1M05	I	0.96	0.41	medium
3UTT *	3UTP	3UTQ	I	0.75	0.4	rigid
3VXR	3VXQ	3VXN	I	0.82	0.38	rigid
3VXS * †	3VXQ	3VXP	I	0.89	0.35	rigid
3W0W * † ‡	3VXT	3VXO	I	0.94	0.42	medium
4JFD * †	4JFH	4JFP	I	1.51	0.51	medium
4JFF	4JFH	1JF1	I	1.54	0.52	medium
5C07	3UTP	5C0E	I	0.57	0.15	rigid
5C08	3UTP	5C0F	I	0.65	0.43	medium
5C09	3UTP	5C0G	I	0.59	0.24	rigid
5C0A	3UTP	5N1Y	I	0.5	0.3	rigid
5C0B	3UTP	5C0I	I	0.59	0.25	rigid
5C0C	3UTP	5C0J	I	0.64	0.35	rigid
5HHM	2VLM	5HHN	I	1.42	0.51	medium
5HYJ	3UTP	5C0D	I	0.55	0.34	rigid
5IVX	5IW1	3ECB	I	1.29	0.38	rigid
5NME	5NMD	2V2W	I	1.07	0.34	rigid
5NMF * †	5NMD	5NMH	I	1.05	0.38	rigid
5NMG * † ‡	5NMD	5NMK	I	1.07	0.42	medium
6AMU	3QEU	6AMT	I	1.16	0.41	medium
6AVF * †	6AT6	6AT5	I	1.95	0.72	medium
6CQL * †	6CPH	6CPN	II	0.78	0.23	rigid
6CQQ * †	6CPH	6CPO	II	0.83	0.21	rigid
6CQR * †	6CPH	6CQJ	II	0.85	0.26	rigid
6EQB	4JFH	2GUO	I	1.62	0.55	medium

Table 3.1: PDB codes for each unbound TCR and pMHC, and bound TCR-pMHC structure, for each docking benchmark case. The interface root-mean-square deviation (IRMSD), the fraction of non-native contacts ($F_{\text{non-nat}}$), and the docking difficulty are provided for each case, and are defined in Section 3.3.
* TCR docking cases that feature in the TCR3d database;
* Cases that differ in IRSMD score to those in the TCR3d database;
‡ Cases that differ in docking difficulty class in the TCR3d database.

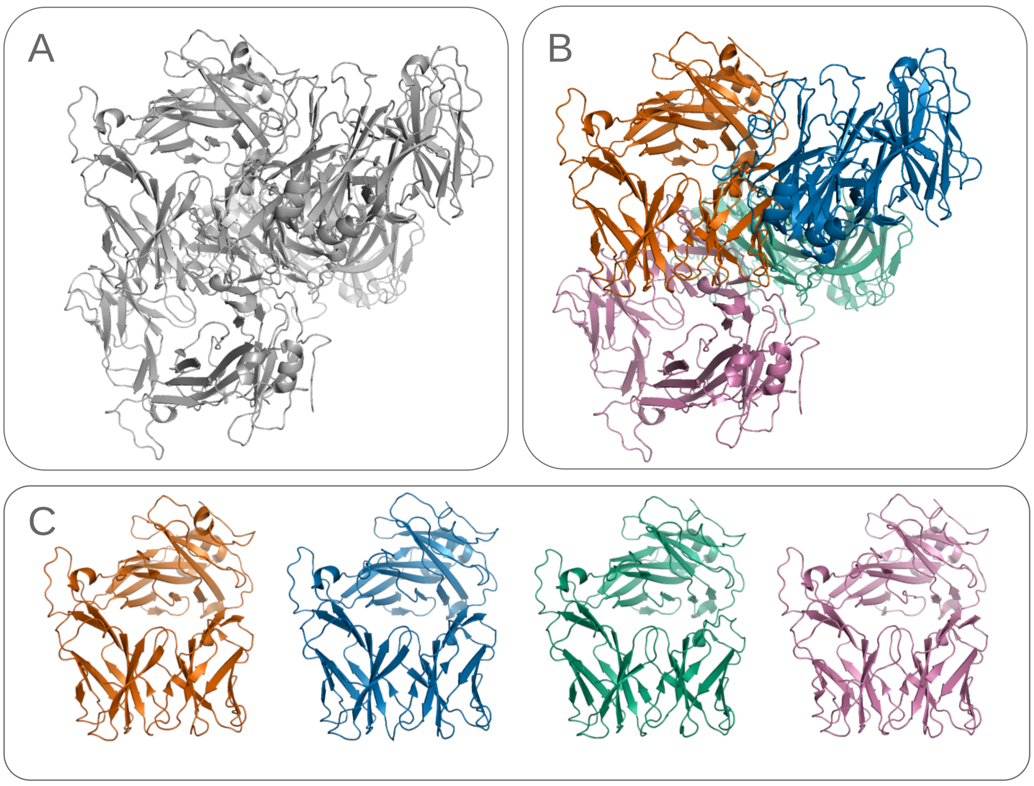


Figure 3.2: Multiple copies of the same TCR found in the PDB file with code 3SKN from benchmark case 3SJV. (A) It is unclear which protein chains make up each TCR in the raw PDB data. Each α chain was paired with a β chain such that each TCR was isolated and distinguishable. Paired α and β TCR chains, with chain IDs `A` and `B`, `C` and `D`, `E` and `F`, and `G` and `H`, are shown coloured orange, blue, green and rose, respectively, according to (B) their original coordinates, and (C) separately.

each pMHC structure was also computed, and the pMHC structure with the lowest RMSD value was retained to complete the benchmark case.

It is worth noting that this decision may lower the conformational change present between the bound and unbound components, resulting in an easier modelling task for docking software. For benchmark cases that feature both in this and previous benchmarks, differences in the interface root-mean-square deviation (IRMSD; defined in Section 3.3) and difficulty class can likely be attributed to alternate strategies for choosing which TCR, TCR-pMHC or pMHC structure to retain for each case, and is discussed further in Section 3.3. The extent of conformational change between unbound and bound structures has been reported previously

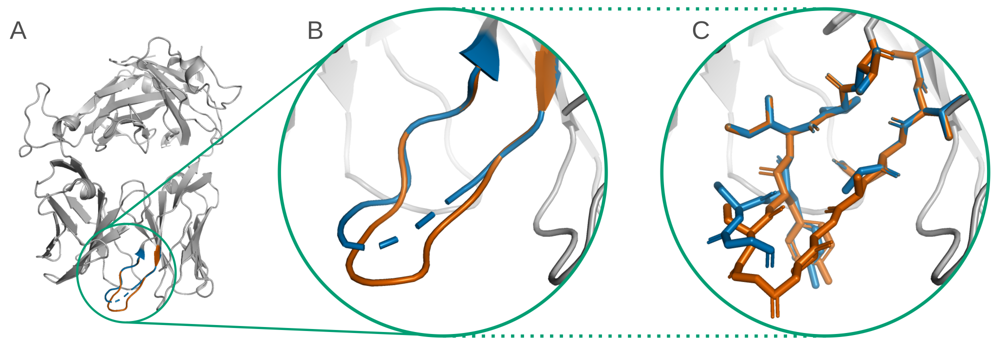


Figure 3.3: Repair of missing residue and atoms in the TCR of case 2NX5 (2NW2). The TCR is shown (A) with repaired loop (orange) superimposed on top of the original broken loop (blue). Close up images of the loop region are shown in cartoon (B) and stick (C) representation.

as higher on average for TCR-pMHC docking test cases than for antibody-antigen test cases (Pierce and Weng, 2013). In this study, IRMSDs between unbound TCR and bound TCR-pMHC structures were found to range from 0.72Å to 1.95Å.

In summary, this preprocessing step assures that each benchmark case comprises only a single TCR structure, single pMHC structure, and single TCR-pMHC structure.

3.2.2 Repair of Structures with Missing Features

A number of structures retained as part of the set of benchmark cases feature missing residues or atoms around the binding interface. The TCR structure of case 2NX5 (2NW2) is missing one CDR3 loop residue and 5 further atoms in two adjacent residues; the pMHC structure of case 2NX5 (1ZSD) is missing 3 side chain atoms in one of its peptide residues; the pMHC structure of case 2OI9 (3ERY) is missing 6 side chain atoms in one peptide residue; and the pMHC structure of case 2PXY (1K2D) is missing 6 side chain atoms in one peptide residue.

The missing residues and atoms were added to these structures using the MODELLER software (Webb and Sali, 2016), version 9.25. The protocol to repair these structures can be found in the Modeller-Repair repository, at `https://github.com/innate2adaptive/Modeller-Repair/`. Input data for the TCR structure of case 2NX5 is provided as an example, alongside instructions for installing and running the protocol. The protocol is described briefly below.

A structure in need of repair should have its amino acid sequence saved with gaps replacing missing residues (or residues with missing atoms) alongside the desired sequence in PIR format (as in `example/2nx5_r_u.ali`). A repaired structure can then be generated by simply supplying the `repair.py` script with the alignment file and with the PDB file of the template structure (the structure to be repaired). Ten models of a replacement protein chain are automatically produced using the MODELLER `automodel` class (via `scripts/model_chain.py`), and the model with the best MODELLER DOPE score is retained. The template structure then has its broken chain replaced with the retained modelled chain (via `scripts/replace_chain.py`) to produce the repaired output structure. Users might optionally make use of the "select range" feature, whereby flexibility in the model refinement is limited to only the specified range of residues, through an additional argument supplied to the `repair.py` script, which should produce structures more closely aligned with the template.

Figure 3.3A shows the TCR of case 2NX5 with the loop repaired by MODELLER (orange) superimposed upon the original loop with missing atoms (blue). Close up images of the loop region are shown in cartoon representation in Figure 3.3B, and with residue side chains visible in stick representation in Figure 3.3C. Existing atoms in the template are close to their original position, though some have been adjusted slightly by MODELLER to make room for the added residues.

3.2.3 Alternate Locations, Solvent and Small Molecules

Most protein structures resolved using x-ray crystallography in the Protein Data Bank contain additional solvent atoms that were present during the experimental crystallisation. Other small molecules used to aid the crystallisation process are also sometimes present in the raw data. As these are generally only present for crystallisation purposes, they were removed for each structure prior to docking, as shown in Figure 3.4A.

Occasionally, molecular flexibility can result in structural data that features multiple spatial coordinates, or alternate locations, recorded for a single atom. Atoms with multiple recorded coordinates cause problems for certain docking plat-

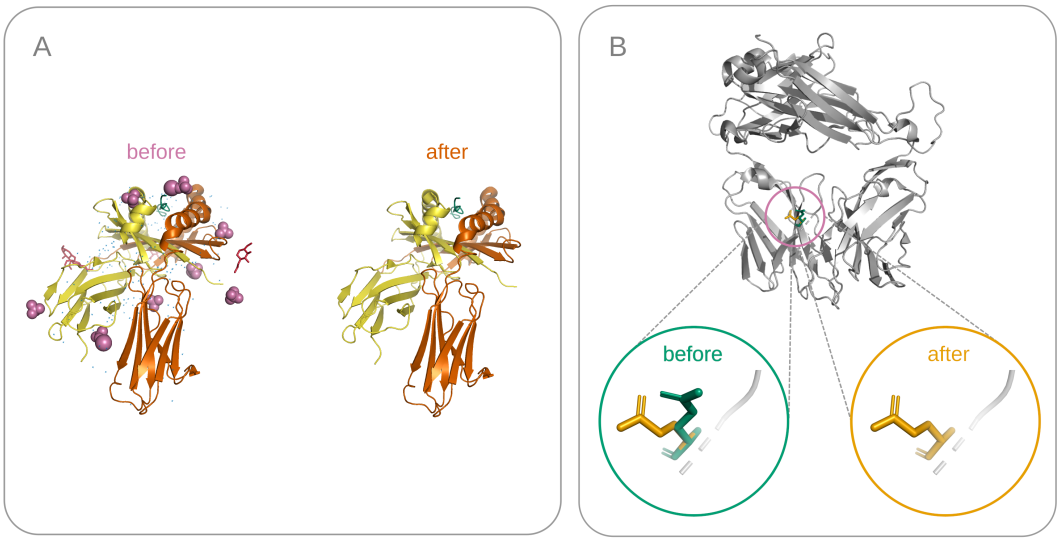


Figure 3.4: (A) The pMHC structure of benchmark case 6CQL (6CPN) is shown before and after the removal of solvent and other small molecules. MHC chains are shown in yellow and orange, the antigenic peptide in green, solvent molecules in blue, and additional small molecules in red and rose. (B) The alternate locations of atoms in a residue of the TCR of benchmark case 3H9S (3QH3) are coloured in green and yellow and shown in stick representation. Insets show close ups of the residue before and after one of the sets of atoms has been removed.

forms, molecular dynamics simulations, and RMSD calculations. Atoms with alternate locations in TCR, pMHC and TCR-pMHC structures typically feature only two sets of coordinates, and are usually labelled as `A` and `B`. For this benchmark dataset, alternate locations were automatically identified using the Bio.PDB Python module, and resolved by simply removing one of the coordinate sets. In this study, those labelled with alternate position `A` were retained, and the others were discarded. This process is illustrated in Figure 3.4B.

An example of the alternate locations for a Cα atom in a cysteine residue in the raw PDB data for the pMHC structure for benchmark case 5HHM is shown below, with `A` and `B` labels highlighted in orange:

```
ATOM   2955  CA  ACYS B 80      -4.338   0.663  51.500  0.50 22.62           C
ATOM   2956  CA  BCYS B 80      -4.288   0.696  51.390  0.50 23.02           C
```

Loop	Residue Numbers
CDR1	27 - 38
CDR2	56 - 65
CDR3	105 - 117

Table 3.2: Standardised residue number ranges for the CDR loops according to the IMGT numbering scheme.

3.2.4 Standardisation of Chain Labels

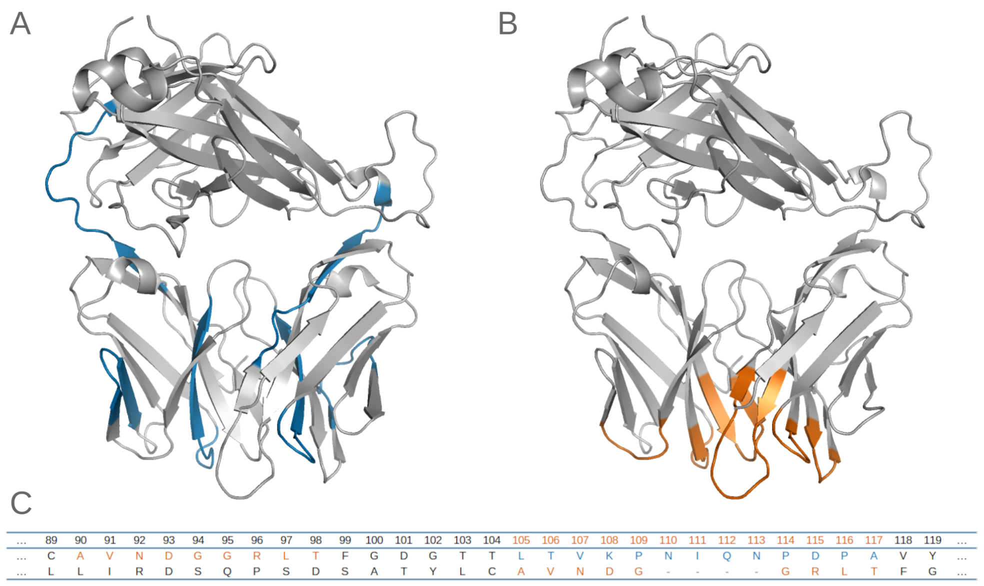
In order to submit each benchmark case to a given docking platform automatically (or at least in reasonable time), the TCR and pMHC chains were relabelled to a standardised convention employed by the original docking benchmark (Pierce and Weng, 2013). TCR chains were given chain IDs `D` and `E`; MHC molecules were given chain IDs `A` and `B`; and peptides were given chain ID `C`.

3.2.5 Standardisation of TCR Residue Numbering

Raw TCR and TCR-pMHC PDB files do not generally specify their residues according to a standardised convention. The TCR residues in these structures were therefore renumbered according to the IMGT (international ImMunoGeneTics information system) numbering scheme. This ensures that the CDR loops, which vary in length, are always specified using the same numbering IDs, making them easy to locate in analysis scripts or to submit to docking platforms as binding residues. Table 3.2 shows the numbering IDs of each CDR loop according to the IMGT scheme. Figure 3.5 shows a TCR structure with these residues highlighted before and after IMGT renumbering. Renumbering was performed using the ANARCI (Dunbar and Deane, 2016) Python module.

3.2.6 Spatial Initialisation

In order to avoid any potential initial orientation bias, all TCR and pMHC structure were randomly translated and rotated before docking. Structures were permitted to translate by up to 20Å along each axis from their initial coordinates, and freely rotate around one or more axes, during preprocessing.



...	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	...
...	C	A	V	N	D	G	G	R	L	T	F	G	D	G	T	T	L	T	V	K	P	N	I	Q	N	P	D	P	A	V	Y	...
...	L	L	I	R	D	S	Q	P	S	D	S	A	T	Y	L	C	A	V	N	D	G	-	-	-	-	G	R	L	T	F	G	...

Figure 3.5: Residues of the TCR of benchmark case 6EQB (4JFH) with IDs matching the numbering of the CDR3 loops according to the IMGT numbering scheme (105-117) are highlighted before (A) and after (B) renumbering in blue and orange, respectively. (C) Sections of the α chain sequence are shown with the CDR3 loop residues highlighted in orange with incorrect numbering (middle) and correct numbering (bottom) before and after renumbering, respectively. Residues incorrectly labelled with the CDR3 IDs are highlighted in blue, matching those in (A).

3.3 Benchmark Difficulty Evaluation Criteria

In an effort to establish the performance and improvement of docking algorithms on sets of benchmark docking cases, the CAPRI community has established metrics that can be used to capture the expected difficulty of modelling any particular protein-protein complex. The protein-protein benchmark provides docking cases categorised (in order of increasing difficulty) as of either "rigid", "medium" or "difficult" difficulty (Hwang *et al.*, 2010). The same approach was taken in the assembly of TCR-pMHC cases for the TCR docking benchmark (Pierce and Weng, 2013). The criteria describing each of these categories is shown in Table 3.3, and the associated metrics — the interface root-mean-square deviation (IRMSD); and the fraction of non-native contacts ($F_{\text{non-nat}}$) — are described below.

The IRMSD is calculated by superimposing the unbound partner structures (here, the unbound TCR and unbound pMHC structures) onto the bound reference

Difficulty	IRMSD (A)	$F_{\text{non-nat}}$
Rigid	≤ 1.5	≤ 0.4
Medium	> 1.5 and ≤ 2.2	-
	or	
	≤ 1.5	> 0.4
Difficult	> 2.2	-

Table 3.3: CAPRI criteria for determining docking case difficulty.

structure (here, the bound TCR-pMHC structure), and then computing the RMSD of the interface residue backbone atoms of the reference structure and those of the superimposed unbound partners. In this context, an interface residue is defined as a residue containing at least one atom within 10Å of an atom in the binding partner (Méndez *et al.*, 2003).

The $F_{\text{non-nat}}$ is here calculated by superimposing the unbound partner structures onto the bound reference structure (as above), and then computing the number of non-native contacts between the unbound partners divided by the total number of contacts between these partners. Non-native (incorrect) contacts are defined as residue pairs that are in contact in the superimposed unbound structures, but are not in contact in the reference structure (Méndez *et al.*, 2003). Contacting residues are here defined as pairs of residues with a distance of at most 5Å between their closest atoms. Mathematical descriptions and illustrations for RMSD, IRMSD and the $F_{\text{non-nat}}$ are provided in Appendix A.

The IRMSD, $F_{\text{non-nat}}$, and docking difficulty is provided alongside each docking case in Table 3.1. It should be noted that in some cases, these measurements differ from those reported in the original TCR Benchmark (Pierce and Weng, 2013). These discrepancies can largely be attributed to differences in which protein chains were selected for cases where multiple TCR, pMHC or TCR-pMHC structures featured in the raw PDB data. In this study, bound and unbound components were chosen such that the RMSD between the two was minimised, as described in Section 3.2.1. Differences to the original benchmark are highlighted in Table 3.1. In general, the difference in IRMSD is minimal compared to the original benchmark.

Benchmark Feature	Number
Contains MHC Class I	38
Contains MHC Class II	6
Contains human TCR	39
Contains mouse TCR	5
Max TCR sharing among cases	8
Max pMHC sharing among cases	2
Total unique TCRs	20
Total unique pMHCs	40
Total cases	44

Table 3.4: A summary of some biological features of the TCR Benchmark cases.

The original benchmark does not provide $F_{\text{non-nat}}$ values, so no direct comparison could be made. Five cases in this study have been reclassified from "rigid" to "medium" difficulty, and one case from "medium" to "rigid" difficulty. The 44 docking cases are all of "rigid" or "medium" difficulty, with no "difficult" docking cases identified.

3.4 Features of the Benchmark

The benchmark cases used in this study feature both TCR and pMHC structures from human and from mouse. The majority of MHC structures in the benchmark are of Class I, though a small number of MHC Class II structures are also represented. A number of cases feature the same TCR but bound to different pMHC molecules, or different TCRs bound to the same pMHC molecule, demonstrating the nature of TCR cross-reactivity. The most prevalent TCR structure (3UTP) features in 8 separate cases. Among the 44 cases, the benchmark contains 20 unique TCR structures and 40 unique pMHC structures. These biological features of the benchmark cases are summarised in Table 3.4.

3.5 Data Availability

All docking benchmark cases used in this study are available at `https://github.com/innate2adaptive/ExpandedBenchmark`. Original data from the PDB can be found (renamed) in the `raw` directory, and preprocessed according to the above specifications in the `imgt` directory.

3.6 Tools

Repair of missing atoms and chains was performed using the MODELLER software (Webb and Sali, 2016), version 9.25, directed with a Python protocol made available at `https://github.com/innate2adaptive/Modeller-Repair/`. Renumbering of TCRs according to the IMGT numbering scheme was performed using the ANARCI (Dunbar and Deane, 2016) Python module. Isolation of single biological assemblies, removal of solvent, small molecule and alternate locations, and chain relabelling, were performed using the Bio.PDB (Hamelryck and Manderick, 2003) Biopython (Cock *et al.*, 2009) module. Structures were randomly translated and rotated using the PyMOL (`http://www.pymol.org`) api.

Chapter 4

Information-Driven Docking for TCR-pMHC Complex Prediction

4.1 Introduction to Study

Understanding the nature of the TCR-pMHC recognition process on a molecular level is essential for TCR based therapeutics and vaccine design. The breadth of the TCR repertoire and the cross-reactive nature of TCRs are obstacles to this understanding, as crystallising even a small sample of proteins remains an expensive and difficult task. Accurate computational modelling of protein structures and structural complexes offers an efficient alternative to traditional crystallography. The study detailed in this chapter explores the ability of four general-purpose computational docking platforms (ClusPro, LightDock, ZDOCK and HADDOCK) to accurately model bound TCR-pMHC complexes and serves as a more detailed description of work recently published in Peacock and Chain (2021).

The ClusPro, HADDOCK, LightDock and ZDOCK docking platforms were chosen for this work as each allows additional information to be supplied alongside unbound structures to guide their protein complex modelling algorithms. These features include the ability to specify particular residues in the unbound components that are expected to form a part of the binding interface in the bound complex. Each of the docking suites make use of this binding site information differently (outlined in Section 4.3), but have the shared goal of restricting their set of output models around the specified interface. This approach is particularly suitable for proteins that form complexes around well-characterised interfaces, such as TCRs and antibodies. Nearly all TCR residues that make contact with pMHC are found in the CDR loop regions. All pMHC residues that make contact with the TCR are found in the peptide and in nearby MHC surface residues in the plane of the peptide groove (Rudolph *et al.*, 2006). Analogously, antibodies form binding interfaces with antigen around their CDR hypervariable (HV) loops.

Ambrosetti *et al.* (2020a) have explored the ability of ClusPro, HADDOCK, LightDock and ZDOCK to accurately model the antibody-antigen interface using varied descriptions of the binding residues to assess modelling performance in the context of incomplete (and possibly unreliable) additional information. These descriptions mimic different levels of information that might be obtained about the antibody and antigen before binding. The first description ("HV-Surf") contains only a list of residues known to be in the antibody HV loops. In these circumstances, the docking suites have vague information about which part of the antibody makes contact with the antigen, but no knowledge of the location on the antigen at which the contact takes place. The second description ("HV-Epi 9") is more detailed, providing the residues in the HV loops as well as any residues in the antigen that are within 9Å of the antibody in the docking case reference structure. The third description ("Real interface") is the most detailed, and provides only those residues in both the antibody and the antigen that are within 4.5Å of their binding partner. This description is intended to represent the ideal case, where all information about the binding interface is known. The docking suite comparison was performed using the 16 new antibody-antigen docking cases in the protein-protein docking benchmark (Vreven *et al.*, 2015).

The basis of this TCR-pMHC modelling study is founded on similar principles. Four scenarios were constructed containing varied descriptions of the TCR-pMHC interface to assess modelling performance in the context of vague or detailed knowledge of the binding. These scenarios are outlined in the following section. The expanded TCR benchmark dataset described in Chapter 3 was used to test the soft-

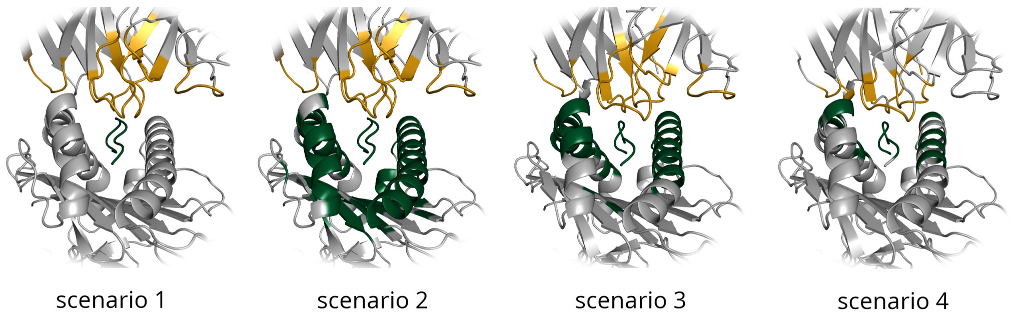


Figure 4.1: TCR and pMHC residues that constitute four varied descriptions of the TCR-pMHC binding interface are highlighted in yellow and green respectively.

ware, and the performance of each platform for each of the four scenarios for these structures is presented in Section 4.5.

4.2 Varied Descriptions of the TCR-pMHC Interface

Various levels of information about each TCR-pMHC binding interface are represented in the four scenarios detailed below. For each docking case, residues that met the requirements of the scenario were provided to the four docking platforms as binding site information to guide the docking. Each docking platform requires the binding residue specification to be formatted differently. Example inputs for docking case 1AO7 for Scenario 4 and submission instructions are provided in Appendix E. The four scenarios are illustrated in Figure 4.1 for TCR recognition of a peptide bound by MHC Class I, with selected TCR residues highlighted in yellow and selected pMHC residues highlighted in green. Residue identification was performed using the Bio.PDB (Hamelryck and Manderick, 2003) Biopython (Cock *et al.*, 2009) module.

4.2.1 Scenario 1

Scenario 1 is the simplest docking scenario in this study, and provides the most vague information about the binding interface. Here, only the TCR CDR loop residues and the residues of the presented peptide are provided as binding residues. No information is provided about likely MHC binding residues. This information is readily available from the unbound TCR and pMHC structure with no additional analysis required, provided that they have already undergone chain relabelling and

IMGT renumbering steps, described in Section 3.2.4 and Section 3.2.5, respectively. For the unbound TCR structure, it is enough to simply supply all residues for chains D and E that have residue IDs that sit in the range of the CDR loops (provided in Table 3.2). For the unbound pMHC structure, all residues of chain C (the relabelled peptide) should be supplied.

4.2.2 Scenario 2

In addition to the CDR loop and peptide residues of Scenario 1, Scenario 2 also provides a vague description of MHC residues likely to be involved in the binding interface. MHC residues were selected as any residues in pMHC structure with at least one atom within a distance of 9Å from any atom in the peptide. For MHC Class I structures, these residues almost always sit exclusively in the HLA chain (relabelled as chain A in the TCR benchmark), which contains the whole of the peptide groove. For MHC Class II structures, these residues sit in the two HLA chains (relabelled as chain A and B) that together form the peptide groove.

4.2.3 Scenario 3

Scenario 3 is similar to Scenario 2, but is intended to provide a more accurate representation of peptide and MHC residues involved in the binding interface. In this scenario, MHC residues are retrieved from the bound reference structure, rather than the unbound pMHC structure. Any residues in the pMHC component with at least one atom within a distance of 9Å of an atom in the TCR binding partner were selected as binding residues. This information is typically not available prior to docking, and is possible only for the constructed benchmark cases where a reference structure exists. As with Scenarios 1 and 2, residues in the CDR loops were specified as binding residues for the TCR component.

4.2.4 Scenario 4

Scenario 4 provides "real interface" information, where all binding residues are selected from the bound reference structure. All TCR residues and pMHC residues with at least one atom within a distance of 4.5Å of an atom in their corresponding binding partner were selected as binding residues. This scenario supplies the docking platforms with the most accurate available information about the binding interface. It is expected that the docking platforms should show the best performance for this scenario. As with Scenario 3, the information in Scenario 4 relies on the reference structures of the benchmark cases, which are not generally available when performing a docking experiment.

4.3 Information-Driven Docking Platforms

An overview of the four platforms used in this study and how they make use of additional information to improve docking accuracy is provided below. Details of the settings for each approach and rough run time estimates are included, as well as a subjective review of their accessibility (which, nevertheless, hopefully provides some insight to the interested reader).

4.3.1 ClusPro

Models from ClusPro were generated using the ClusPro online web server (`https://cluspro.org`) (Kozakov *et al.*, 2017) with the default settings. The ClusPro platform makes use of the FFT rigid-body approach (see Section 1.5.4) for rapid protein-protein docking. By default, the web server provides results using four different scoring functions — `balanced`, `electrostatic-favored`, `hydrophobic-favored`, and `van der Waals + electrostatics`. The 1000 lowest-energy structures sampled by ClusPro are clustered by RMSD into up to 30 clusters. Retained structures undergo fixed-backbone energy minimisation using only the van der Waals term of the CHARMM potential (Brooks *et al.*, 1983) to remove steric clashes. Conformation change in output models has been reported as very minimal (Kozakov *et al.*, 2017).

The ClusPro web server provides a variety of advanced options to tailor the docking search, including the option of providing either known attractive or repulsive residues in initial unbound components. In this study, binding residues were submitted as attractive residues. An additional attractive force is applied to the specified residues in the scoring function.

The ClusPro web server has a simple, minimal interface and is very easy to use.

Submission of unbound proteins, binding residues, and downloading results should be straightforward for researchers with minimal computational experience. Despite heavy usage and sometimes long queue times, a ClusPro docking job generally finishes within 4 hours (Kozakov *et al.*, 2017). A submission allowance of 15 jobs is permitted at any given time, and are easy to monitor if using the server with a free account.

ClusPro results can be downloaded from a url based on the ID of the ClusPro job as a compressed archive. Security restrictions require logging in to a ClusPro account to download output data, and therefore these archives must be downloaded manually and individually. A Python script was written to automatically unpack and organise the downloaded data.

4.3.2 ZDOCK

Models from ZDOCK were generated using the ZDOCK online web server (`http://zdock.umassmed.edu/`) version 3.0.2 (Pierce *et al.*, 2014b) with the default settings. Like ClusPro, ZDOCK makes use of the FFT rigid-body approach for highly efficient docking. The top 1000 models scored by ZDOCK are made available for download.

The ZDOCK web server allows users to provide information about the binding interface in two ways. The first method allows users to choose specific residues to block from being included in the binding interface. Atoms in these residues are assigned a highly unfavourable contact energy by the scoring function that makes them much less likely to feature as binding residues in output models. The second alternative method allows users to choose specific contacting residues to include in the binding interface. Rather than assigning atoms in these residues with highly favourable contact energies, output models are instead filtered to remove results that do not feature these residues as being part of the binding interface. For this project, residues found not to be involved in the binding for each scenario were supplied to the blocklist, allowing ZDOCK to tailor the docking algorithm to the additional information, rather than using the information as a post-modelling process. This approach is consistent with that of Ambrosetti *et al.* (2020a) in the context of antibody-antigen docking, and allows for comparison between the studies.

Like the ClusPro web server, ZDOCK also features a simple and easy to use interface. Users may upload their own PDB files or use structures directly from the PDB by supplying the relevant PDB code (although an error produced by a Python script appears to have broken this feature at the time of writing). The residue selection screen includes visualisation of uploaded PDB files for web browsers with java applet support. Blocked or binding residues must be selected manually from lists of every residue in the two proteins (rather than inputting as free text to a text box, as with ClusPro). This is very inconvenient for large scale docking studies, and consequently custom JavaScript commands were written to automatically select the appropriate residues. The commands can be copied and pasted into the console of web browser development tools. This clumsy workaround proved more efficient at this scale than resolving difficulties in installing antiquated libraries relied upon by the ZDOCK command line tool to use locally or on alternate remote computing resources. Examples of the custom JavaScript commands and usage guidance are provided in Appendix E.

ZDOCK output files must be individually downloaded from the results page of the ZDOCK job, and then processed using the ZDOCK `create.pl` Perl script to obtain all 1000 complexed models. In order to process output files from ZDOCK efficiently, a helper script was written in Python to parse the HTML of the results webpage automatically based on a user specified ZDOCK job ID, and then to build and organise the output models from the relevant results file using the ZDOCK Perl script.

ZDOCK jobs submitted as part of the project were the fastest to complete out of the four docking platforms, and results were generally available within about ten minutes of submission.

4.3.3 LightDock

During the span of this project, LightDock did not have an available web server for modelling jobs. Therefore, LightDock v0.8.0 (Roel-Touris *et al.*, 2020) was installed via the PyPi package (`https://pypi.org/`) in a Miniconda virtual environment (https://docs.conda.io/en/latest/miniconda.html) on Thomas, the UK National Tier 2 High Performance Computing (HPC) Hub in Materials and Molecular Modelling (MMM). Thomas was particularly suited to large multi-core computational jobs, and has recently been retired as an MMM machine and replaced with the new MMM hub, Young. LightDock has since been made available through a web server (https://server.lightdock.org/) as a beta release.

The sampling process in LightDock is driven by an implementation of the glowworm swarm optimisation (GSO) algorithm (Krishnanand and Ghose, 2009). A complex modelled by the docking platform is here imagined as a glowworm that emits a quantity of the bioluminescence-generating compound luciferin as it traverses the energetic landscape of possible docking solutions. The luciferin value emitted by each glowworm agent is determined by an objective function (in this case, the docking scoring function) and is broadcast to its neighbouring glowworms. In each cycle of the algorithm, each glowworm moves towards a probabilistically selected neighbour with a higher luciferin value than its own. These movements result in convergence of glowworms upon locally (or globally) optimal solutions (favourably energetic complexes). Results from each swarming simulation are merged and clustered to remove redundant models. The final representative of each cluster corresponds to the structure with the best energy (Jiménez-García *et al.*, 2018). The LightDock swarming algorithm has been implemented for broad scalability for HPC architectures, and allows users to design their own custom scoring functions.

LightDock makes use of normal mode analysis through an implementation of the anisotropic network model (ANM) for modelling protein flexibility. Under this framework, a protein is treated as an elastic network with nodes representing the alpha carbon atoms of the amino acid residues and edges representing inter-residue potentials that stabilise the protein folding conformation (Atilgan *et al.*, 2001; Doruker *et al.*, 2000). Treating these potentials as harmonic oscillators (springs), the fluctuation dynamics of residues within the protein are described by a superposition of the normal modes (harmonics) of the system. Mathematical decomposition of these modes allows for the displacement of each residue to be calculated, providing a description of protein flexibility. Under this framework, each glowworm represents the extent of deformation along each normal mode for the receptor and ligand, as well as the translation and orientation between the respective components. The mathematics of normal mode analysis and its application to biomolecular structures have been reviewed in detail by numerous research groups (Case, 1994; Ma, 2005; Skjaerven *et al.*, 2009; Bahar *et al.*, 2010; Bauer *et al.*, 2019).

LightDock makes uses of additional information in the form of specified binding residues in both the sampling and scoring procedures. Initial swarms that are not in the proximity of the provided interfaces are filtered out, limiting the search to the binding region. This process also considerably reduces computation time (Roel-Touris *et al.*, 2020). Glowworms are also initially oriented based on random pairings of binding residues in the receptor and ligand. Furthermore, the scoring function is biased according to the percentage of satisfied restraints. LightDock allows binding residues to be defined as either "active" or "passive". Both active and passive restraints are included in the filtering and orienting of the initial poses, but only "active" residues are used to adjust the scoring function.

Default settings were used for each LightDock docking job — 400 initial swarms, with 200 glowworms per swarm and 100 simulation steps. The first 10 non-trivial normal modes for both receptor and ligand were calculated using the LightDock ANM implementation. The default fastdfire function, a fast C implementation of the DFIRE scoring function (Zhou and Zhou, 2002), biased according to the percentage of satisfied restraints (Roel-Touris *et al.*, 2020), was chosen to score docked models. For models produced using Scenario 1 information, TCR loop residues were defined as active and peptide residues as passive; for Scenarios 2 and 3, TCR loop residues were defined as active and pMHC residues as passive; and for Scenario 4, TCR and pMHC residues were all provided as active. This mirrors the active and passive settings used for HADDOCK in this study and related work modelling antibody-antigen complexes (Ambrosetti *et al.*, 2020a).

Prior to the recent LightDock web server, the modelling software was available only via the command line. Despite this, LightDock was very easy to install and run using the online documentation (`https://lightdock.org`). For interested researchers, performing large docking runs will likely require some familiarity with setting up parallel jobs on an available HPC cluster. LightDock jobs in this project were run across 24 cores, and generally completed within 2 hours.

4.3.4 HADDOCK

Models from HADDOCK were generated using the HADDOCK web server (`https://haddock.science.uu.nl`) (van Zundert *et al.*, 2016) version 2.4. Docking runs on HADDOCK are computationally demanding. Modelling through the interactive web server offsets potentially long job times with access to over 100,000 CPU cores made available by the European Grid Initiative (EGI) and associated National Grid Initiatives (van Zundert *et al.*, 2016; Wassenaar *et al.*, 2012).

HADDOCK makes use of a data-driven sampling strategy, encoding known biochemical or biophysical information about the interaction as "ambiguous interaction restraints" (AIRs) that are used to drive the docking (Dominguez *et al.*, 2003). The types of information that can be integrated into the docking protocol are remarkably diverse, including data obtained from mutagenesis and chemical cross-linking experiments (Rodrigues and Bonvin, 2014), nuclear magnetic resonance (NMR) titration analysis (Dominguez *et al.*, 2003), and small angle x-ray scattering (SAXS) profiles (Karaca and Bonvin, 2013). As with the other platforms discussed in this study, interaction restraints were encoded in HADDOCK based on the predicted interfaces described in Section 4.2.

HADDOCK restraints can be defined as either "active" or "passive". Active residues are considered central to the interaction. They are restrained throughout the simulation to always be a part of the interface where possible, and are accounted for in the HADDOCK scoring function (Dominguez *et al.*, 2003). Passive residues are expected to contribute to the interface, but are considered less important to the interaction, and models that contain them are neither penalised nor rewarded by the scoring function. Furthermore, HADDOCK allows for a random fraction of AIRs to be discarded for each docking trial. This option can produce more accurate docked models, due to the possibility of randomly removing bad restraints when working with speculative binding interface data (de Vries *et al.*, 2010).

The HADDOCK docking protocol is split into three stages of increasing levels of model refinement (de Vries *et al.*, 2007). In the first stage (`it0`), rigid body docking is performed through energy minimisation, whereby the sampling is driven by the interaction restraints. In the second stage (`it1`), semi-flexible simulated annealing refinement is performed whereby side-chains and backbone atoms of the interface residues are permitted to move. In the third stage, a short molecular dynamics refinement process is performed in explicit solvent (typically water). Additionally, HADDOCK automatically clusters its final models by either the positional interface ligand RMSD (iL-RMSD) or the fraction of common contacts (FCC) (Rodrigues *et al.*, 2012). Representative models of each cluster are made available for download alongside the full set of models of each refinement stage at the end of a docking job.

Default sampling settings were used for each scenario using HADDOCK: 1000 models for the rigid body (`it0`) stage, and 200 models for the semi-flexible (`it1`) and water refinement stages. While it has been recommended that increased sampling should be used when less information about the binding interface is available, a recent benchmarking of antibody structures using HADDOCK did not show an improvement when sampling was increased compared to the default parameters (Ambrosetti *et al.*, 2020a). For Scenarios 1, 2 and 3, the random removal of restraints was set to the HADDOCK default of 50% for each docking run, while for Scenario 4, the random removal of restraints was disabled. For Scenario 1, the CDR loops were specified as active and the peptide as passive. For Scenarios 2 and 3, the CDR loops were specified as active and the pMHC residues as passive. For Scenario 4, residues selected in both the TCR and in the pMHC were specified as active. HADDOCK models were clustered using the default FCC method using default parameters (FCC cutoff of 0.6 and a minimum cluster size of 4). The average HADDOCK score of the best 4 models of each cluster was used to produce a ranked list of the clusters (de Vries *et al.*, 2010).

The HADDOCK web server offers users three tiers of access for setting up docking jobs. The "easy" tier allows only minimal parameters to be customised. While perhaps more daunting to the novice user than platforms such as ClusPro or ZDOCK due to its wide variety of customisable parameters, it is relatively straightforward to generate models with HADDOCK by using the locked default parameters of this access tier, and by exploring the extensive and impressive set of online tutorials. For more bespoke docking jobs, users may request to be upgraded to "expert" or "guru" tier. While the majority of this study is repeatable using the "easy" tier access level, disabling the random removal of restraints for Scenario 4 requires "expert" level access. Like ClusPro, users are able to monitor the progress of their HADDOCK docking jobs in the workspace of their free (academic) user account. Docking jobs were slowest for HADDOCK out of the four platforms in this study. The run time for HADDOCK jobs has been reported as varying between half an hour and several days depending on the type of docking task (van Zundert *et al.*, 2016). Most jobs in this study completed within approximately 8 hours.

4.4 Model Evaluation Criteria

Class	F_{nat}	LRMSD (Å)	IRMSD (Å)
High	≥ 0.5	≤ 1.0	or ≤ 1.0
Medium	≥ 0.3	≤ 5.0	or ≤ 2.0
Acceptable	≥ 0.1	≤ 10.0	or ≤ 4.0
Incorrect	< 0.1	-	-

Table 4.1: Docked models are assigned a quality based on their calculated F_{nat}, LRMSD and IRMSD.

Every model produced for the four scenarios on each platform was compared with the corresponding reference structure and classified as incorrect, acceptable, medium, or high quality, according to the CAPRI evaluation criteria (Janin *et al.*, 2003; Méndez *et al.*, 2003), shown in Table 4.1. The interface root-mean-square deviation (IRMSD), the ligand root-mean-square deviation (LRMSD) and the frac-

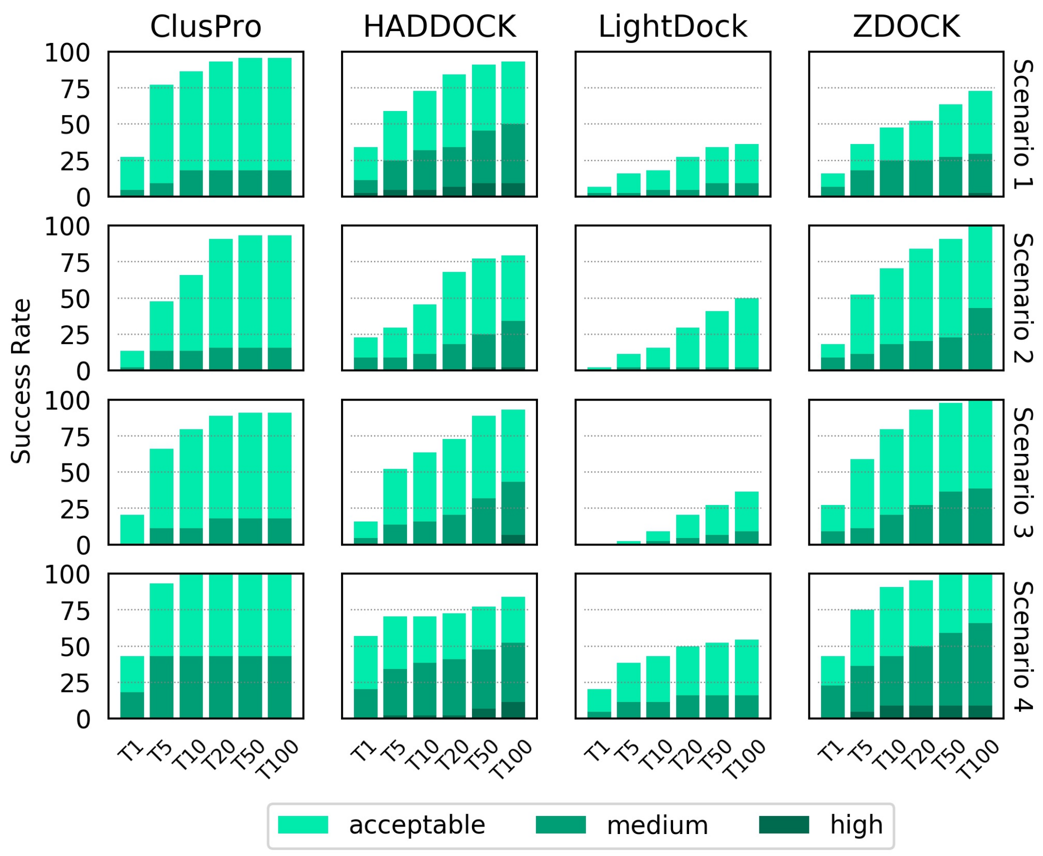


Figure 4.2: Success rate of the top 1, 5, 10, 20, 50 and 100 ranked models for ClusPro, HADDOCK, LightDock and ZDOCK for each of the four docking scenarios. Colour coding indicates the quality of the best model found in a given number of ranked models according to the CAPRI criteria.

tion of native contacts (F_{nat}) are detailed in Appendix A, and were calculated with scripts using the Bio.PDB (Hamelryck and Manderick, 2003) Biopython (Cock *et al.*, 2009) module, with the TCR set as the receptor and the pMHC set as the ligand.

4.5 Results

4.5.1 Docking Performance

The success rate is commonly defined in benchmark docking studies as the percentage of benchmark cases for which a docking platform has achieved a model of adequate quality within the top N ranked models (with N a chosen integer number) (Chen and Weng, 2003; Karaca and Bonvin, 2013; Pierce and Weng, 2013; Ambrosetti *et al.*, 2020a). A comparison of the success rates of ClusPro, HADDOCK, LightDock and ZDOCK across the expanded TCR benchmark data is shown for each scenario in Figure 4.2 as the percentage of cases that feature at least an acceptable, medium or high quality model in the top 1, 5, 10, 20, 50 and 100 ranked models. The 44 cases that feature in this study mean a difference in success rate of approximately 2.3% corresponds to one additional or one fewer benchmark case featuring a successful model. The LightDock results are here left unfiltered and are explored further in Section 4.5.4. ClusPro results are shown according to the default `balanced` scoring function, which is compared to other ClusPro scoring functions in Section 4.5.5.

The top row of Figure 4.2 shows the success rate of each platform when only the CDR loop and peptide residues were provided as binding interface information (Scenario 1). With a success rate of 34.1%, HADDOCK was found to be the best performer for the top ranked model. ClusPro, ZDOCK and LightDock achieved success rates of 27.3%, 15.9% and 6.8% respectively. Broadening the performance analysis to the top 10 ranked models, ClusPro was found to be the best performer, achieving a success rate of 86.4%. HADDOCK, ZDOCK and LightDock achieved success rates of 72.7%, 47.7% and 18.2%, respectively. Across the top 100 models, ClusPro was again found to be the best performer with a success rate of 95.5% (actually achieved within the top 50 models), closely followed by HADDOCK with a success rate of 93.2%. ZDOCK and LightDock achieved success rates of 72.7% and 36.4%, respectively. HADDOCK generated the highest number of medium quality models, followed by ZDOCK, ClusPro and LightDock respectively. HADDOCK generated at least one high quality model for 9.1% of cases in the top 100 models, while ZDOCK generated at least one high quality model for a single case in the top 100 models.

The two middle rows of Figure 4.2 show the success rates of each platform when additionally provided with information about MHC residues involved in, or close to, the binding interface (Scenarios 2 and 3). While HADDOCK achieves the highest success rate for the top ranked model for Scenario 2 with 22.7% (compared to success rates of 18.2%, 13.6% and 2.3% for ZDOCK, ClusPro and LightDock respectively), ZDOCK shows the best performance when using Scenario 3 information, with a success rate of 27.3% (compared to 20.5%, 15.9% and 0% for ClusPro, HADDOCK and LightDock, respectively). Outside of the top ranked model, ZDOCK shows the best performance when using Scenario 2 information, achieving 100% success rate for the top 100 models. When using Scenario 3 information, ZDOCK and ClusPro show the joint best performance for the top 10 models, achieving a success rate of 79.5% (compared to 63.6% and 9.1% for HADDOCK and LightDock, respectively). Outside the top 10 models, ZDOCK shows the best performance using Scenario 3 information and achieves success rates of 97.7% and 100% for the top 50 and top 100 models. In general, Scenario 3 information (derived from the reference structure) achieves slightly higher success rate than Scenario 2 (derived from the unbound structures). ZDOCK was the only platform to show improved performance when using Scenario 2 and Scenario 3 information compared to Scenario 1 information, with ClusPro, HADDOCK and LightDock all achieving lower success rates. HADDOCK was the only platform to generate any high quality models, achieving a success rates of 2.3% and 6.8% in the top 100 ranked models for Scenarios 2 and 3 respectively.

The final row of Figure 4.2 shows the success rate of the four platforms when using the real binding interface information (Scenario 4). HADDOCK showed the best performance for the top ranked model, achieving a success rate of 56.8%, compared to 43.2%, 43.2% and 20.5% for ClusPro, ZDOCK and LightDock, respectively. ClusPro showed the best performance for the top 10 ranked models, achieving a success rate of 100%, compared to 90.9%, 70.5% and and 43.2%, respectively. ZDOCK achieved a success rate of 100% for the top 50 ranked models. Providing the true interface information improved the performance in comparison to the other three scenarios for all the top N ranked models for ClusPro, ZDOCK and LightDock, and for the top ranked and top 5 ranked models for HADDOCK. All platforms showed a higher success rate for the generation of medium quality models. Additionally, both HADDOCK and ZDOCK achieved slightly higher suc-

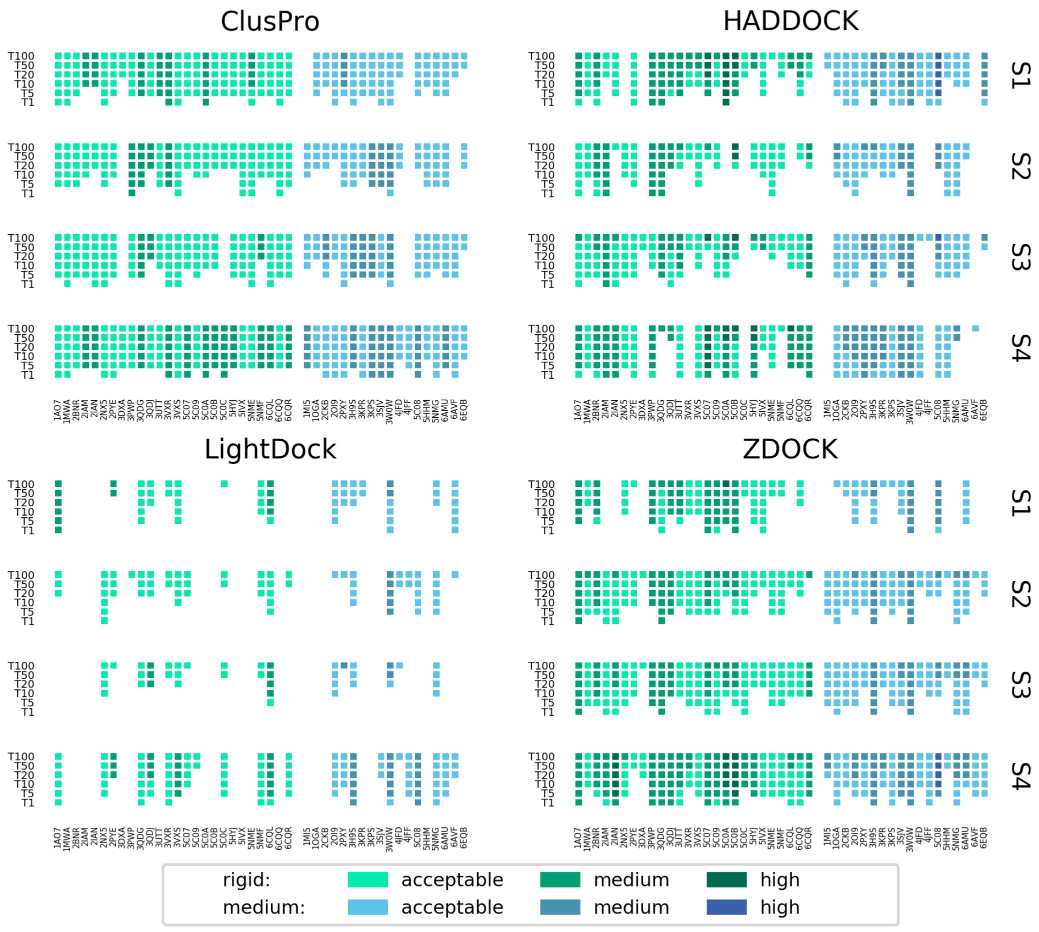


Figure 4.3: Success rate of the top 1, 5, 10, 20, 50 and 100 models for each complex modelled by ClusPro, HADDOCK, LightDock and ZDOCK for each of the four docking scenarios. Complexes are coloured and grouped by their docking difficulty.

cess rates for high quality model generation compared to Scenario 1 for the top 100 models. In general, as might be expected, all platforms show the best performance when supplied with the true binding interface of Scenario 4.

4.5.2 Docking Performance Per Case

Figure 4.3 shows a breakdown of the success rate of each platform for each scenario by individual docking case. Results are grouped and coloured according to the difficulty class, calculated as described in Section 3.3. ClusPro appears to show a largely consistent performance across the benchmark data, whereby if a successful model is generated for a case using Scenario 1 information, it is likely to also be

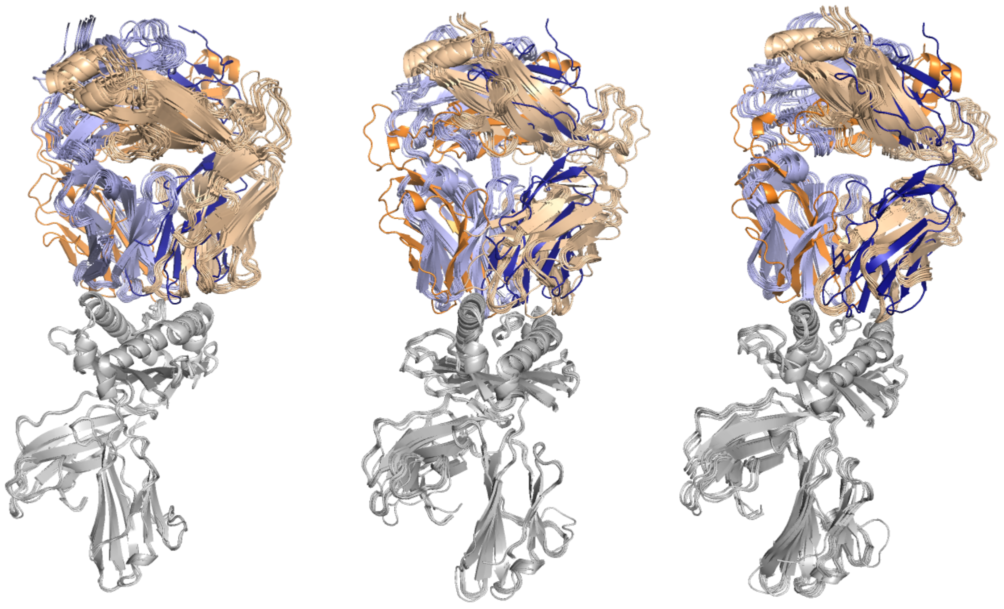


Figure 4.4: Top 10 models produced by HADDOCK using Scenario 4 information and the reference structure for three docking cases: 3DXA (left), 4JFF (middle) and 6EQB (right). For each case, the pMHC chains in the models are superimposed onto the pMHC chains of the reference structure. TCR chains are coloured light orange and light blue in the docked models, and dark orange and dark blue in the reference structures. For these cases, each of the top 10 predictions by HADDOCK has the TCR reversed in orientation relative to the reference structure.

found using the information in Scenario 2, 3 and 4. There are no cases for which a successful model was generated for Scenario 1 but not for Scenario 4. This is not the case for HADDOCK, which, despite achieving the best performance using Scenario 4 information, fails to find acceptable quality models in the top 100 for certain cases for which an acceptable model was generated using Scenario 1 information (for example, cases 3VXR, 5C0C, and 4JFF). Furthermore, HADDOCK appears to perform very well for certain cases across all scenarios (for example, 1AO7, 2PYE and 3W0W) but very poorly for others (for example, 1MI5, 3DXA and 6AVF). ZDOCK, while perhaps more inconsistent that ClusPro across docking cases, generates a higher quantity of medium quality models and some high quality models. In general, rigid difficulty TCR-pMHC complexes do not seem noticeably easier to predict than medium difficulty complexes.

While a complete visual inspection of all models produced by the docking platforms is not feasible due to the quantity of data produced, valuable insight can still be obtained through manual exploration. For certain benchmark cases, HADDOCK performs very well, achieving an acceptable, medium or high quality model as its top ranked model. For other cases, no acceptable model is found in the top 100 ranked models. Examining some of the individual models produced by HADDOCK goes some way towards explaining this variation in success rate across the benchmark data. For many of the cases where HADDOCK performs poorly when provided Scenario 4 information, generated models feature the TCR in what appears to be a sensible position and orientation over the pMHC, but with the α and β chains reversed — that is to say, the TCR α chain sits where the β chain sits in the reference model, and the TCR β chain sits where the α chain sits in the reference model. This is illustrated in Figure 4.4 for cases 3DXA, 4JFF and 6EQB. This pattern was also observed in models produced for cases 1MI5, 3QDG, 5C0C, 5NMF, 5NMG and 6AVF. The TCR and pMHC residues of Scenario 4 are provided independently to HADDOCK — no information is included about which specific residues are in contact between the binding partners. Therefore it appears that HADDOCK generates internally high scoring models, with a large percentage of satisfied restraints, but is blind to the the fact that it has positioned the TCR in a reversed orientation. This phenomenon and its implications for TCR-pMHC modelling are discussed further in Section 4.6.

Docking platform performance per benchmark case are additionally shown grouped and coloured according to MHC class in Figure 4.5. No docking platform appears to disproportionately struggle to model complexes composed of one MHC class relative to the other, although though this conclusion is limited by the small number of Class II structures in the benchmark.

4.5.3 Docking Performance Compared to Bespoke TCR-pMHC Modelling Platform

The TCRFlexDock software is an extension of RosettaDock (Sircar *et al.*, 2010), and the only bespoke platform designed to model TCR-pMHC complexes using computational docking. The TCR-pMHC modelling accuracy of two tailored pro-

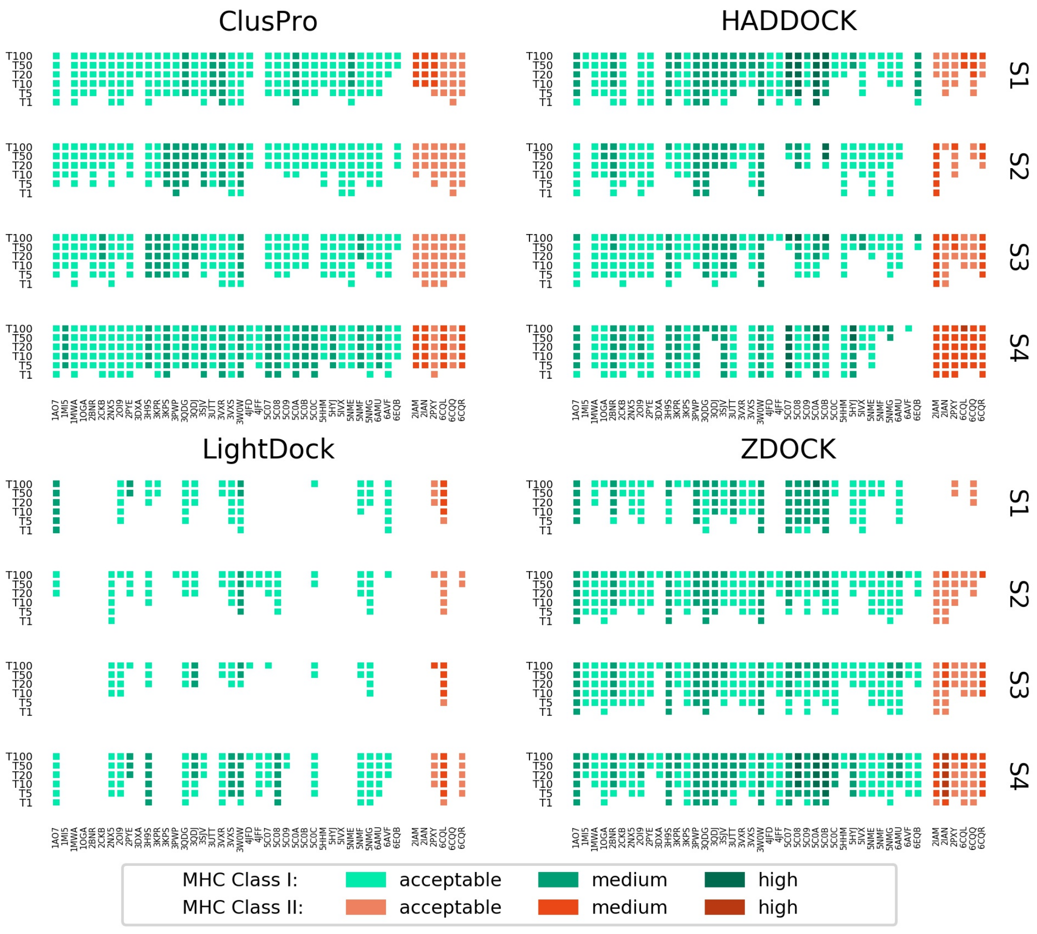


Figure 4.5: Success rate of the top 1, 5, 10, 20, 50 and 100 models for each complex modelled by ClusPro, HADDOCK, LightDock and ZDOCK for each of the four docking scenarios. Complexes are grouped and coloured by their MHC class.

tocols has been reported for the original TCR benchmark data set of 20 TCR-pMHC complexes (Pierce and Weng, 2013). The extent to which these two protocols account for protein flexibility varies — the `CDR3` protocol allows movement to take place in the backbones of CDR3 loops; the `CDRpep` protocol allows movement to take place in all CDR loops as well as in the peptide. The accuracy of these protocols has been compared to the fixed backbone protocol of RosettaDock, `Fixedbb`, and the `CDRPep` protocol was shown to have the best performance against the benchmark data (Pierce and Weng, 2013).

The success rates of ClusPro, HADDOCK, LightDock and ZDOCK were compared to that of the TCRFlexDock `CDRPep` protocol. Using only Scenario 1 information across the expanded 44 benchmark cases, HADDOCK approximately

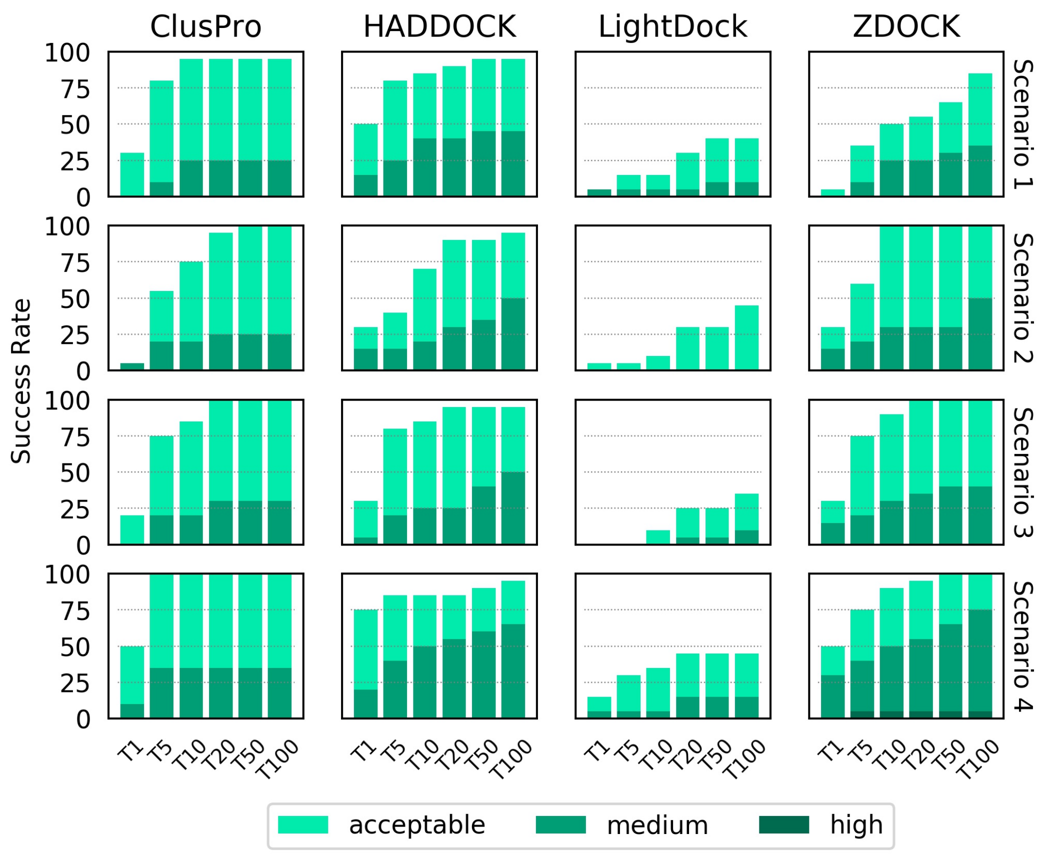


Figure 4.6: Success rate of the top 1, 5, 10, 20, 50 and 100 ranked models for ClusPro, HADDOCK, LightDock and ZDOCK for each of the four docking scenarios for the 20 docking cases assembled to test the TCRFlexDock platform. Colour coding indicates the quality of the best model found in a given set of ranked models according to the CAPRI criteria.

matches the performance of CDRPep across the original 20 benchmark cases for the top ranked model (34.1% vs. 35.0%). Across the top 10 ranked models, HADDOCK performs worse than `CDRPep` (72.7% vs. 80.0%), but ClusPro achieved a higher success rate (86.4% vs. 80.0%). If using the detailed Scenario 4 information, HADDOCK, ClusPro and ZDOCK all outperform `CDRPep` for the top ranked model (56.8%, 43.2%, and 43.2% respectively, vs. 35.0%), and ClusPro and ZDOCK outperform `CDRPep` for the top 10 ranked models (100.0% and 90.1% respectively, vs. 80.0%).

Figure 4.6 shows the success rates of the general purpose platforms for only the original 20 benchmark cases used to evaluate the performance of TCRFlexDock. For this smaller data set, modelling accuracy is improved. Using Scenario 1 information, HADDOCK outperforms `CDRPep` for the top ranked model (50.0% vs. 35.0%), and ClusPro and HADDOCK outperform `CDRPep` for the top 10 ranked models (95.0% and 85.0% respectively, vs. 80.0%). Using Scenario 4 information, HADDOCK, ClusPro and ZDOCK each outperform `CDRPep` for the top ranked model (75.0%, 50.0% and 50.0%, respectively, vs. 35.0%), as well as for top 10 ranked models (85.0%, 100.0% and 90.0% respectively, vs. 80.0%).

4.5.4 LightDock Filtering Performance

After model generation and clustering, the LightDock software recommends filtering out models that do not satisfy a set percentage of the supplied restraints. This approach has been shown to improve the success rate of docking using LightDock for other protein benchmark cases (Roel-Touris *et al.*, 2020). A filtering threshold of 40% of satisfied restraints was applied to the LightDock model, chosen in accordance with the LightDock tutorial documentation and the antibody-antigen modelling of Ambrosetti *et al.* (2020a). The percentage of satisfied restraints is calculated as

$$percentage = \frac{c_{rec} \cap r_{rec} + c_{lig} \cap r_{lig}}{r_{rec} + r_{lig}} \tag{4.1}$$

where c_{rec} and c_{lig} are the contacts in the receptor and ligand of a modelled complex respectively, and r_{rec} and r_{lig} are the supplied receptor and ligand restraints respectively.

The first two columns of Figure 4.7 show the LightDock success rate across the benchmark cases before and after filtering. The filtering step proved to have a negative impact on results generated using Scenario 1 information, and particularly using Scenario 2 and 3 information where all models were filtered out. Scenario 4 results remained unchanged. In an attempt to reduce potential over-filtering, the threshold was reduced, but results failed to improve. Consequently, results shown in Figure 4.2 for LightDock remain unfiltered, as published in Peacock and Chain (2021).

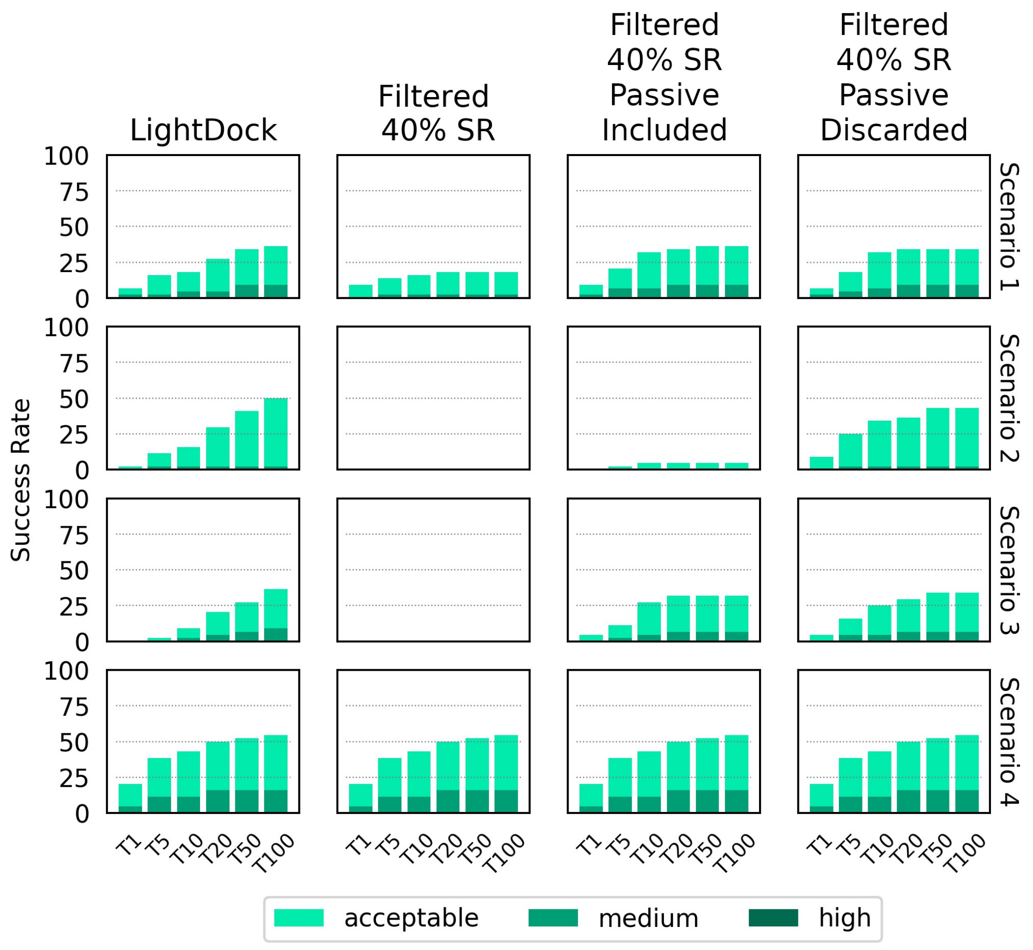


Figure 4.7: The docking success rate is shown for LightDock without the recommended filtering of models by the satisfaction of spatial restraints (first column). This is compared to filtering by the satisfaction of at least 40% of spatial restraints using: the LightDock v0.8.0 filtering script (second column); an adapted filtering script that takes into account passive restraints (third column); and an adapted filtering script that ignores passive restraints (fourth column).

Recent analysis of the `lgd_filter_restraints.py` script has highlighted an issue in LightDock v0.8.0, whereby restraints, r_{rec} and r_{lig}, labelled as "passive", are not correctly formatted for use in Equation 4.1. This problem explains why results generated using Scenario 2 and 3 were the most badly affected by filtering, as they feature the largest numbers of passive restraints. Information provided as Scenario 4 features only active restraints, and the success rate results were unaffected.

Two approaches were taken in an attempt to rectify the LightDock filtering step. The first approach involved modifying the `lgd_filter_restraints.py` script to correctly format the passive restraints for inclusion in the percentage of satisfied restraints calculation, such that

$$c_{rec} = c_{rec\_active} \cup c_{rec\_passive} \tag{4.2}$$

$$c_{lig} = c_{lig\_active} \cup c_{lig\_passive} \tag{4.3}$$

$$r_{rec} = r_{rec\_active} \cup r_{rec\_passive} \tag{4.4}$$

$$r_{lig} = r_{lig\_active} \cup r_{lig\_passive} \tag{4.5}$$

The LightDock success rates after applying this filtering are shown in the third column of Figure 4.7.

The second approach involved modifying the `lgd_filter_restraints.py` script to ignore the passive restraints entirely, such that

$$c_{rec} = c_{rec\_active} \tag{4.6}$$

$$c_{lig} = c_{lig\_active} \tag{4.7}$$

$$r_{rec} = r_{rec\_active} \tag{4.8}$$

$$r_{lig} = r_{lig\_active} \tag{4.9}$$

The LightDock success rates after applying this filtering are shown in the fourth column of Figure 4.7.

Filtering with correctly formatted passive restraints produces a modest improvement in the success rate of LightDock when using Scenario 1 and Scenario 3 information. The performance worsens when using Scenario 2 information in comparison to the unfiltered results. Filtering with only active restraints produces a modest improvement in the success rate when using Scenario 1, Scenario 2, and Scenario 3 information. In both cases, filtering for Scenario 4 information did not produce a change in the results. While these new filtering approaches improve the success rates relative to leaving results unfiltered, the performance of LightDock remains below that of ClusPro, HADDOCK and ZDOCK for the TCR docking

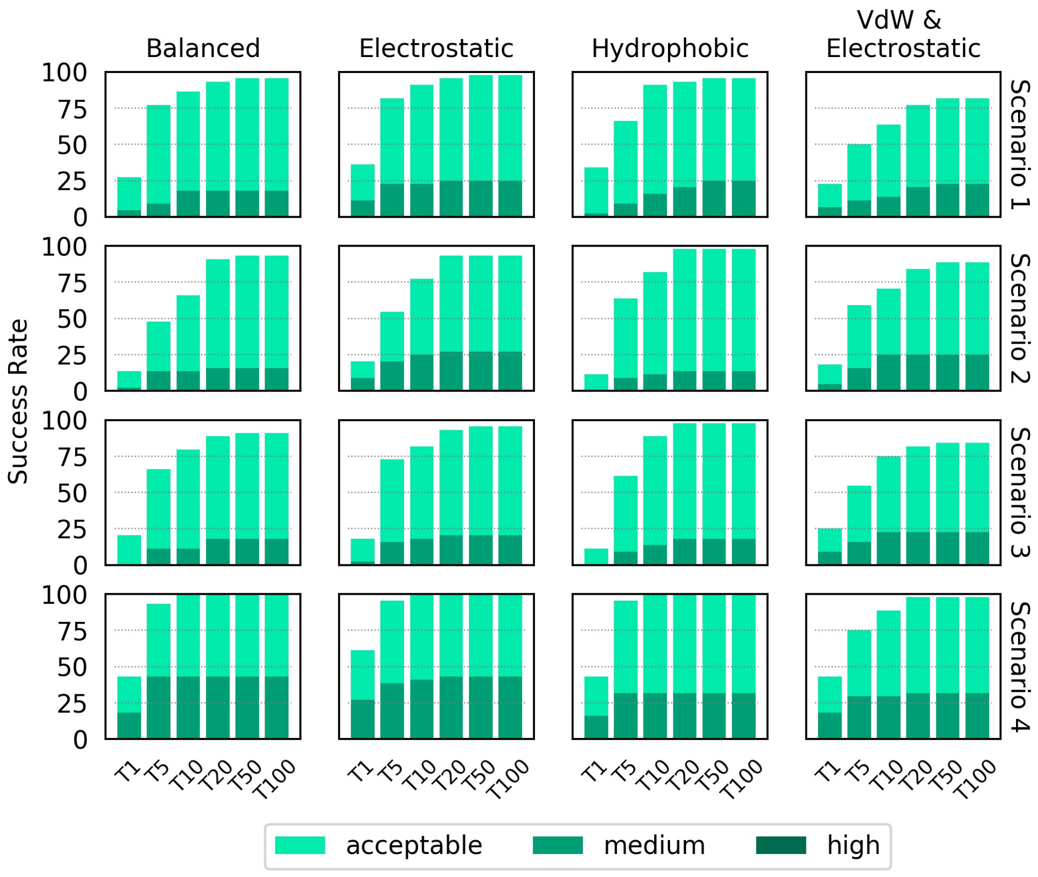


Figure 4.8: The docking success rate is shown for the top 1, 5, 10, 20, 50 and 100 ranked models generated by ClusPro using the four scoring functions provided by the ClusPro web server.

benchmark structures. Were these changes to be implemented into a new version of LightDock, it would be worthwhile verifying these improvements against an independent set of protein-protein test cases.

4.5.5 ClusPro Scoring Function Performance

At the end of a docking task, the ClusPro server produces four sets of models generated using four different scoring functions — `balanced`, `electrostatic-favored`, `hydrophobic-favored`, and `van der Waals + electrostatics`. The `balanced` function is recommended for the modelling of complexes where no particular interaction properties are assumed (Kozakov *et al.*, 2017), and is shown in Figure 4.2.

Results generated using the other three scoring functions are compared to the `balanced` function in Figure 4.8. The `electrostatic-favored` and `hydrophobic-favored` functions outperform the `balanced` function for the top ranked model and top 10 ranked models using Scenario 1 information. The `electrostatic-favored` function outperforms the `balanced` function for the top ranked model using Scenario 2 information, and for the top 10 ranked models, alongside the `hydrophobic-favored` function. The `electrostatic-favored` and `hydrophobic-favored` functions worsen performance for the top ranked model using Scenario 3 information, but improve performance across the top ten models. While generally worsening performance, the `van der Waals + electrostatics` function improves performance for the top 10 ranked models using Scenario 2 information and for the top ranked model using Scenario 3 information. Finally, the `electrostatic-favored` function improves performance for the top ranked model using Scenario 4 information.

4.5.6 HADDOCK Cluster Performance

The models generated by the HADDOCK platform were automatically clustered by the fraction of common contacts (FCC) using the default cut-off of 0.6 and a minimum cluster size of 4. Clusters were ranked according to the average HADDOCK score of the best 4 models of each cluster. The success rates of the top four members of the top five clusters is shown in Figure 4.9. Additionally a breakdown by individual docking case is shown in Figure 4.10. When using Scenario 3 and 4 information, the HADDOCK success rates improve using the cluster-based scoring method. However, when using the more vague interface descriptions of Scenario 1 and Scenario 2, performance decreases. As has been highlighted by previous studies (Ambrosetti *et al.*, 2020a), the decision to rely on the clustered HADDOCK results rather than the full set of models should be carefully chosen based on the level of detail known about the TCR-pMHC binding interface.

4.5.7 Sampling Performance

The sampling stage of the docking process involves the exploration of possible solutions through the generation of thousands of modelled complexes of the two binding partners. Figure 4.11 shows the percentage of models generated for a specific

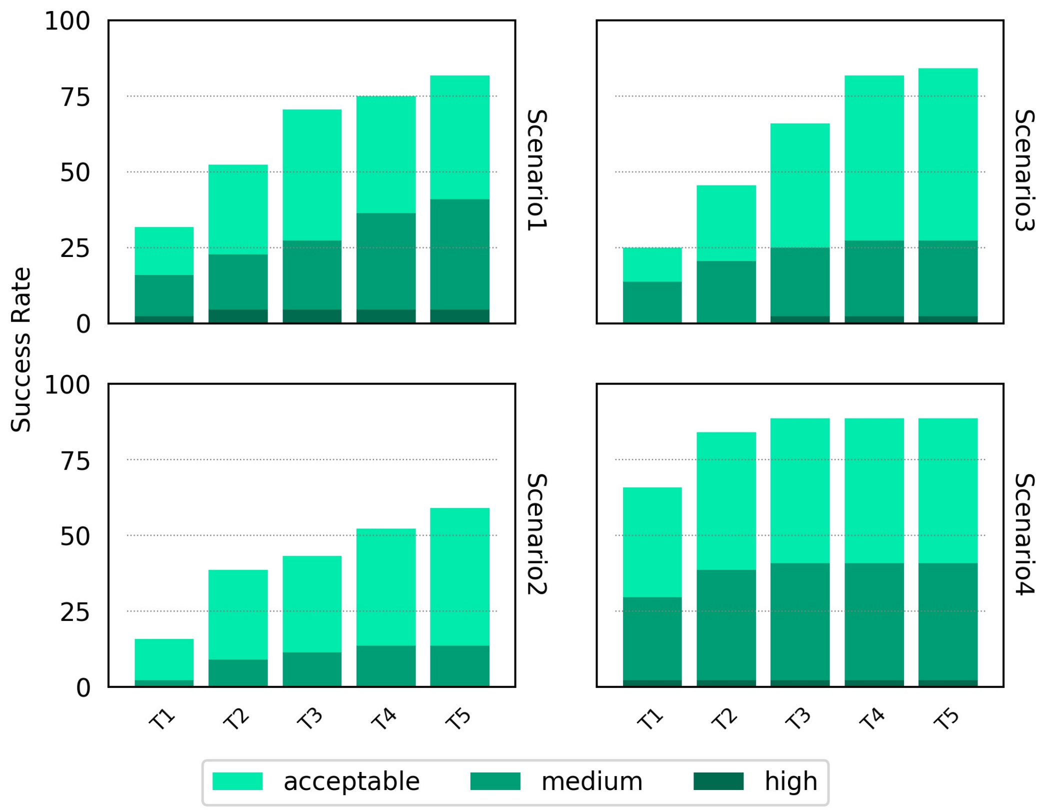


Figure 4.9: Success rate for the top 1, 2, 3, 4 and 5 ranked clusters for HADDOCK for the four docking scenarios. Colour coding indicates the quality of the best model found in a given set of ranked models according to the CAPRI criteria.

docking case that were of acceptable, medium and high quality for each docking platform and scenario. It should be noted that a different scale is used for each of the four platforms to aid readability.

The sampling performance of HADDOCK is striking in comparison to the other platforms. For several benchmark cases, every single model generated by HADDOCK using Scenario 4 information was of at least acceptable quality (for example: 2NBR, 3PWP, 2PXY). Furthermore, providing the real information about the binding interface through Scenario 4 produces a stronger sampling performance than the other three scenarios. The strengths of the HADDOCK sampling strategy have been reported before and attributed to its use of the supplied binding information to drive the energy minimisation and molecular dynamics steps of the simulation, rather than only at the scoring stage like most docking protocols (Ambrosetti *et al.*, 2020a).

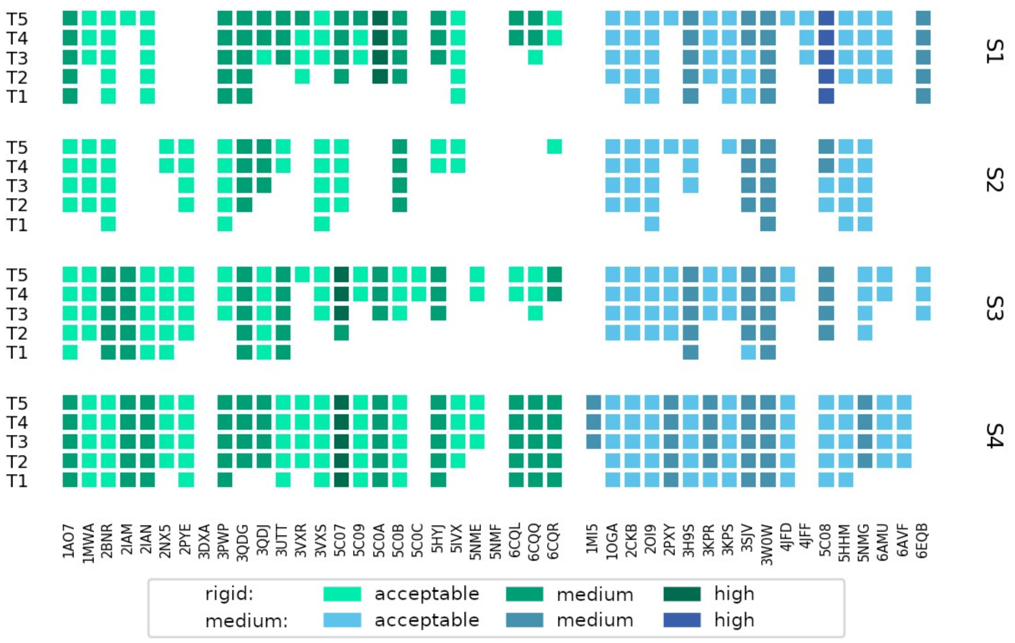


Figure 4.10: Success rate for the top 1, 2, 3, 4 and 5 ranked clusters for each complex modelled by HADDOCK for the four docking scenarios. Complexes are coloured and grouped by their docking difficulty.

ClusPro achieved the second best sampling performance, which can be likely attributed to the removal of somewhat redundant solutions through its eponymous clustering algorithm, resulting in a low number of total models. Nevertheless, for no docking case were more than 25% of the models of at least acceptable quality for any scenario.

4.5.8 CDR3 Loop Modelling Performance

While ClusPro and ZDOCK are rigid-body docking platforms, HADDOCK and LightDock allow the protein components to undergo conformational change during the docking procedure. The ability of the two platforms to accurately model TCR CDR loops in the TCR-pMHC complex structure from the loops found in the independently crystallised TCR and pMHC components was explored following a strategy that has been recently applied to the modelling of H3 antibody loops (Ambrosetti *et al.*, 2020a).

For each benchmark case, the framework region residues of the unbound TCR were superimposed upon those of the bound reference structure. The all-atom

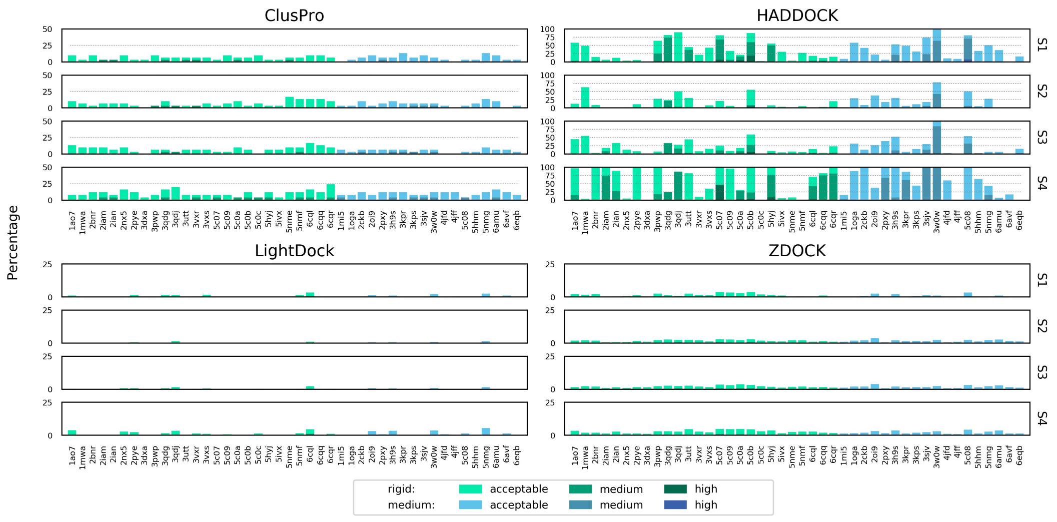


Figure 4.11: Percentage of total models for each complex of acceptable, medium or high quality according to the CAPRI criteria, shown for ClusPro, HADDOCK, LightDock and ZDOCK for each of the four docking scenarios. Colour coding indicates the quality of models for both rigid and medium difficulty docking cases. The y axis is scaled differently for each docking platform to aid readability.

RMSD of the residues in each CDR loop of the two structures was calculated to determine a baseline measure of similarity between the bound and unbound loops prior to docking. The same procedure was then carried out for each docked model. The RMSD between each docked model loop and the reference structure loop was compared to the baseline RMSD to assess whether the modelled loops were closer to (or further away from) the reference structure loop than the starting unbound structure loop.

Figure 4.12 shows the modelled-to-reference RMSD compared to the unbound-to-reference RMSD for the α chain CDR3 loop for each of the top 100 models produced for each docking case. The results are displayed for both HADDOCK and for LightDock, for each scenario. Figure 4.13 shows the results of the same procedure applied to the β chain CDR3 loop. Values below the diagonal line correspond to an improvement in the loop conformation, whereas values above the line correspond to a worsening in the loop conformation. Models are coloured according to their quality.

HADDOCK produces models with both improved and worsened CDR3 loop

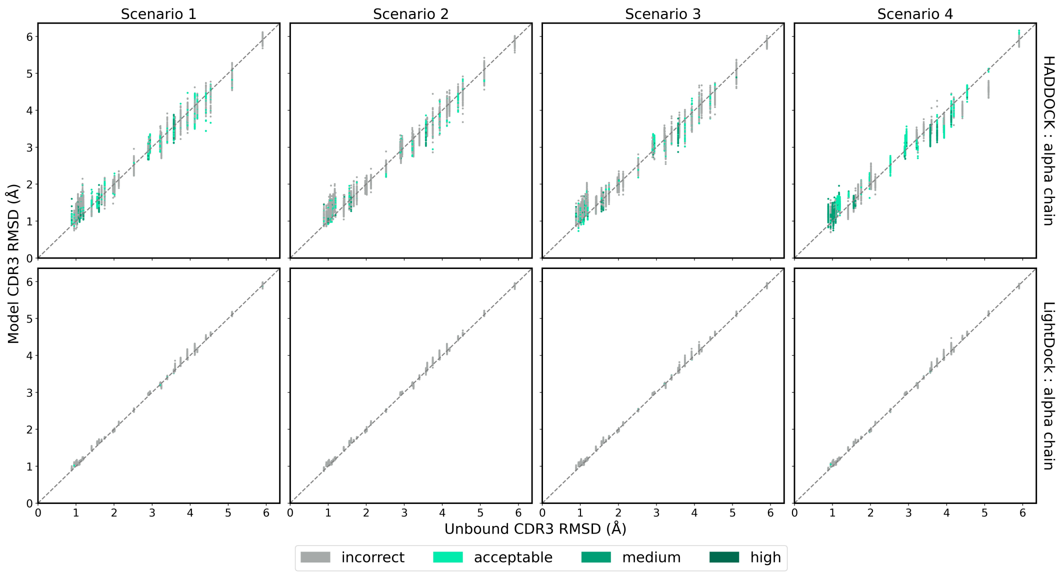


Figure 4.12: The RMSD of the TCR α chain CDR3 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

RMSDs. In general, for complexes that undergo low conformation change upon binding, the flexible refinement leads to a worsening of the loop CDR3 RMSD. However, for complexes that undergo greater conformational change, the flexible refinement does not appear to worsen RMSD overall. This pattern is also evident in the modelling of the CDR1 and CDR2 loops, shown in Appendix F. Loop flexibility in models produced by LightDock are minimal, but generally worsen the CDR3 loop RSMD for both α and β chains. Again, the same pattern is observed in the CDR1 and CDR2 loops.

4.6 Discussion

The work in this chapter has comprised an assessment of the ability of four general purpose docking platforms — ClusPro, HADDOCK, LightDock and ZDOCK — to model TCR-pMHC bound complexes accurately from unbound TCR and pMHC components. This serves as an expansion of work published in Peacock and Chain (2021) and mirrors a recently published comparison of the same software suites in the context of antibody-antigen modelling (Ambrosetti *et al.*, 2020a). Each platform

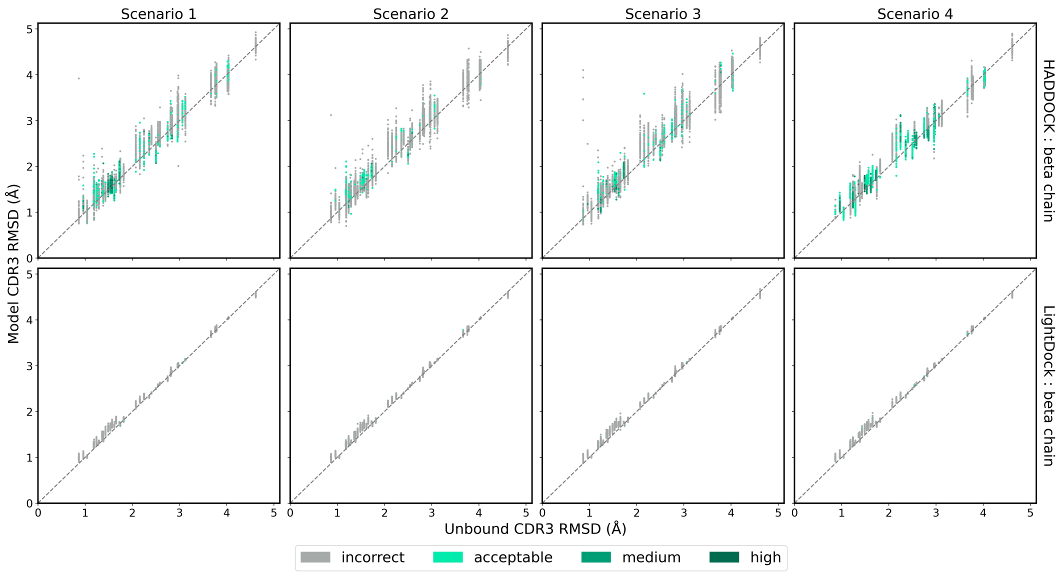


Figure 4.13: The RMSD of the TCR β chain CDR3 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

facilitates the inclusion of additional information to aid its modelling capabilities through the specification of binding interface residues. As these residues are not often known when aiming to perform computational docking, each platform was benchmarked across 44 TCR-pMHC docking cases in the context of four different descriptions of the binding interface that varied in their specificity.

The binding interface interactions of TCRs and antibodies are both dominated by six flexible CDR loops. Despite this similarity, as well as their relatively more conserved binding mode (Dunbar *et al.*, 2014), the results of the four platforms are less impressive when modelling TCR-pMHC structures than when modelling antibody-antigen structures (which are, in turn, typically less impressive than when modelling other types of protein-protein complexes). Antibody-antigen complexes are more frequently modelled than TCR-pMHC complexes and have long been a staple of the protein-protein docking benchmark (Chen *et al.*, 2003), with which most docking software platforms are assessed. Consequently, the docking platforms included in this study have likely been designed, to some extent, with antibody-antigen modelling in mind. ClusPro, in particular, has a dedicated antibody-antigen modelling feature available as part of its docking suite (Brenke *et al.*, 2012). The attention received by antibodies when new docking approaches are developed could partly explain the superior performance for antibody-antigen modelling in comparison with TCR-pMHC modelling. The binding affinity between antibody and antigen is known to be much stronger than that between TCR and pMHC (van der Merwe and Davis, 2003). While docking scoring functions are not an accurate reflection of binding affinity measurements (Pantsar and Poso, 2018), it could be speculated that the low affinities of TCRs additionally contribute to the increased difficulty in TCR-pMHC modelling.

The LightDock platform was found to struggle particularly with TCR-pMHC modelling, despite strong performances when tested with antibodies (Ambrosetti *et al.*, 2020a; Roel-Touris *et al.*, 2020). It remains unclear whether the main challenges in improving results lie in the sampling or scoring stages, as neither have been extensively explored. HADDOCK offers the best sampling performance of the four platforms and is the only platform to drive the sampling with the provided binding information. This more restricted type of search in the range of the typical TCR binding mode could yield higher quality results for future bespoke TCR-pMHC methods rather than incorporating binding residues into only the model scoring process.

It would also be informative to explore the effect of different scoring functions and parameters in the context of TCR-pMHC modelling. This chapter has highlighted improvements in overall model accuracy by ClusPro when the weighting of the electrostatic term in the scoring function was increased (the `electrostatic-favored` function), and a decline in performance when the hydrophobic term was removed (the `van der Waals + electrostatics` function). Both hydrophobic and electrostatic interactions have been highlighted as important effects in TCR-pMHC recognition (Singh *et al.*, 2017), and it would be interesting to explore the effects of these scoring function terms in more detail in future research. Through a visual inspection of modelled complexes for which HADDOCK performed poorly, it was found that the α and β chains were often reversed in the model compared to the reference structure. This information could be taken into account in bespoke TCR-pMHC docking scoring or filtering approaches, provided the researcher is confident in which way round the chains should sit relative to the pMHC prior to modelling. The HADDOCK unambiguous pairwise restraints functionality was not explored in this project, but might offer a solution for these complexes. The LightDock platform offers a number of built in scoring functions aside from the default `fastdfire` function and additionally facilitates the incorporation of custom scoring functions. The design and exploration of bespoke TCR-pMHC scoring functions would also make for an intriguing prospective research project.

The two strongest performing docking platforms both make use of clustering to improve their modelling accuracy — ClusPro as a fundamental step in its algorithm and HADDOCK as a optional step to produce an alternate set of final models. The results in this chapter suggests that, for HADDOCK, a balance must be struck between the level of detail known about the binding interface and the decision to use the clustered output. Performance noticeably worsened when supplying binding residues from Scenario 2, which features the most MHC residues out of all four scenarios. The results suggest that an over-specification of MHC residues, many of which are not directly involved in the binding, has a negative impact on the clustering. In these circumstances, as HADDOCK performs a restricted search around the supplied restraints, models are likely to be more frequently clustering in regions where MHC residues not involved in the true interface are located. On the other hand, when supplying only the true binding residues of Scenario 4, models are likely to more frequently cluster in regions including true binding residues, and, as such, performance noticeably improves. The alternative approach of not restricting the search space also seems to prove effective in combination with model clustering, as highlighted by the consistent performance of the ClusPro platform.

Flexible docking has long been considered an important method for improving modelling accuracy (Pagadala *et al.*, 2017). The interaction between the TCR and pMHC is known to be driven by the CDR loops, with the CDR3 loop in particular being key to the recognition process. The modelling of these flexible loops remains a difficult and important problem in the field. TCRFlexDock, a bespoke platform for TCR docking, has shown improvement when allowing for flexibility in the CDR loops and the peptide. However, the two platforms that offer flexible refinement analysed in this study — HADDOCK and LightDock — were unable to improve the conformation of CDR loops consistently regardless of the additional information provided. It can be concluded that the current successes of all four platforms are a result of accurate modelling of position and orientation rather than of conformational change.

Nevertheless, despite an inability to model flexibility effectively, HADDOCK and ClusPro in particular are competitive with TCRFlexDock for TCR-pMHC docking. The success rate for HADDOCK using Scenario 1 information across the 44 benchmark cases used in this study approximately matches that of TCRFlexDock across the original 20 benchmark cases. If only the original 20 cases are considered, HADDOCK noticeably outperforms TCRFlexDock using Scenario 1 information. This information is easily accessible from the unbound components. If the top 10 ranked models are considered, ClusPro outperforms TCRFlexDock using Scenario 1 information for both the expanded and original TCR docking benchmarks. Additionally, when supplying the true interface residues of Scenario 4, ClusPro, HADDOCK and ZDOCK all achieve higher success rates than TCRFlexDock for the top, and top 10, ranked models. Two major advantages of these three platforms, in comparison to TCRFlexDock, are their ease of use and their computational run times. Each platform is supported by a readily accessible online modelling server, with little (if any) additional software or computational resources required by the modeller. The ZDOCK server often completes a docking run within ten minutes, while the HADDOCK server can take several hours. In contrast, TCRFlexDock requires some familiarity with the somewhat complex Rosetta modelling framework for running docking jobs and has been recently reported as taking over 100 hours per complex in its current implementation (Jensen *et al.*, 2019).

As might be expected, ClusPro, HADDOCK, LightDock and ZDOCK all showed the strongest performance when provided with only the residues that were determined to be directly involved in the binding interface (Scenario 4), highlighting the role accurate binding information can play in TCR-pMHC modelling. When provided with the vaguest information about the binding interface — simply the CDR loop and peptide residues (Scenario 1) — ClusPro, HADDOCK, and LightDock all achieved higher success rates than when additionally providing MHC residues likely to be close to the interface (Scenarios 2 and 3). The poorer performances achieved with Scenario 2 and Scenario 3 information suggests that forcing these platforms to accommodate a large number of MHC residues in their sampling and scoring processes leads to the true binding residues being lost in a background of residues that are not directly involved in the binding, resulting in a less specific set of results. In contrast to these findings, the accuracy of ZDOCK improved when supplied with the broad MHC residue selections of Scenarios 2 and 3. Rather than scoring models based upon the residues supplied as being involved in the binding, ZDOCK scores models based on the blocking of residues known to not be involved in the binding. This approach is less restrictive in satisfying the MHC residues restraints of Scenarios 2 and 3, and likely explains the improved success rate for ZDOCK for these scenarios compared with Scenario 1. Therefore, it is important to consider carefully which residues to include when specifying unknown binding information. When specifying binding residues that would be desirable in the output models, a narrow selection is preferable. Alternatively, when specifying residues that should not be blocked by the scoring function, a wider selection may be provided.

Which platform is most suitable for modelling depends upon the amount of information available and the required quality of the output model. Overall, the HADDOCK platform is the best performer for generating accurate TCR-pMHC complexes as the top ranked model. If the required results are not limited to the top ranked model, ClusPro is the most consistent performer. In the absence of detailed information about the binding interface, it is recommended that users specify only the CDR loop residues of the TCR and the peptide residues of the pMHC as binding information for both HADDOCK and ClusPro. If using ClusPro, users may yield more accurate models by making use of the `electrostatic-favored` scoring function. If using ZDOCK, specifying MHC residues close to the interface in addition to the peptide residues will likely improve modelling results.

Despite some of the successes shown in this study, it is clear that there is opportunity for improvement in the computational docking of TCR-pMHC complexes. Firstly, in many of the examined docking cases across the four platforms, acceptable (or higher) quality models can be found well outside of the top ranked model. Novel tools for re-ranking or filtering docked TCR-pMHC models could improve the overall success rate of the platforms considered in this model. Bespoke scoring functions could be designed for TCR-pMHC-specific modelling platforms.

Secondly, there are improvements to be made in the modelling of the flexible CDR loop regions. To date, a number of groups have formulated descriptions of canonical forms for TCR CDR loops (Al-Lazikani *et al.*, 2000; Klausen *et al.*, 2015; Wong *et al.*, 2019; Fernández-Quintero *et al.*, 2020), while others have attempted to model TCR CDR loop flexibility in unbound contexts (Gowthaman and Pierce, 2018). Some investigation has been conducted into the modelling of conformational change that takes place in the loops upon binding (Jensen *et al.*, 2019), though only the TCRFlexDock platform does so in a docking setting (Pierce and Weng, 2013). Although very challenging problems, improvements in these areas would no doubt produce more accurate TCR-pMHC models. Thirdly, results in this chapter have shown that docking platforms do not appear to show stronger modelling accuracy for "rigid" difficulty TCR-pMHC cases than for "medium" difficulty cases. While this classification has proved informative for protein docking cases in general, perhaps the modelling of TCR-pMHC complexes would benefit from a new difficulty classification system.

Finally, it is evident that when exact residues of the binding interface are known, model accuracy dramatically improves. A number of computational methods have been developed that attempt to predict contact residues between antibody and antigen (Kunik *et al.*, 2012; Krawczyk *et al.*, 2013; Liberis *et al.*, 2018; Daberdaku and Ferrari, 2019; Deac *et al.*, 2019; Ambrosetti *et al.*, 2020b; Akbar *et al.*, 2021), however no such methods have been applied to TCR-pMHC structures. Tools for the accurate prediction of TCR-pMHC contact residues would be an extremely useful aid for information-driven modelling of TCR-pMHC complexes.

Chapter 5

Structural Optimisation for TCR Design

5.1 Introduction to TCR Design

Traditionally, the use of structural information in the design of novel TCRs has been used to carefully analyse and modify particular residues of a TCR known to already bind a given antigen in order to enhance some property, such as its binding affinity. The ever-increasing accuracy of protein modelling tools and computational speeds offered by modern hardware provide an opportunity for us to speculate about how structural modelling may be integrated with other approaches or pipelines for automated protein design.

This chapter outlines a prototype computational framework that aims to enhance desirable properties in TCR structures by artificially evolving them over successive generations using a genetic algorithm optimisation routine. TCR or TCR-pMHC structures are generated from sequence using homology modelling tools. The modelled structures are evaluated according to a function that assigns high scores to structures with desirable properties and low scores to structures that do not have desirable properties. High scoring structures are preferentially selected by the algorithm, and undergo crossover and mutation of various amino acids within their sequences. These sequences are then modelled to produce a new generation of structures, with an aim of producing fitter and fitter structures over the course of the run.

The following sections describe the principles of genetic algorithms, and a bespoke platform for TCR design. Two applications of the pipeline are demonstrated, whereby the number of contacting residues between TCR homology model α and β chains, and between TCR and pMHC homology modelling chains, are increased in quantity. Finally, limitations and future perspectives of the framework are discussed.

5.2 Genetic Algorithms

The genetic algorithm (GA) is a heuristic problem solving technique that belongs to a larger class of evolutionary algorithms that are inspired by natural selection. These algorithms are typically employed to generate high quality solutions to complex problems where the search space is too large to sufficiently test every candidate solution. An initial population of candidate solutions are evaluated according to some measure of "fitness", and then evolved over successive generations, through breeding and mutation, selecting for higher quality (fitter) solutions. Other population-based evolutionary algorithms include ant colony, glowworm and particle swarm algorithms. More broadly, other heuristic algorithms routinely used for optimisation problems include first order algorithms such as gradient descent, second order algorithms such as the Newton-Raphson Method, and stochastic algorithms such as simulated annealing, amongst others.

GAs have been used extensively over several decades as a computational approach for a wide range of biological problems. In structural biology they have seen considerable application to the protein folding problem (Unger, 2004), and have been used to intelligently navigate conformation space for complex prediction via protein-protein docking (Gardiner *et al.*, 2001). GAs have been used for lead optimisation of compounds (Singh *et al.*, 1996; Spiegel and Durrant, 2020) and peptides (Fjell *et al.*, 2011; Röckendorf *et al.*, 2012; Burnside *et al.*, 2019) in virtual drug discovery. Furthermore, integration of a GA with structural modelling techniques has been employed in the optimisation of peptides for MHC binding (Knapp

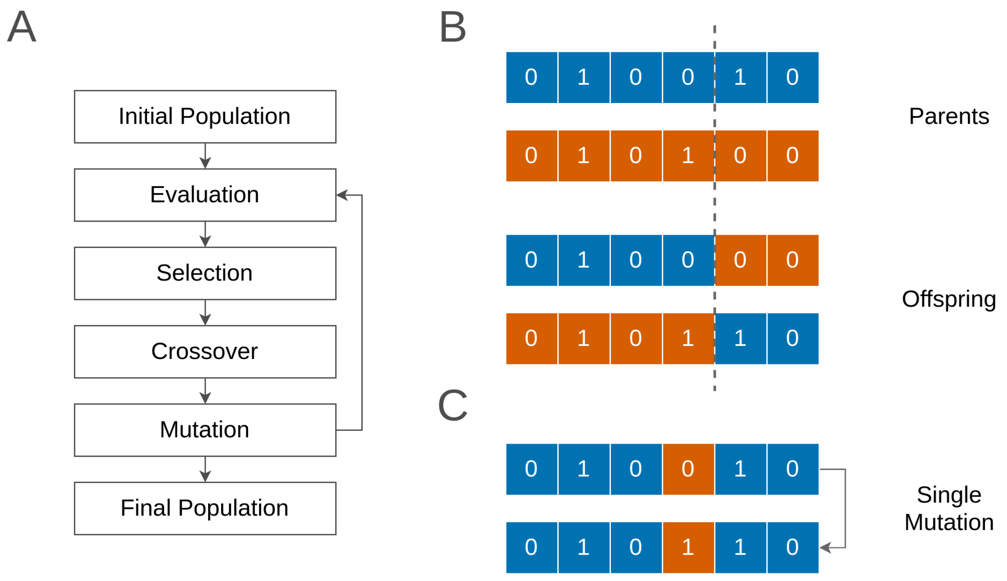


Figure 5.1: (A) Schematic of a simple genetic algorithm. (B) Single point crossover operation, whereby two parent individuals are mated to produced two offspring individuals. The dashed line denotes the crossover point beyond which the encoded "genes" are swapped between individuals. (C) Mutation operator, whereby a randomly selected "gene" in an individual is mutated.

et al., 2011).

The theory and implementation of genetic algorithms are supported by a dense body of research which are covered in detail elsewhere (Holland, 1992; Mitchell, 1996). The principles of a generic GA are described below, and illustrated in Figure 5.1.

5.2.1 Initialisation

The algorithm begins with the generation of an initial population of individuals. Each individual is a candidate solution to the given optimisation problem. Borrowing from biological terminology, an individual is often referred to as a "chromosome" and is usually encoded as a bit string. The "genes" that make up this chromosome are represented as either single bits or blocks of bits, and encode features of the candidate solution. In most problem settings, it is simplest, but by no means necessary, to represent an individual using a single chromosome. Candidate solutions are typically randomly generated, but are sometimes seeded according to some known information to narrow the search space and computational run time.

5.2.2 Evaluation

The quality of each individual in a given generation is evaluated according to a problem-specific fitness function. The fitness function assigns a score to each individual that reflects how successful the candidate solution is in exhibiting the desired properties of the optimisation task. The function should assign high fitness scores to those individuals that are performing well for the set task, and low fitness scores for those individuals failing to meet the desired criteria. Individuals with higher fitness scores are more likely to be selected as “parents” for the next generation of individuals.

5.2.3 Selection

The selection operator selects evaluated individuals from the population for reproduction. The fitter an individual, the more likely it is to be selected. Those that are selected are considered “parents” of the next generation, and are subject to crossover and mutation operations to produce “offspring” individuals. Selection must be balanced against crossover and mutation to avoid high-quality but sub-optimal individuals taking over the population, as well as to avoid the persistence of too many low-quality individuals in the population, resulting in slow evolution.

A wide range of strategies have been developed to perform selection. A commonly used approach is fitness-proportionate selection, implemented through “roulette wheel” sampling. Here, each individual is assigned a slice in the roulette wheel, with the slice size proportional to the fitness of the individual. The wheel is spun a number of times equal to the population size, with an individual selected as a parent chromosome on each spin. This approach provides low-quality solutions a chance of making it through to the next generation, thereby maintaining diversity in the population.

Tournament selection is a more efficient selection strategy that can be implemented in parallel. Two (or more) individuals are chosen at random from the population and enter a tournament. A random number is chosen between 0 and 1 and compared to a predefined threshold. If the number is below the threshold, the fitter individual is selected to be a parent. If the number is above the threshold, the less fit individual is selected to be a parent. The tournament process is repeated a number of times equal to the population size. Various alternative strategies have been accessibly outlined by Mitchell (1996).

5.2.4 Crossover

The crossover operator exchanges subsequences between two selected parent chromosomes to produce two offspring chromosomes that share information from both parents. Not all selected parents are subject to crossover, but rather are given some set probability of being subject to the operation. The crossover process is illustrated in Figure 5.1B using a single crossover locus. Single-point crossover is the simplest form of the operator, but two-point and parameterised uniform crossover (which avoids positional bias), amongst many others, have also been commonly implemented. The locus beyond which subsequences are exchanged is typically randomly selected, although some approaches have implemented crossover "hotspots", specifying particular locations at which crossover should take place in a given bit string.

5.2.5 Mutation

The mutation operator introduces mutations into selected parent chromosomes. For binary strings, this is accomplished by randomly flipping some of the bits of the given chromosome. This process is illustrated in Figure 5.1C. For alternate base systems, such as for chromosomes encoded as amino acids, a mutation is accomplished by choosing a new amino acid at random at a given locus. As with the crossover operator, not all selected parent sequences need be subject to mutation, but are rather chosen according to some set probability. Similarly, for chosen individuals, each locus is subject to mutation given some additional (typically very small) probability.

5.2.6 Final Population

Fitness evaluation, selection, crossover and mutation are iteratively performed on successive generations of individuals in the genetic algorithm run, as illustrated in

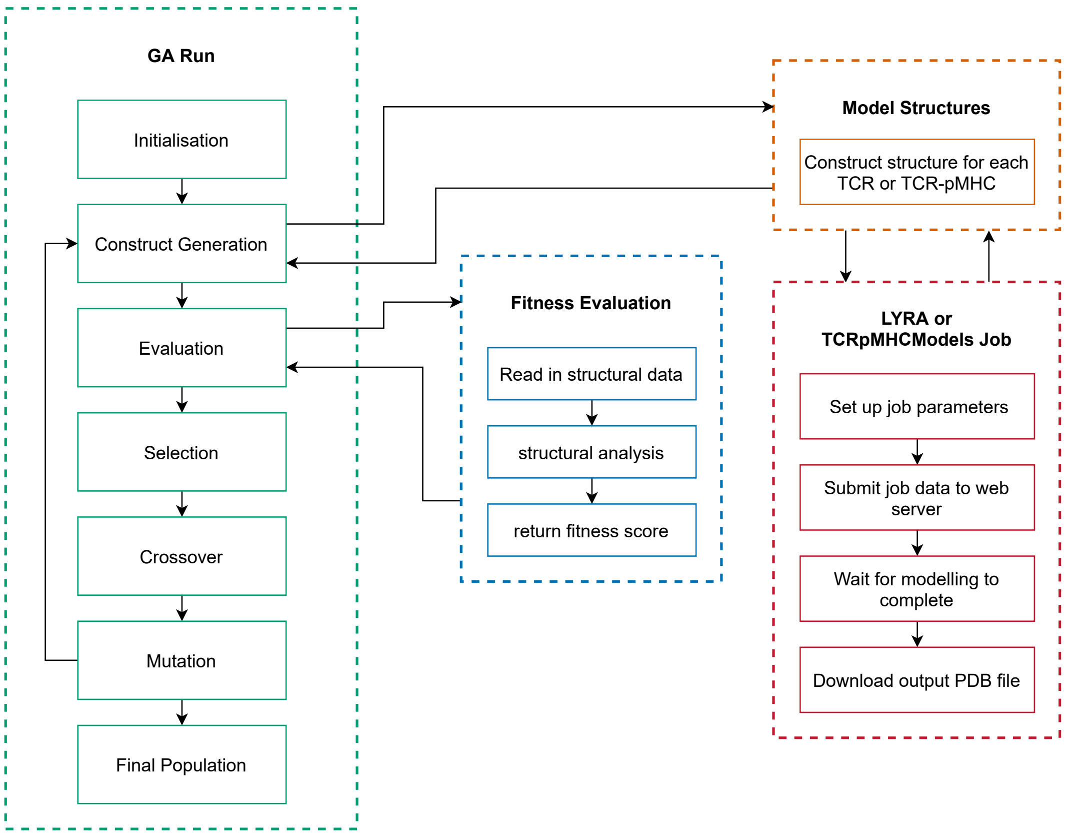


Figure 5.2: A schematic of the main components of the genetic algorithm pipeline for TCR design.

Figure 5.1A. A genetic algorithm is usually terminated after a desired fitness score has been achieved by an individual in the population, a maximum number of generations has been constructed, or maximum wall-clock time has been reached. The final population of individuals should, on average, be fitter than the initial population.

5.3 GA Pipeline for TCR Structural Optimisation

An overview of the main components of the GA pipeline for TCR structural optimisation is shown in Figure 5.2, and each step is detailed below.

5.3.1 Initialisation

At the start of a new optimisation run, the pipeline constructs a tree structure for appropriately storing output data. A results directory is created with a user-provided name. Within this directory, subdirectories are created for storing PDB structures of

TCRs or TCR-pMHC complexes and for storing plots. Additionally, a tab-separated database file is created to keep a record of every individual produced by the run. The database contains the ID and generation of a given individual, the ID of the parent of the individual (if it has one), the fitness score, the sequences of each chain, whether crossover has occurred, which mutations have taken place, the ID of the original templates (if used to construct the TCR), and paths to relevant structure files and snapshot images.

The population size, maximum number of generations, crossover probability, mutation probability, and base mutation probability (for individuals selected for mutation), should be chosen by the user. Initial TCR sequences can be chosen by providing the ID of TCR α and β chain template sequences (stored in a pre-built `templates` directory) as input arguments. All TCRs in the initial population will be seeded with these sequences. Alternatively, TCR sequences can be selected at random from the templates. In the case of optimising TCR-pMHC complexes, the peptide and MHC sequences must be specified manually.

5.3.2 Construct Generation

5.3.2.1 Modelling Structure from Sequence

An initial generation of TCR or TCR-pMHC individuals is constructed using the parameter information and specified sequences. Each individual in the generation then has its structure modelled from sequence using external homology modelling tools.

Subsequent generations are constructed in a similar manner, using the initial parameter information, the TCR sequences of offspring individuals from the selection, crossover and mutation operators, and structures modelled from the evolved sequences.

5.3.2.2 LYRA and TCRpMHCModels

For runs involving the optimisation of only TCRs, the LYRA homology modelling platform (Klausen *et al.*, 2015) is used to generate structures from α and β chain sequences. For runs involving the optimisation of TCR-pMHC complexes, the

TCRpMHCModels homology modelling platform (Jensen *et al.*, 2019) is used to generate structures from α chain, β chain, peptide chain and MHC chain sequences. The TCRpMHCModels software has been designed for only MHC Class I complexes, and this therefore extends to the current implementation of the GA pipeline.

The two processes are implemented in a similar way. For TCR structures, a POST request containing the α and β chain sequences is submitted to the LYRA modelling web server. For TCR-pMHC structures, the sequence information is first assembled into FASTA format, and then submitted as a POST request to the TCRpMHCModels web server.

Once every individual in the generation has been submitted for modelling, the pipeline waits until a structure has been generated for each individual, and then downloads them in PDB format. If the structure of an individual fails to generate, then the individual is replaced by its parent individual.

5.3.3 Evaluation

Each structure is evaluated and assigned a fitness score. The fitness function is imported as a separate module into the core protocol, so can be easily swapped for alternative custom-written functions.

5.3.3.1 Fitness functions

For unbound TCR structures, the current implementation parses the PDB format of each individual and then computes the distances between atoms of the α chain and those of the β chain. The fitness score is calculated as,

$$f = \sum_{0}^{j} \sum_{0}^{i} \begin{cases} (c - d_{a_i b_j})^2 & \text{if } d < c \\ 0 & \text{if } d \geq c \end{cases} \tag{5.1}$$

where $d_{a_i b_j}$ corresponds to the Euclidean distance between the i-th atom of chain a and the j-th atom of chain b, and c is a threshold representing the minimum distance required for each pair of atom to be considered interacting. The α and β chains are set as a and b.

The closer the two chains are to one another, the higher the fitness score. The function has been designed such the shorter the distance between a given pair of atoms, the more this distance contributes to the fitness score. The network of interfacial contacts has been shown as a reasonable predictor of binding affinity across a diverse set of proteins (Vangone and Bonvin, 2015). Suggestions for more sophisticated fitness functions are discussed in Section 5.6.2.

A similar approach has been implemented for TCR-pMHC complexes. Using Equation 5.1, the TCR α and β chains are here set as a and the peptide and MHC chain are set as b.

5.3.4 Selection

Each generation of individuals is subject to tournament selection. Individuals with higher fitness scores are more likely be chosen to be parents for the next generation, but individuals with low fitness scores are still provided a chance of selection to maintain diversity in the population and avoid fixation.

During selection, n tournaments are constructed between x randomly chosen individuals. The individual with the highest fitness score for a given tournament is selected as a parent. n is taken as the number of individuals in the generation, such that the offspring generation will be the same size as the parent generation.

5.3.5 Crossover

A selected individual is subject to crossover with another selected individual with probability `cxpb`. The current implementation swaps the TCR α chain sequence between the two individuals to produce two offspring which are retained as part of the next generation.

5.3.6 Mutation

A selected individual is subject to mutation with probability `mutpb`. If selected for mutation, each amino acid in the α and β chain sequences of the individual is subject to being mutated with probability `mpb`. An amino acid selected for mutation may be substituted for another random amino acid, deleted from the sequence, or have a random amino acid inserted after it in the sequence.

Parameter	Value
c (Å)	7.0
x	3
n	40
g	61
`cxpb`	0.05
`mutpb`	0.20
`mpb`	0.01

Table 5.1: TCR $\alpha\beta$ interfacial contact optimisation parameters: c, the distance cut-off in the fitness function; x, the tournament size; n, the population size; g, the total number of generations; `cxpb` the crossover probability; `mutpb`, the mutation probability; and `mpb`, the per-base mutation probability.

5.3.7 Final Population

The genetic algorithm pipeline runs for a set number of generations. The final population is equal in size to the initial population but should contain individuals with higher fitness scores, provided that fixation has not been reached early in the run.

5.4 Example application: increasing quantity of TCR $\alpha\beta$ interfacial contacts

To demonstrate the functionality of the software, the GA pipeline was applied to an unbound TCR and tasked with increasing the quantity of interfacial contacts between the α and β chains over a number of generations. The BM3.3 TCR (from structure with PDB code 1FO0) was selected for this experiment at random from the set of publicly available resolved crystal structures. An initial population of 40 individuals was set to evolve over an additional 60 generations. The fitness function described by Equation 5.1 was implemented using only the Cα atoms of the α and β chains. The distance cut-off, c, was set to 7 Å, such that any pair of residues with a distance between them lower than this value would contribute to the fitness score.

Selected individuals were given a 20% chance of being subjected to mutation.

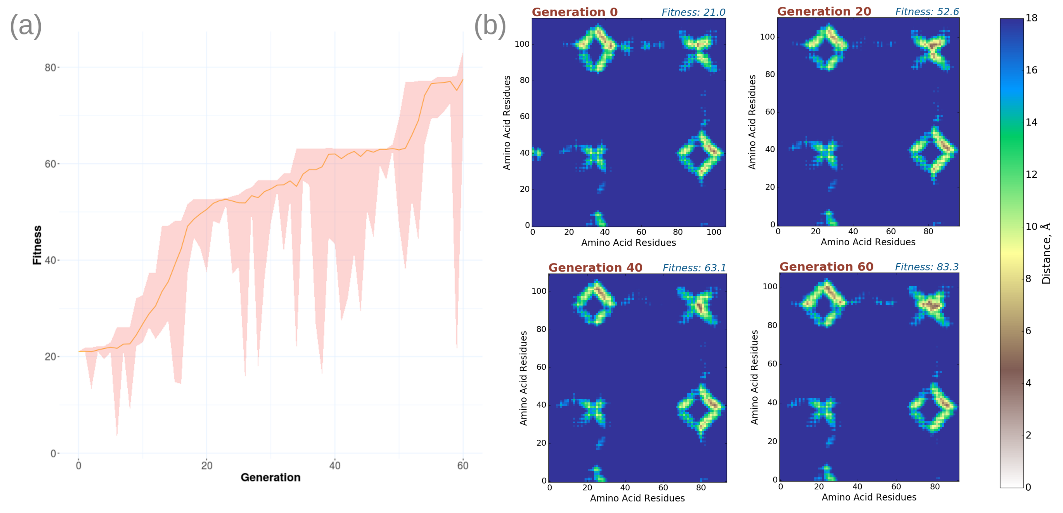


Figure 5.3: (a) The average fitness of the TCRs of each generation is shown, with ribbon lines denoting the maximum and minimum fitness scores achieved in each generation. The average fitness increases over the generations. (b) Example contact maps of the fittest individual in generations 0, 20, 40 and 60. For each map, the amino acids of the TCR α chain extend along the y-axis and those of the β chain along the x-axis. The colour bar denotes the pairwise distance between a given α chain Cα atom and a given β chain Cα atom (up to a maximum distance of 18Å). A subtle increase in the quantity of shorter distances is noticeable across the generations.

For these individuals, each amino acid in its sequence was given a 1% chance of being substituted for another random amino acid, of being deleted from the sequence, or of being followed in the sequence by a random inserted amino acid, each with equal probability. The consequences of these settings are discussed further in Section 5.6.4. A full parameter set can be found in Table 5.1.

Figure 5.3A shows the average fitness of the population per generation. The maximum and minimum fitness scores of each generation are indicated by the ribbon lines. This plot is fairly typical of genetic algorithm optimisation, whereby regions of rapid increase in fitness are interspersed with flat regions of local fixation during which the algorithm explores new solutions. The very last generation shows a small spike in maximum fitness, suggesting that perhaps higher fitness scores might be achieved if the run had been allowed to evolve over more generations.

A notable feature of this plot is the dramatic downwards spike in the minimum fitness for many generations. These features are the result of a poorly chosen crossover operator for the run — the swapping of entire α and β chains between individuals. As the aim of the run is to decrease the interfacial distance between the two chains, the swapping of chains tends to eliminate any optimisation that has occurred specific to a given $\alpha\beta$ pairing. Consequently, the optimisation of this run can be largely attributed to the mutation operator. A more sophisticated crossover operator might seek to swap only specific sequence regions between individuals, rather than entire chains.

Figure 5.3B shows contact map plots for the modelled structures of the fittest individuals of four sampled generations. These heatmaps represent the pairwise distance between the Cα atom of every amino acid in the α chain and the Cα atom of every amino acid in the β chain. For each plot, the α chain sequence extends along the y-axis, and the β chain sequence along the x-axis. The colour of each pixel represents the distance between a given α chain residue and a β chain residue. A cut-off has been applied to the colour bar such that any pairwise distance greater than 18Å is displayed as 18Å. Over the generations, as fitness increases, an increase in the number of residues with pairwise distance of less than 7Å (the fitness function distance cut-off) can be seen in the heatmaps, indicating that the α and β chains in the optimised models are closer to one another than in the initial structure. The limitations of this study and suggestions for future improvements are discussed in Section 5.6.

5.5 Example application: increasing TCR-pMHC interfacial contacts

The GA pipeline may also be applied to TCR-pMHC structures. A GA run was tasked with increasing the quantity of interfacial contacts between the α and β chains of the DM5 TCR (from the structure with PDB code 3QEU), and the pMHC chains of the nonameric MART-1 peptide complexed with the HLA-A2 MHC Class I α chain (from the structure with PDB code 2GUO). An initial population of 30 individuals was set to evolve over an additional 50 generations. The fitness function described by Equation 5.1 was implemented using all atoms of the TCR and pMHC

Parameter	Value
c (Å)	7.0
x	3
n	30
g	51
`cxpb`	0.05
`mutpb`	0.20
`mpb`	0.1

Table 5.2: TCR-pMHC interfacial contact optimisation parameters: c, the distance cut-off in the fitness function; x, the tournament size; n, the population size; g, the total number of generations; `cxpb` the crossover probability; `mutpb`, the mutation probability; and `mpb`, the per-base mutation probability.

chains. As in Section 5.4, the distance cut-off, c, was set to 7 Å. A full parameter set can be found in Table 5.2. Mutation of amino acids was limited to substitution only, with deletion and insertion of residues disallowed. Additionally, mutations were only permitted within the CDR regions (plus two additional residues at each end of each region) of the TCR α and β chains.

Figure 5.4 shows the average fitness of the population per generation. The maximum and minimum fitnesses of each generation are indicated by the ribbon lines. As in Section 5.4, the GA exhibits generations of rapid optimisation interspersed with generations of local fixation. Small spikes in maximum fitness in the final few generations suggest that higher fitness scores might be achieved if the system were allowed to evolve over additional generations.

The ribbon bands on this plot are fairly broad. This is likely to be a feature of a too high per-residue mutation rate, which produces a wide array of individuals with multiple mutations in each generation. An excessively high mutation rate can also have a negative impact on optimisation speed, as advantageous mutations can be more easily lost in a sea of disadvantageous mutations. Nevertheless, a modest increase in fitness can be seen over the generations of the run, indicating that the most fit structural models feature more binding interface contacts than the initial structures.

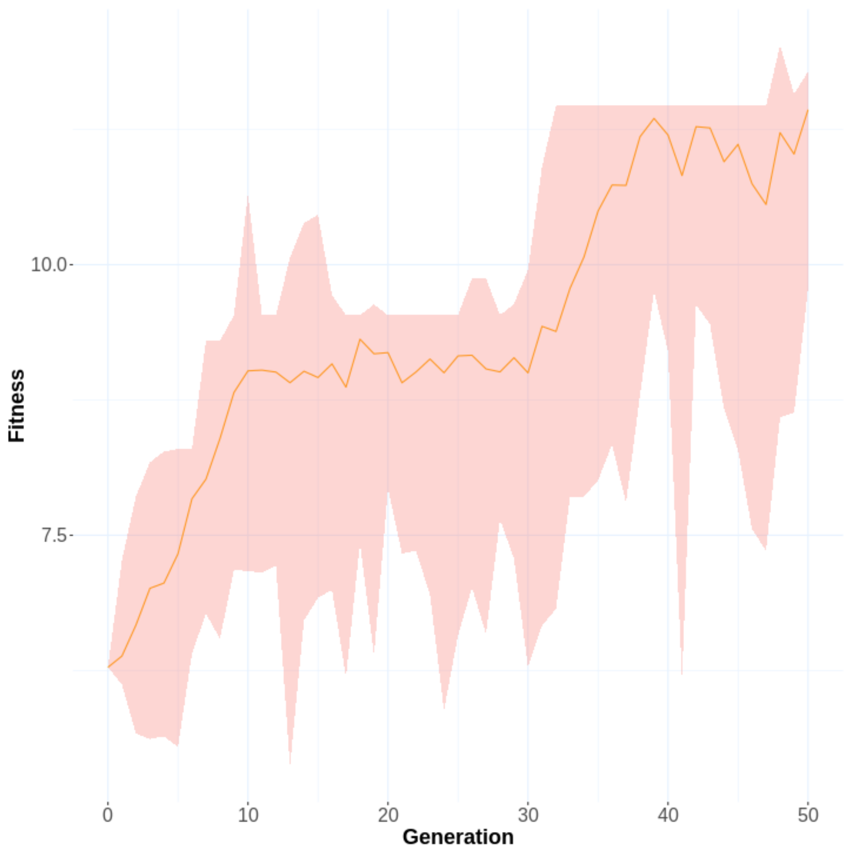


Figure 5.4: The average fitness of the TCR-pMHC complexes is shown to increase across the generation of the GA run. The ribbon lines denote the maximum and minimum fitness scores achieved by each generation.

Figure 5.5 shows a comparison of the starting TCR-pMHC model to that of the fittest TCR-pMHC of the GA run (found in generation 48). Residues that feature atoms within 4.5Å of (and deemed to be in contact with) the binding partner are shown in stick representation. These atoms are highlighted in blue and red for the initial and fittest TCR structures, and in green and orange for the initial and fittest pMHC structures, respectively. The complexes are very similar in their docking mode, but the composition of contacting atoms is noticeably different. Subsequent panels in this plot show the TCR and pMHC components rotated such that the binding interface region points out from the page and with the surface displayed, aiding visibility of the highlighted atoms. The initial TCR-pMHC structure features a binding interface of 82 TCR atoms (from 22 residues) and 95 pMHC atoms (from 24 residues). The fittest TCR-pMHC structure features a binding interface of 125 TCR

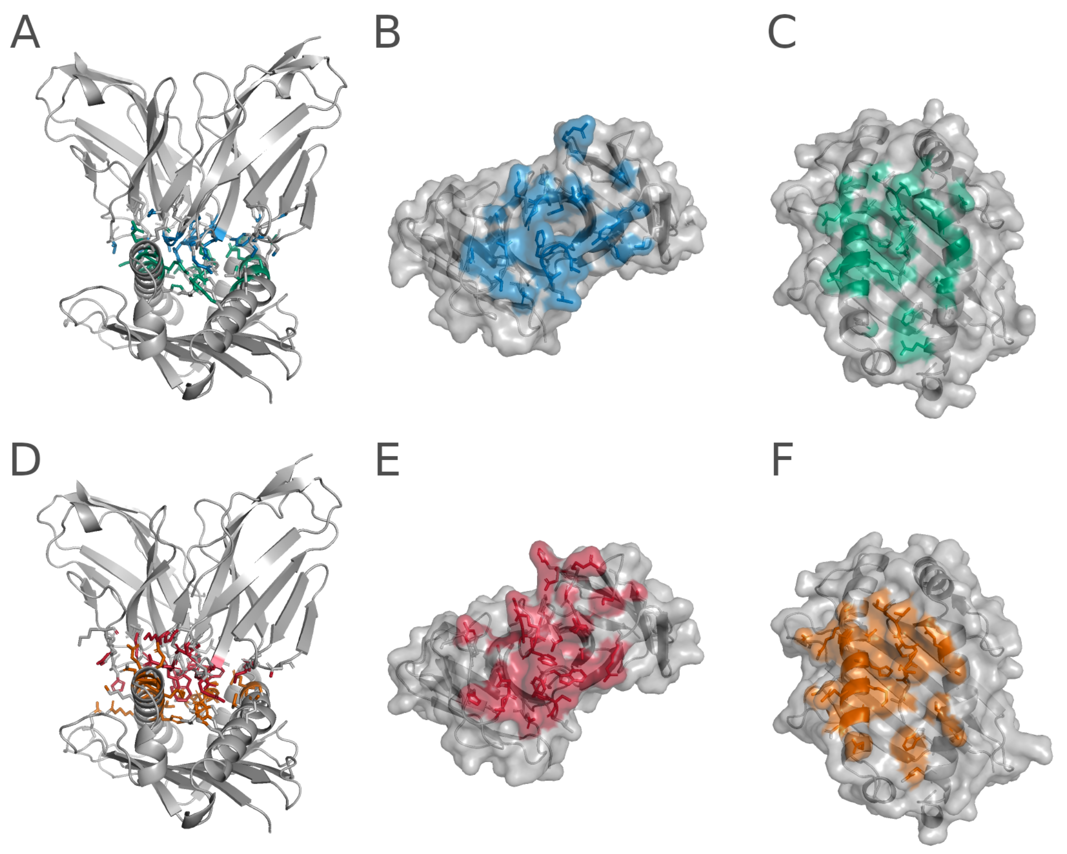


Figure 5.5: (A) The TCR-pMHC complex model used to initiate the GA run. Residues that feature atoms in contact with the binding partner are shown in stick representation. These atoms are coloured blue and green for the TCR and pMHC components respectively. (B) The TCR component and (C) the pMHC component of the initial model are shown, rotated such that the binding interface points out of the page, and with the surface displayed to aid visibility of the highlighted atoms. (D) The highest scoring TCR-pMHC complex model in the GA run. Residues that feature atoms in contact with the binding partner are shown in stick representation. These atoms are coloured red and orange for the TCR and pMHC components respectively. (E) The TCR component and (F) the pMHC component of the fittest model are shown, rotated and with the surface displayed.

atoms (from 24 residues) and 124 pMHC atoms (from 26 residues). The limitations of this study and suggestions for future improvements are discussed in Section 5.6.

5.6 Limitations and Future Perspectives

This chapter has described the principles of a prototype computational pipeline for TCR design through a genetic algorithm optimisation routine. A simple fitness function has been constructed to maximise the number of interfacial contacts between TCR α and β chains, or between TCR and pMHC chains, illustrating how desirable features can be automatically engineered into structural models. This work serves as a preliminary investigation into how advances in structural modelling may be employed for future automated design projects. The current implementation is not without its limitations, a number of which are discussed below.

5.6.1 Limitations of Structural Modelling

An individual is assigned a fitness score based on the analysis of a structure modelled from its sequence. The fitness evaluation is the basis upon which the population is optimised. The challenges of protein structure prediction have been discussed throughout this thesis and in many other works. While it is possible to generate high quality models with existing prediction tools in some circumstances, modelling inaccuracy continues to permeate the discipline.

Consequently, we must assume that a reasonable number of the structural models generated and selected by the genetic algorithm pipeline may lack sufficient accuracy to be considered fitter than their predecessors. Until technology has developed to a point at which we can consider protein structure and protein-protein complex prediction a solved problem, tools that make use of structural modelling for rational design will themselves contain inherent inaccuracies.

The structural biology community continues to innovate. The TCRpMHC-Models software had not yet been published when the development of the pipeline described in this chapter began. Now it has been implemented as a tool for iteratively modelling generations of TCR-pMHC complexes. Similarly, the LYRA platform for TCR modelling was the only automated tool for TCR modelling (Klausen *et al.*, 2015). More recently, the TCRModel (Gowthaman and Pierce, 2018) platform has been published as an alternative tool for TCR homology modelling. It has been shown to exhibit higher accuracy than LYRA for certain cases, and has been released with an additional loop refinement protocol which further improves model accuracy (at the cost of increased run time). Furthermore, a recently published deep learning approach has made such dramatic improvements in structural modelling accuracy (Jumper *et al.*, 2021) that the folding problem for single chain proteins is considered to be solved by some members of the community (AlQuraishi, 2021). It is worth acknowledging, however, that this approach has not yet proven particularly well-suited to modelling antibody and TCR structures (Evans *et al.*, 2021; Yin *et al.*, 2021; Rollins *et al.*, 2022).

In Chapter 4, four general-purpose platforms have been compared in their modelling of TCR-pMHC complexes. Whether these docking platforms would prove more effective for TCR-pMHC modelling than TCRpMHCModels would be worthwhile investigating as an extension of this project. If a significant increase in accuracy were to be demonstrated, a docking-based approach could be implemented into the pipeline as an alternative method for TCR-pMHC modelling. Without access to very large grid computing, it is likely that the incorporation into the pipeline of a long-running and intensive software such as HADDOCK would prove infeasible. Even if entire generations could be processed in parallel, GA runs would likely extend into multiple weeks of run time. Implementation of faster platforms such as ZDOCK and ClusPro serves as a more achievable goal.

It is worth noting that if docking tools were implemented as an alternative approach to homology modelling, the unbound TCR and pMHC components of each individual would themselves require modelling from sequence. Docking platforms are less successful when assembling modelled unbound structures, which feature their own inaccuracies, into complexes. Both homology modelling and docking approaches additionally suffer from their inability to capture the dynamics of proteins over time, particularly those of flexible loop regions.

In the short term, the accuracy of structural modelling approaches form a major limitation of the optimisation pipeline. However, the scientific community continues to make steady progress in modelling accuracy. The availability of new cutting-edge technologies with high accuracies and fast modelling times will no doubt increase the viability of approaches such as that described in this chapter in the future.

5.6.2 Alternative Scoring Functions

The current implementation of the scoring function aims to maximise the number of interfacial contacts between either TCR α and β chains or between TCR chains and pMHC chains. The function has been design to reward a range of achieved distances between atoms within a set distance threshold. Shorter achieved distances contribute more to the fitness score than longer achieved distances.

The scoring function is based on an assumption that a greater number of interfacial contacts reflects a higher binding affinity. This has been shown by Vangone and Bonvin (2015) in a study which demonstrates that additionally incorporating the physico-chemical properties of the residues, and contributions of the non-interacting surface (Kastritis *et al.*, 2014), provides a yet stronger predictor of binding affinity. These features could be incorporated into a more sophisticated scoring function for the GA optimisation.

Furthermore, the binding affinity may not be the only property of interest for novel TCR design. Section 1.4 has discussed geometric descriptions that have been developed for both TCRs and TCR-pMHC complexes. Specific angles and distances could be encoded into a new fitness function to optimise TCR models towards desirable geometries. Alternatively, a fitness function could be designed to search models for specific charge pairings or for desirable interaction types in the structural models (such as hydrogen bonds or salt bridges), or to tune hydrophobicity.

5.6.3 Optimal Parameter Sets and Computational Resources

The use of an evolutionary algorithm involves choosing values for a number of parameters (the population size, mutation and crossover rates, and so on). It can be difficult to know what the best values are for these parameter when setting up a new run. Section 5.5, for example, has highlighted the detrimental effect on run time produces by a too high mutation rate.

The parameters for the runs shown in this chapter were chosen by experimental trial and error. A continuation of this project would benefit from a more sophisticated approach. An automated parameter sweep employed to systematically adjust appropriate parameters could be conducted as a simple (yet potentially very lengthy) approach to parameter tuning. Optimal tuning of evolutionary algorithm parameters has been the subject of numerous studies (DeJong, 2007; Smit and Eiben, 2009;

Sipper *et al.*, 2018), and is an area that would be worthwhile exploring to improve the pipeline.

The runs described in this chapter were performed relatively quickly in terms of computational run time (< 48 hours per run), using limited population sizes and numbers of generations. Increasing the population size would require additional parallelisation efforts, likely involving the installation of the structural modelling software (LYRA and TCRpMHCModels) on a dedicated high performance computing server, rather than relying on external web servers, to model all individuals from a single generation simultaneously. It is harder to avoid an increase in run time if increasing the number of generations in a run. However, so long as a single generation can be modelled quickly, there is scope to increase this value beyond the settings used in this chapter.

5.6.4 Alternative Crossover and Mutation Operators

5.6.4.1 Crossover

The results shown in Section 5.4 have highlighted how the current simplistic implementation of the crossover operator, whereby α and β chains are switched between individuals, has had a negative impact on the optimisation of contacts between the two TCR chains. Moreover, the optimisation is largely driven by the mutation operator for this run.

The chain swapping crossover operator is less detrimental to evolution in runs for TCR-pMHC optimisation, as shown in Section 5.5. This can be attributed to the two TCR chains largely optimising independently for contact with the pMHC, rather than for contact with one another.

It may be worthwhile exploring more sophisticated crossover operators as an extension to this project, particularly for runs optimising contacts between TCR α and β chains. "Hotspot" crossover, for example, permits the swapping of only certain regions of the encoded chromosomes. This could be used to limit crossover to only specific regions of the TCR sequences deemed unlikely to excessively disrupt the interface.

5.6.4.2 Mutation

Section 5.4 and Section 5.5 make use of two similar, but distinct, approaches to mutation. For the TCR modelling, substitution, insertion and deletion of amino acids were all permitted. For TCR-pMHC modelling, only substitution was permitted. In the former case, addition and deletion introduces additional diversity into the population. In the latter case, the mutation was restricted to reduce the number of cases where the structural modelling platform would fail to generate a model. The platform could be improved by introducing greater flexibility to these features. Addition and deletion could be weighted with a lower probability of occurring than substitution. Maximum and minimum thresholds could be imposed on sequence and CDR loop lengths. The different types of mutation could be restricted to only certain regions of the sequence.

The current implementation of where a mutation can take place in a sequence is fairly naïve. In Section 5.4, any residue was permitted to undergo mutation. In Section 5.5, the pipeline was upgraded to support mutation only in the CDR regions. The CDR loops are the primary regions of the TCR that contact the pMHC, and variation in these regions drives most of the diversity in the population. Currently, any mutating amino acid can be substituted for any other random amino acid. A more sophisticated approach would allow for greater customisability. The likelihood of a residue mutating could be weighted by its position in the sequence. Functionality could be introduced to allow certain key residues to be blocked from mutating entirely. Mutation could also be restricted based on physico-chemical similarity, such that substitution is permitted only according to observed residue distributions.

5.6.5 Affinity, Specificity, and Activity

The optimisation routines outlined in this chapter have aimed to maximise the number of interfacial contacts between TCR and TCR-pMHC chains as a surrogate for binding affinity. The engineering of TCRs with arbitrarily high binding affinities must be approached with a great degree of caution. Due to the cross-reactive nature of TCRs, enhancement of TCR affinity poses a risk of reduced specificity, leading to unexpected side effects (Pierce *et al.*, 2014a). These side effects can lead to tragic consequences in clinical settings (Linette *et al.*, 2013). Designed TCRs, particularly those with high affinity, must therefore be carefully evaluated to ensure that their specificity is maintained. The role that structural biology can play in this evaluation has been recently reviewed by Singh *et al.* (2017).

An additional side effect of TCRs with very high binding affinities is a noted reduction in T cell activity beyond a given "affinity threshold" (McKeithan, 1995; Rabinowitz *et al.*, 1996; Carreño *et al.*, 2007). This threshold is consistent with the "half-life" model of TCR activation, which proposes that a TCR must bind an antigen with a strength and duration sufficient for productive signalling, and must additionally bind for a time short enough that adjacent TCRs may bind the same antigen in a given window for signal amplification (Zoete *et al.*, 2013). A higher binding affinity increases the TCR dwell-time, and may inhibit serial engagement, thereby attenuating T cell signalling. For the GA pipeline, this suggests that it may be beneficial to constrain the optimisation of the binding affinity, to preserve T cell activity.

5.6.6 Experimental Validation

A complete assessment of the successes of this pipeline would require experimental validation, whereby the fittest TCRs discovered by the algorithm could have their properties measured in the laboratory. The preliminary nature of this research has meant that wet lab work has not yet been feasible, but would make for an exciting and hopefully illuminating continuation of this project.

5.7 Tools

The genetic algorithm implementation was constructed using the DEAP evolutionary computation framework (Fortin *et al.*, 2012).

Chapter 6

Conclusions

Our understanding of the complexities of the immune system and our ability to guide it towards more effective elimination of invasive pathogens continues to blossom. T cell activity is a critical feature of the adaptive immune response. Breakthroughs in sequencing technology have allowed us to probe the TCR repertoire in great depth. As we aim to convert discoveries into immunologically driven medicines, it is important to also understand TCR interaction at a structural level. This thesis has explored methods and tools for analysing TCRs at various scales. At the population level, methods for TCR repertoire analysis have been reported. At the molecular level, methods for structural modelling of TCR-pMHC complexes and automated structure-guided design have been explored.

In particular, this thesis has described the development of the latest version of the Decombinator platform, Decombinator V4, for TCR sequencing analysis. The efficient pattern-matching algorithm that forms the basis of its TCR annotation allows Decombinator to achieve shorter run times than most other platforms. New sequencing error and PCR amplification bias correction routines have improved Decombinator annotation accuracy, and the implementation of a unique dual index protocol has mitigated the effects of artefactual sample mixing through index hopping. As TCR repertoire sequencing and analysis have matured to become routinely used methods for the study of adaptive immunity, a need has arisen for standardised formats for the management, sharing and compatability of datasets and computational tools. Accordingly, Decombinator V4 has been upgraded to comply with the international standards proposed by the AIRR community.

The inability to pair α and β chain sequences in historical TCR repertoire analyses has hindered our understanding of the immune response *in vivo*. The emergence of affordable and reliable single cell sequencing has provided the immunological community with the means to overcome this challenge for the first time. Traditional repertoire analysis tools are not necessarily suitable for the short, fragmented reads produced by single cell sequencing. An adaption of Decombinator that searches for single V or J genes in single cell reads and for overlap between these reads to reconstruct full TCR sequences has been outlined in this thesis, and shows comparable performance to two other single cell repertoire analysis platforms.

A number of ways in which the platform could be extended in the future have also been discussed. An integration of Decombinator and Single Tag Decombinator into a unified pipeline would streamline analysis and enhance useability. At present, this would be likely to constitute a large amount of work. In the long term, however, the requirement of maintaining two similar, but independent, pipelines would be eliminated. A formalised testing framework would aid in development and routine benchmarking of the platform for speed and TCR annotation accuracy for both bulk and single cell protocols. Parallelisation of the pipeline has been experimented with, and could be implemented to improve run time further. Useability could be extended by providing greater flexibility regarding input data formats.

An understanding of protein interaction at the molecular level is critical for the development of new therapeutics. Enormous diversity in the TCR repertoire means that interesting TCRs identified in high-throughput sequencing experiments are very unlikely to have had their structure resolved. Traditional methods for structural resolution of TCRs can be challenging, time-consuming, and out of reach for many research groups. Computational modelling of TCR structures and TCR-pMHC complexes provides an efficient and accessible alternative. The accuracy of the modelled structures produced computationally is of primary concern.

Protein-protein docking algorithms, that aim to assemble unbound protein structures into models that accurately resemble native bound complexes, are often tested against the Protein-Protein Docking Benchmark (Vreven *et al.*, 2015). However, no TCR-pMHC complexes are found among these assembled test cases and most algorithms are not specifically benchmarked in the context of TCRs. An independent set of bound and unbound TCR and pMHC structures has been previously assembled as the TCR docking benchmark (Pierce and Weng, 2013; Gowthaman and Pierce, 2019). In this thesis, research conducted to expand the set of benchmark cases has been described. Details are provided of the preprocessing methods to prepare these structures for use with docking software, including a protocol for repairing missing atoms or residues. Scores have been calculated that describe the expected difficulty of modelling each case, and the structures are made available online in raw and preprocessed form for general use.

Each docking case assembled as part of the expanded TCR benchmark was used to assess the ability of four general-purpose protein-protein docking platforms to accurately model TCR-pMHC complexes. ClusPro, HADDOCK, LightDock and ZDOCK all include features that allow guiding of their docking algorithms with additional information known about the proteins of interest. The performance of each platform was evaluated and compared in the context of varying provided information about the TCR-pMHC binding interface. The performance per docking case has been discussed, with little evidence that “medium” difficulty cases were harder to model than “rigid” (easy) cases. The effects of alternative scoring functions have been discussed in the context of ClusPro; the effects of model filtering in the context of LightDock; and the effects of clustering of models in the context of HADDOCK. HADDOCK was shown to have a notably stronger sampling performance than the other platforms. The ability of the two flexible docking platforms, HADDOCK and LightDock, to model conformational change in the CDR loops was assessed, with neither platform notably improving resultant models. Choosing the most appropriate modelling platform for a given task depends on the amount of available information. In general, HADDOCK was shown to be the most accurate docking software for TCR-pMHC modelling when considering the top ranked model. Clus-

Pro was found to be the most consistent performer outside the top ranked model. Each of the platforms showed the best performance when detailed information about the binding interface (determined from the reference structure) was provided.

Despite some of the successes shown in this thesis, it is clear that there is still progress to be made in the computational docking of TCR-pMHC structures. Discussion has been provided on some of the paths this progress might take, including novel re-ranking and filtering methods, loop region modelling, and in tools to accurately predict binding residues in the CDR loops and pMHC.

As methods for structural modelling continue to improve, there is an opportunity to speculate how they may be integrated into broader automated scientific pipelines. A prototype platform for TCR design using homology modelling and genetic algorithm routines has been outlined in this thesis. The platform has been used to automatically generate TCR and TCR-pMHC models with more interfacial contacts than initial structures. Potential extentions to the software have been discussed, including alternative scoring functions to engineer other features into models, the optimisation of parameters, and improved implementation of genetic operators.

The usefulness of such pipelines depends fundamentally on the accuracy of the structural modelling tools they employ. Discussion has been provided about these limitations and what options might improve accuracy in the short term. However, as the scientific community continues to make progress in structural modelling accuracy, approaches such as those outlined in this thesis stand to increase in viability for protein design.

Chapter 7

Tools

Visualisation

Protein structures are visualised using PyMOL (`http://www.pymol.org`). Plots were designed using Matplotlib (Hunter, 2007), Seaborn (Waskom, 2021) and ggplot2 (Wickham, 2016). Diagrams were designed using diagrams.net (`https://www.diagrams.net/`; formerly draw.io).

Structural Analysis

Structural data was analysed using the Bio.PDB (Hamelryck and Manderick, 2003) Biopython (Cock *et al.*, 2009) module, and the PyMOL (`http://www.pymol.org`) api.

IMGT Renumbering

IMGT renumbering of sequences was performed using ANARCI (Dunbar and Deane, 2016).

Genetic Algorithm Infrastructure

The genetic algorithm implementation was constructed using the DEAP evolutionary computation framework (Fortin *et al.*, 2012).

Appendix A

Metrics for Sequence and Structural Analysis

Equations for Structural Geometric Descriptions

Centre of Mass

The centre of mass (CoM) of a protein is defined by a weighted average of the atomic positions in the protein:

$$\boldsymbol{R} = \frac{1}{M}\sum_{i=1}^{N} m_i \boldsymbol{r}_i \qquad \text{(A.1)}$$

where $\boldsymbol{R} = (x, y, z)$ gives the coordinates of the CoM, M gives the total mass of all the atoms in the system, and N gives the total number of atoms in the system each with mass m_i and coordinates $\boldsymbol{r}_i = (x_i, y_i, z_i)$. Often, hydrogen atoms are omitted from the calculation, to provide the CoM of only the heavy atoms of the protein.

Metrics for Sequence Analysis

Hamming Distance

The Hamming distance is a measure of dissimilarity (edit distance) between two strings of equal length. It is defined as the minimum number of substitutions required to transform one string into the other. Mathematically, the Hamming distance, $d(u, v)$ between two strings, $u = u_1, ..., u_n$ and $v = v_1, ..., v_n$, can be expressed as:

$$d(u,v) = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } u_i \neq v_i \\ 0 & \text{if } u_i = v_i \end{cases} \tag{A.2}$$

For example the Hamming distance between:

"pathogen" and **"patients"** is 5

and the Hamming distance between

"pathogen" and **"pangolin"** is 4.

Levenshtein Distance

The Levenshtein distance is a measure of dissimilarity (edit distance) between two strings that may be of equal or unequal length. This distance is measured as the minimum number of insertions, deletions or substitutions required to transform one string to the other. For example, the Levenshtein distance between:

"pathogen" and **"pattern"** is 4

corresponding to 3 substitutions and 1 deletion. The Levenshtein distance between:

"pathogen" and **"patterns"** is 5

corresponding to 5 substitutions (and equal, in this case, to the Hamming distance). The Levenshtein distance between:

"pathogen" and **"pathogens"** is 1

corresponding to 1 insertion.

Numerous algorithmic approaches have been implemented to efficiently compute the Levenshtein distance (Navarro, 2001; Berger *et al.*, 2021). The Decombinator software package has historically made use of the `python-Levenshtein` package (`https://github.com/ztane/python-Levenshtein`) for Levenshtein distance calculations, however the Collapsinator module now makes use of the more efficient `polyleven` package (`https://github.com/fujimotos/polyleven`).

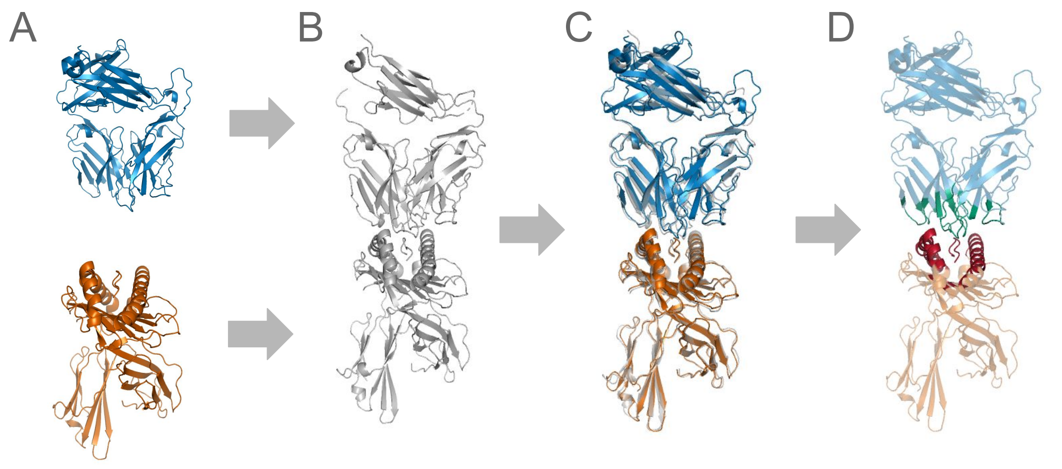


Figure A.1: (A) The unbound TCR (blue) and unbound pMHC (orange) structures for docking case 1AO7. (B) The reference bound TCR-pMHC complex structure (grey). (C) The unbound components are superimposed onto the reference structure. (D) The TCR and pMHC residues that make up the interface for the superimposed unbound structures (highlighted in green and red, respectively) are extracted.

Metrics for Structural Analysis

Root-Mean-Square Deviation (RMSD)

The root-mean-square deviation (RMSD) is a measure of dissimilarity in the geometry of two sets of superimposed atoms. It is calculated as:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2} \tag{A.3}$$

where δ_i is the distance between atom i in the model structure and atom i in the reference structure.

Interface Root-Mean-Square Deviation (IRMSD)

The interface root-mean-square deviation (IRMSD) is a measure of dissimilarity in the geometry of atoms that form the binding interface in two protein complexes. The interface region is of critical importance in protein-protein complex modelling, and IRMSD measurements provide a more accurate assessment of the modelling

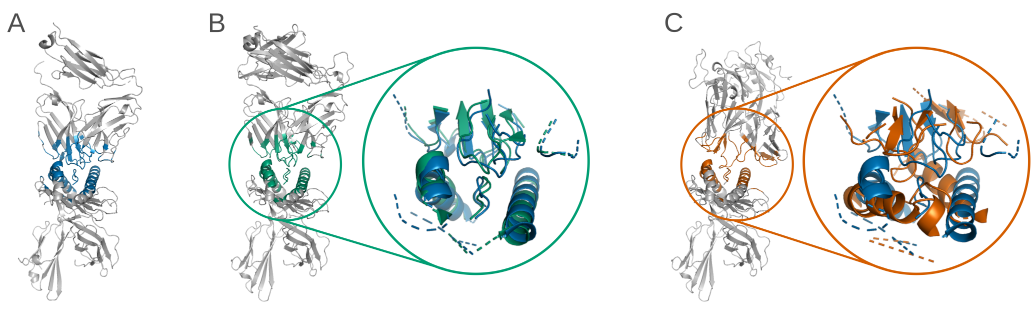


Figure A.2: (A) The reference TCR-pMHC complex for docking case 1AO7, with interface residues highlighted (blue). (B) A docked model with an accurately captured interface (green). Inset, superimposition of the reference and model interfaces, showing good structural alignment. (C) A docked model with a poorly captured interface (orange). Inset, superimposition of the reference and model interfaces, showing poor structural alignment.

than global RMSD measurements (that take into account unimportant regions of the protein) in this context.

The IRMSD has been used as a measure to stratify the difficulty of a given docking test case (Chapter 3). In a protein-protein complex, the interface is composed of all the residues that contain at least one atom within 10Å of an atom in the binding partner (Méndez *et al.*, 2003). To determine the IRMSD, the unbound proteins (here, the TCR and pMHC) are first superimposed on the reference structure (here, the bound TCR-pMHC complex). The interface can then be extracted from the superimposed components. This procedure is shown in Figure A.1. The IRMSD is then found by calculating the RMSD (Equation A.3) between the backbone atoms in the reference interface and the superimposed unbound components interface.

The IRMSD is similarly computed as a measure of evaluating the quality of docked models (Chapter 4). The interface is extracted from the reference structure and from the docked model. The IRMSD is then found by calculating the RMSD between the backbone atoms of these two interfaces. Figure A.2A shows the reference structure for the docking case with PDB code 1AO7, with interface residues highlighted in blue. Figure A.2B shows a model that has accurately captured the binding interface, highlighted in green. The inset shows the model interface superimposed with the reference interface, demonstrating good structural alignment

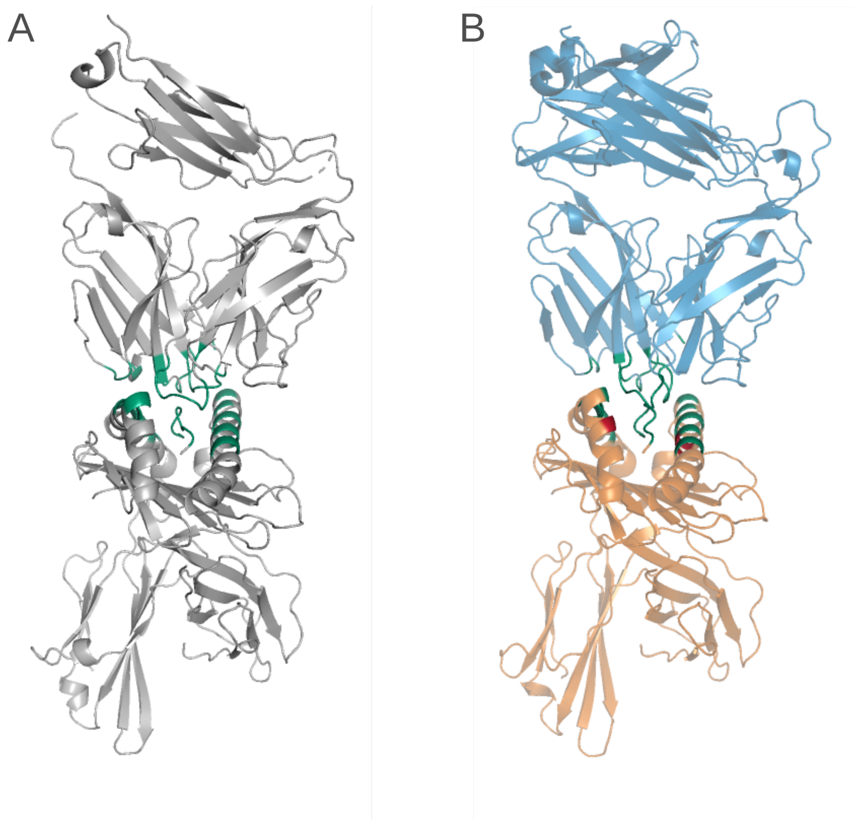


Figure A.3: (A) The reference TCR-pMHC complex for docking case 1AO7 with residues involved in residue-residue contacts highlighted (green). (B) The unbound TCR (blue) and pMHC (orange) structures for docking case 1AO7 arranged as superimposed upon the reference structure. Residues involved in residue-residue contacts that do (green) and do not (red) feature in the reference structure are highlighted.

(IRMSD = 1.63Å). Figure A.2C shows a model that has poorly captured the binding interface, highlighted in orange. The inset shows the model interface superimposed with the reference interface, demonstrating poor structural alignment (IRMSD = 13.00Å).

Fraction of Non-Native Contacts ($F_{\text{non-nat}}$)

The fraction of non-native contacts ($F_{\text{non-nat}}$) is defined as the number of non-native (incorrect) residue-residue contacts in an assembled complex divided by the total number of residue-residue contacts in that complex. A residue-residue contact, in this context, is defined as a pair of residues on either side of the interface that have at least one pair of atoms within 5Å of one another (Méndez *et al.*, 2003).

The $F_{\text{non-nat}}$ has been used as a measure to stratify the difficulty of a given dock-

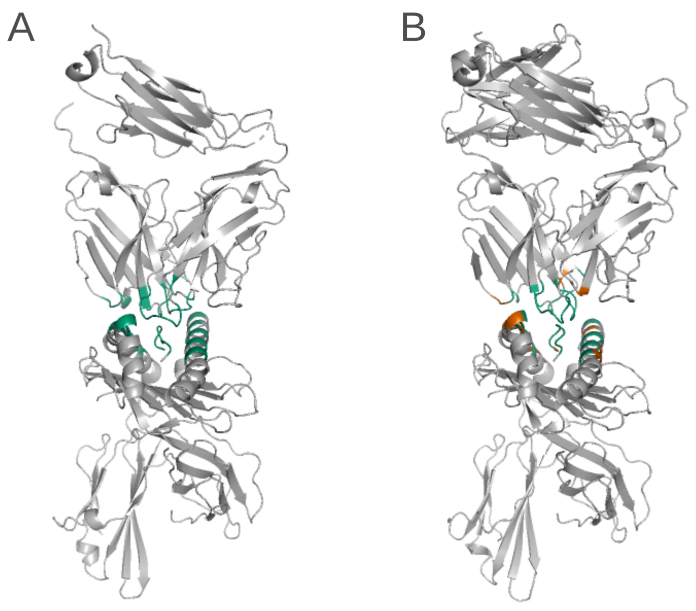


Figure A.4: A) The reference TCR-pMHC complex for docking case 1AO7 with residues involved in residue-residue contacts highlighted (green). (B) A docked model with residues involved in residue-residue contacts highlighted in green if they are found in both the reference and model complexes, and in orange if they are found only in the reference complex.

ing test case (Chapter 3). Figure A.3A shows the reference structure for the docking case 1AO7, with residues involved in residue-residue contacts highlighted in green. Figure A.3B shows the unbound TCR (blue) and pMHC (orange) components of the docking case assembled through their superimposition with the reference structure (as in Figure A.1). Residues involved in residue-residue contacts are highlighted in green if they also feature in the reference model, and in red if they do not. It should be noted that a given residue may form multiple residue-residue contacts, which is not illustrated in Figure A.3. Therefore, the $F_{\text{non-nat}}$ of 0.33 for this docking case is perhaps higher than Figure A.3 might suggest.

Fraction of Native Contacts (F_{nat})

The fraction of native contacts (F_{nat}) has been used as a measure of evaluating the quality of docked models (Chapter 4). The F_{nat} is defined as the number of native (correct) residue-residue contacts found in the docked model divided by the total number of residue-residue contacts in the reference complex (Méndez *et al.*, 2003). A residue-residue contact, in this context, is defined as a pair of residues on either

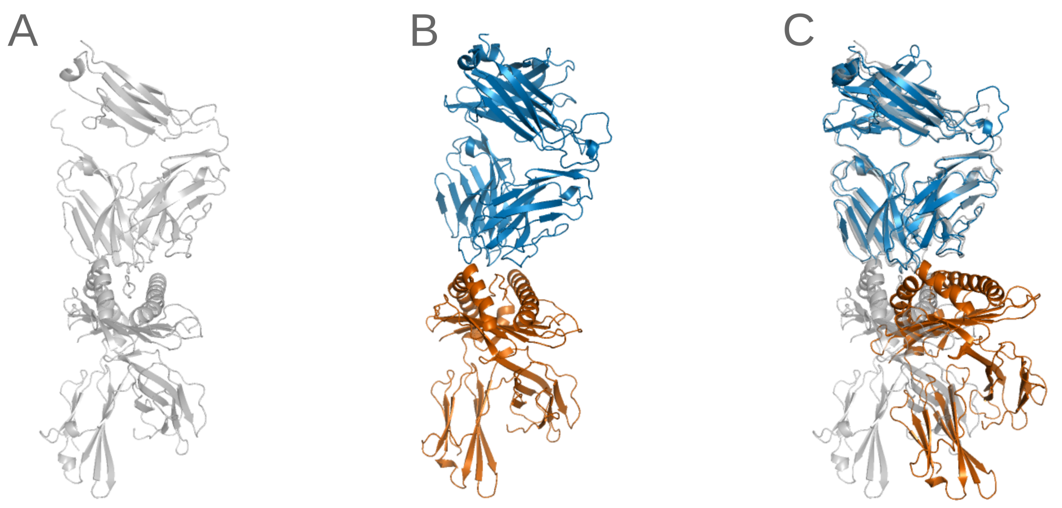


Figure A.5: (A) The reference structure for docking case 1AO7. (B) A docked model with TCR shown in blue and pMHC in orange. (C) The TCR (receptor) of the docked model is superimposed with the TCR of the reference structure, highlighting a large deviation in the position of the pMHC (ligand) component.

side of the interface that have at least one pair of atoms within 5Å of one another.

Figure A.4A shows the reference complex for docking case 1AO7 with residues involved in residue-residue contacts highlighted in green. Figure A.4B shows a docked model with residues involved in residue-residue contacts highlighted in green if they are found in both the reference and model complexes, and in orange if they are found only in the reference complex. It should be noted that a given residue may form multiple residue-residue contacts, which is not illustrated in Figure A.4. The F_{nat} score for this model is 0.49.

Ligand Root-Mean-Square Deviation (LRMSD)

The ligand root-mean-square deviation (LRMSD) is a measure of geometric fit and has been used to evaluate the quality of docked models (Chapter 4). To calculate the LRMSD, the receptor of the modelled complex is superimposed upon the receptor of the reference structure. The RMSD (Equation A.3) is then computed between the backbone atoms of the ligand of the reference structure and the backbone atoms of the ligand of the model structure (Janin *et al.*, 2003; Méndez *et al.*, 2003). In this study, the TCR is taken as the receptor and the pMHC as the ligand.

Figure A.5A shows the reference TCR-pMHC complex for docking case

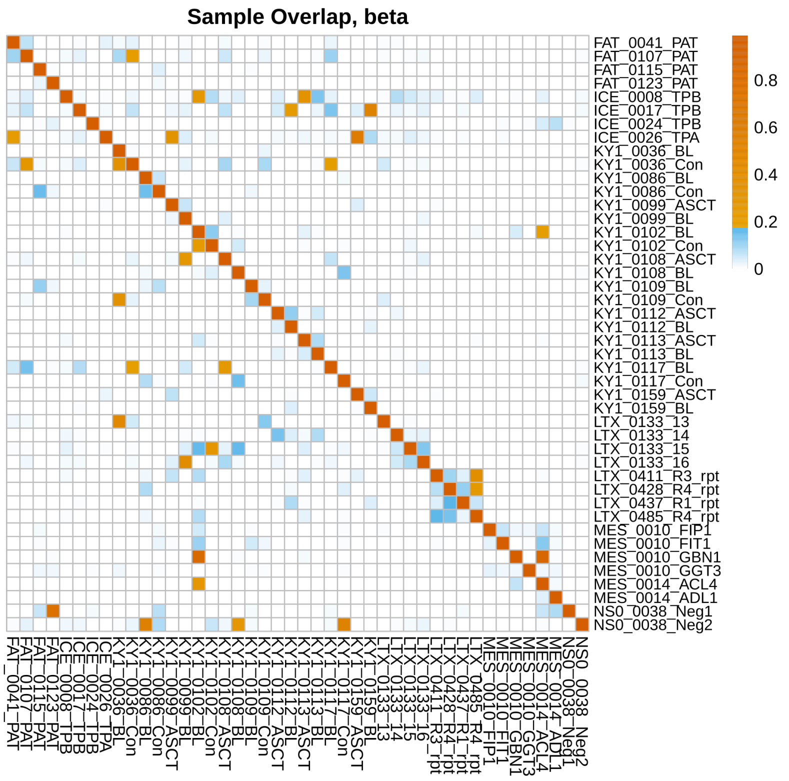
1AO7. Figure A.5B shows a docked TCR-pMHC model for this case, with TCR shown in blue and pMHC in orange. Figure A.5C shows the TCR (receptor) in the model superimposed with the TCR in the reference structure, highlighting a large deviation in the position of the pMHC (ligand) component. In this example, the LRMSD was computed as 20.80Å.

Appendix B

Beta Chain Internal and Unique Dual Index Decombinator Protocols

The following page provides a comparison of the overlap between samples for the β chain data for two NextSeq runs using different protocols. Figure B.1A shows the overlap for a run using the legacy IDI protocol, and Figure B.1B shows the overlap for a run using the new UDI protocol. As with the α chain data shown in Section 2.2.1, the IDI plot shows overlap between disparate patients and experiments, whereas the UDI plot shows only inter-patient overlap.

A



B

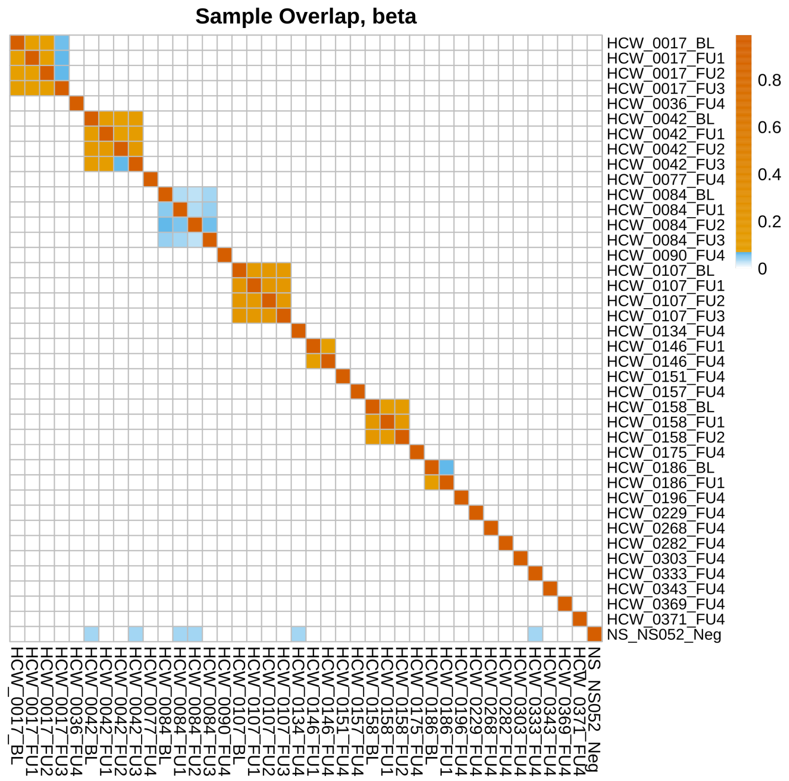


Figure B.1: (A) TCR overlap between patients and experiments for β chain data from an IDI protocol NextSeq run. (B) TCR overlap for β chain data from a UDI protocol NextSeq run. Rows and columns with suffixes "Neg", "Neg1" and"Neg2" provide negative controls.

Appendix C

Single Tag Decombinator Supplementary Data

ID	DCR	ID	DCR
1	4, 45, 0, 8, AGTAC	21	31, 19, 2, 8, AGGG
2	1, 22, 5, 1, GAGGG	22	43, 41, 7, 15, GAACGCCCCCCC
3	35, 17, 2, 0, CCCTGGGG	23	17, 36, 5, 0, TG
4	18, 34, 0, 0, ATGAGG	24	35, 20, 8, 1, GGTT
5	19, 31, 1, 11, ACCG	25	39, 11, 10, 0, CTGTCG
6	31, 32, 6, 11, CGCTTCTGCGG	26	41, 45, 3, 7, GGGCCG
7	39, 49, 4, 5, TCTC	27	41, 45, 9, 7, AGTGAGGGGCCG
8	24, 46, 1, 6, GAGGG	28	38, 9, 7, 6, GCC
9	16, 48, 0, 20, CACAGTGCAACCCAGGCTCCCAGAAC	29	40, 0, 9, 6, CCTG
10	12, 8, 6, 4, GGAGGG	30	14, 57, 0, 8, ACCTTGTG
11	32, 49, 8, 4, TTATT	31	8, 32, 1, 0, TTT
12	43, 41, 7, 15, GAAAGCCCCCCCCTAGCTCTCGAAAGCCCCCCC	32	7, 18, 3, 13, GATTAAGG
13	31, 23, 2, 16, GATTCTCG	33	31, 38, 5, 1, T
14	43, 41, 7, 15, GAAAGCCCCCCC	34	41, 9, 8, 7, ATGGG
15	43, 41, 7, 15, GAAATCCCCCCCCTAGCTCTCGAAAGCCCCCCC	35	4, 10, 1, 4, GAAG
16	23, 41, 0, 10, GAGA	36	44, 19, 0, 18, TGACCTTCCTATTATTTGGGGGCCTTACACCGATAAAC
17	21, 29, 0, 4, CACG	37	21, 40, 6, 3, CTGGTGG
18	20, 21, 12, 11, CCATGGGG	38	12, 45, 6, 7, ACTGGGATC
19	17, 1, 6, 1, TTCTCGGTGG	39	35, 16, 4, 6, G
20	39, 29, 0, 3, TCCG		

Table C.1: Mapping between representative IDs and α chain Decombinator classifiers (DCRs) used in Figure 2.16 for the NSCLC single cell analysis comparison of Single Tag Decombinator and TraCeR.

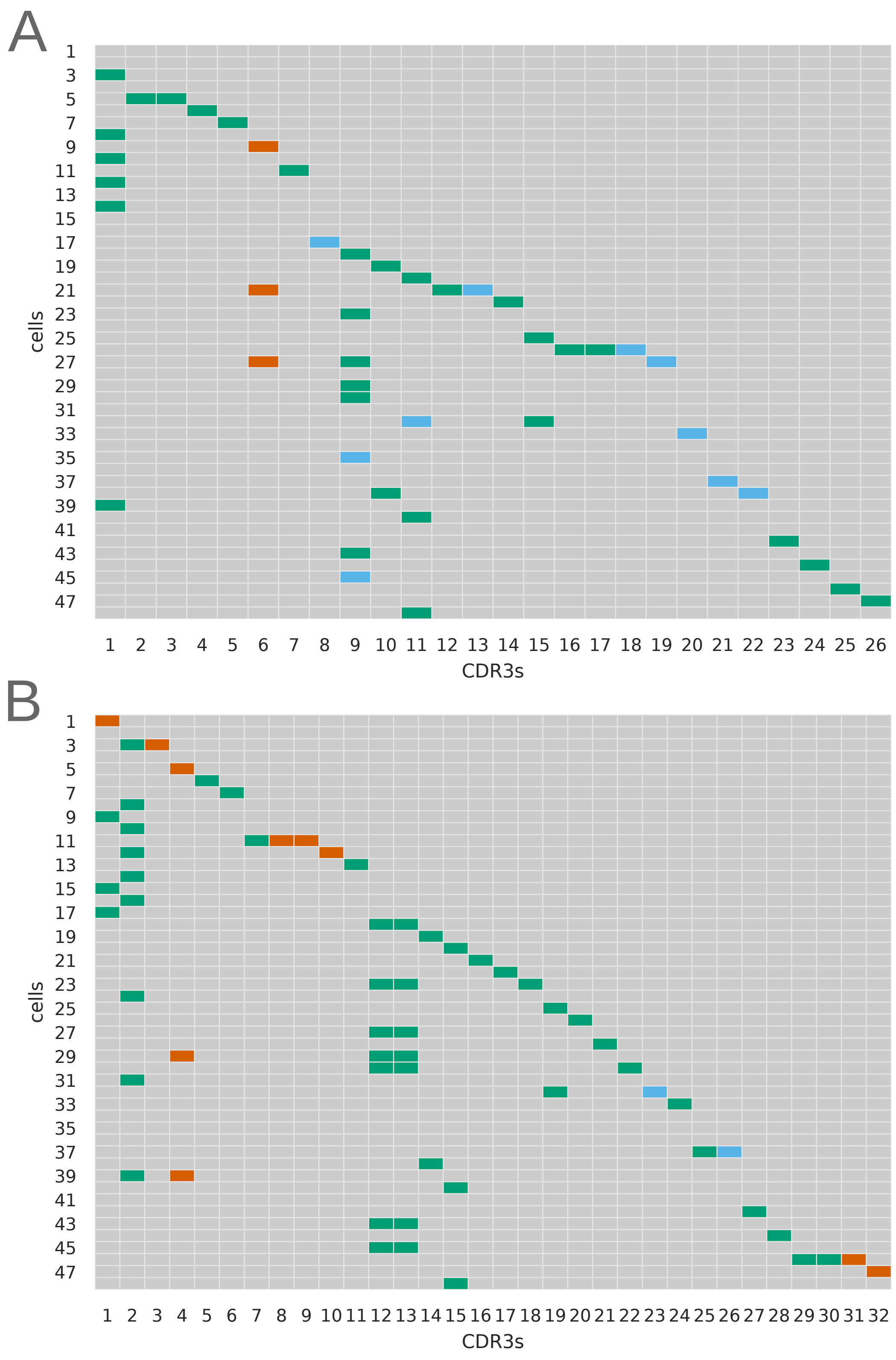


Figure C.1: TCR (A) α chains and (B) β chains identified in the SARS-CoV-2 dataset by only Single Tag Decombinator, only by MiXCR, or by both platforms, are shown in orange, blue and green, respectively, using CDR3 information only.

ID	DCR	ID	DCR
1	44, 8, 0, 2, CCGAAGGACTAGACCT	31	31, 6, 5, 2, TACAGCGGGGGGG
2	19, 4, 5, 0, AAATGGGGACGGATCTTCGAACA	32	25, 5, 5, 0, GAATTTATGGGAGA
3	14, 10, 4, 2, GTTCTCGGACAGGA	33	25, 6, 5, 0, GAATTTATGGGAGA
4	25, 6, 0, 4, TCGAAGCGGGACT	34	12, 9, 3, 7, TCTTGCAGGGGGCCCACTG
5	15, 12, 3, 2, TGTACCCA	35	4, 0, 3, 1, TCCGGG
6	19, 6, 6, 6, CCATAGCGGTGCCCT	36	42, 5, 5, 1, CGCGGCCCGGGGTCA
7	19, 6, 2, 9, GGTTGGCTAAG	37	42, 5, 5, 24, CGCGGCCCGGGGTCA
8	43, 11, 2, 0, CGGGACGGATT	38	31, 0, 6, 2, ATTCCGACAGGGCGCCATT
9	28, 11, 2, 0, ACGGGACGGATT	39	43, 2, 4, 19, CAACCAGCCCGAGGACAGGGCCTCTCTGGAAACACCATATA
10	6, 0, 5, 11, CCCCGGGCAGGATGGAGG	40	25, 0, 0, 1, TTTCGGGGGGGGGGG
11	42, 1, 3, 8, TGACTCGGGACAGGGGTATATTTGGGAA	41	25, 0, 0, 0, TTTCGGGGGGGGGG
12	37, 1, 3, 8, TGACTCGGGACAGGGGTATATTTGGGAA	42	44, 5, 4, 17, TCCCGGACAGGAGAGA
13	6, 6, 7, 9, CCTTGACCCTGCC	43	15, 0, 5, 7, GATTGGTCTCG
14	13, 0, 4, 1, GCCCATCGGA	44	6, 0, 6, 6, CCTACTTGGGATTGCCG
15	24, 3, 1, 6, GGTGGGGGACCTTAG	45	6, 3, 5, 6, CTAGACAGGGCTTTCG
16	36, 12, 5, 6, CCCTCCGGGACAGGGCCCAC	46	25, 3, 0, 0, CCTGCCGCCCTGGGACAGGGTTG
17	33, 12, 5, 6, CCCTCCGGGACAGGGCCCAC	47	25, 4, 0, 14, CCTGCCGCCCTGGGACAGGTATGCAACCAGAGTACGGG
18	18, 6, 3, 2, TGATATAGCGGAGG	48	33, 3, 5, 0, CCAGGGACCTTTG
19	37, 6, 1, 0, GACTTCG	49	6, 4, 6, 5, CTATGCCGGGG
20	31, 6, 9, 12, TAGCGGGAGGGCCCTCGTAGG	50	31, 6, 5, 6, ACTGGACCAAG
21	44, 7, 3, 9, GCCGGGACCCCAACCCGACCA	51	36, 3, 5, 0, CCAGGGACCTTTG
22	37, 6, 5, 4, GCCCCCCCGGACCAT	52	19, 4, 6, 3, CCCGAC
23	44, 7, 3, 5, GCCGGGACCCCAACCCG	53	4, 12, 3, 1, CCCTTCAGGA
24	15, 0, 0, 3, TCTGGGGT	54	27, 0, 0, 3, GCCCACACAGGTTCCGGT
25	30, 4, 2, 0, CCAGGGCTA	55	40, 1, 1, 4, GGATTACGGG
26	19, 0, 6, 3, CCCCAGGACAGGTACAG	56	43, 12, 4, 1, CAGGG
27	8, 0, 4, 4, CCAGGGACAGGGGGGGTGG	57	54, 7, 3, 0, GTCGAAAGGATCCGAAGTGCCGGGACAGGTCTTCG
28	2, 0, 3, 4, AGGGGGGCTTGGGGT	58	43, 12, 5, 3, CCTCGGACAGGGGG
29	54, 12, 3, 4, TGGCCCAGGAGGA	59	3, 12, 4, 1, CAGGG
30	29, 1, 0, 6, CGGCAGGGGAGG		

Table C.2: Mapping between representative IDs and β chain Decombinator classifiers (DCRs) used in Figure 2.17 for the NSCLC single cell analysis comparison of Single Tag Decombinator and TraCeR.

ID	Cell	ID	Cell	ID	Cell	ID	Cell
1	VT10_N701_S502	36	VT10_N705_S505	71	VT10_N709_S517	106	VT8_N703_S504
2	VT10_N701_S503	37	VT10_N705_S506	72	VT10_N710_S502	107	VT8_N703_S517
3	VT10_N701_S504	38	VT10_N705_S507	73	VT10_N710_S503	108	VT8_N704_S502
4	VT10_N701_S505	39	VT10_N705_S508	74	VT10_N710_S504	109	VT8_N704_S503
5	VT10_N701_S506	40	VT10_N706_S502	75	VT10_N710_S505	110	VT8_N704_S504
6	VT10_N701_S507	41	VT10_N706_S503	76	VT10_N710_S506	111	VT8_N704_S517
7	VT10_N701_S508	42	VT10_N706_S504	77	VT10_N710_S507	112	VT8_N705_S503
8	VT10_N701_S517	43	VT10_N706_S505	78	VT10_N710_S508	113	VT8_N705_S504
9	VT10_N702_S502	44	VT10_N706_S506	79	VT10_N710_S517	114	VT8_N705_S517
10	VT10_N702_S503	45	VT10_N706_S507	80	VT10_N711_S502	115	VT8_N706_S502
11	VT10_N702_S504	46	VT10_N706_S508	81	VT10_N711_S503	116	VT8_N706_S503
12	VT10_N702_S505	47	VT10_N706_S517	82	VT10_N711_S504	117	VT8_N706_S504
13	VT10_N702_S506	48	VT10_N707_S502	83	VT10_N711_S505	118	VT8_N706_S517
14	VT10_N702_S507	49	VT10_N707_S503	84	VT10_N711_S506	119	VT8_N707_S502
15	VT10_N702_S508	50	VT10_N707_S504	85	VT10_N711_S507	120	VT8_N707_S503
16	VT10_N702_S517	51	VT10_N707_S505	86	VT10_N711_S508	121	VT8_N707_S504
17	VT10_N703_S502	52	VT10_N707_S506	87	VT10_N711_S517	122	VT8_N707_S517
18	VT10_N703_S503	53	VT10_N707_S507	88	VT10_N712_S502	123	VT8_N708_S502
19	VT10_N703_S504	54	VT10_N707_S508	89	VT10_N712_S503	124	VT8_N708_S503
20	VT10_N703_S505	55	VT10_N707_S517	90	VT10_N712_S504	125	VT8_N708_S504
21	VT10_N703_S506	56	VT10_N708_S502	91	VT10_N712_S505	126	VT8_N708_S517
22	VT10_N703_S507	57	VT10_N708_S503	92	VT10_N712_S506	127	VT8_N709_S502
23	VT10_N703_S508	58	VT10_N708_S504	93	VT10_N712_S507	128	VT8_N709_S503
24	VT10_N703_S517	59	VT10_N708_S505	94	VT10_N712_S508	129	VT8_N709_S517
25	VT10_N704_S502	60	VT10_N708_S506	95	VT10_N712_S517	130	VT8_N710_S502
26	VT10_N704_S503	61	VT10_N708_S507	96	VT8_N701_S502	131	VT8_N710_S503
27	VT10_N704_S504	62	VT10_N708_S508	97	VT8_N701_S503	132	VT8_N710_S517
28	VT10_N704_S505	63	VT10_N708_S517	98	VT8_N701_S504	133	VT8_N711_S502
29	VT10_N704_S506	64	VT10_N709_S502	99	VT8_N701_S517	134	VT8_N711_S503
30	VT10_N704_S507	65	VT10_N709_S503	100	VT8_N702_S502	135	VT8_N711_S517
31	VT10_N704_S508	66	VT10_N709_S504	101	VT8_N702_S503	136	VT8_N712_S502
32	VT10_N704_S517	67	VT10_N709_S505	102	VT8_N702_S504	137	VT8_N712_S503
33	VT10_N705_S502	68	VT10_N709_S506	103	VT8_N702_S517	138	VT8_N712_S517
34	VT10_N705_S503	69	VT10_N709_S507	104	VT8_N703_S502		
35	VT10_N705_S504	70	VT10_N709_S508	105	VT8_N703_S503		

Table C.3: Mapping between representative IDs and cell names used in Figure 2.16 and Figure 2.17 for the NSCLC single cell analysis comparison of Single Tag Decombinator and TraCeR.

ID	V	J	CDR3	ID	V	J	CDR3
1	TRAV8-6	TRAJ22	CAVSGPQGSARQLTF	17	TRAV8-2	TRAJ3	CAVTPLGSSASKIIF
2	TRAV12-1	TRAJ38	CVVNPPSAGNNRKLIW	18	TRAV8-1	TRAJ23	CAVNALYNQGGKLIF
3	TRAV14/DV4	TRAJ8	CAMREGTGFQKLVF	19	TRAV8-4	TRAJ3	CAVTPLGSSASKIIF
4	TRAV12-2	TRAJ18	CAVAVMALGRLYF	20	TRAV3	TRAJ3	CAVTPLGSSASKIIF
5	TRAV12-3	TRAJ36	CAMNNNLFF	21	TRAV8-6	TRAJ3	CAVTPLGSSASKIIF
6	TRAV3	TRAJ40		22	TRAV8-1	TRAJ23	CDVNALYNQGGKLIF
7	TRAV26-1	TRAJ28	CIVSALLRESYQLTF	23	TRAV8-3	TRAJ39	
8	TRAV25	TRAJ24	CAGWAEGWGKLQF	24	TRAV17	TRAJ7	CATGGNNRLTF
9	TRAV17	TRAJ7	CATGGNNRLAF	25	TRAV20	TRAJ35	CAVQAEGFGNVLHC
10	TRAV25	TRAJ15	CAGPNQAGTALIF	26	TRAV14/DV4	TRAJ43	CAMRVQLNDMRF
11	TRAV35	TRAJ52	CAGQWAGGTSYGKLTF	27	TRAV25	TRAJ15	CVGPNQAGTALIF
12	TRAV12-1	TRAJ13		28	TRAV24	TRAJ22	CAPRGPGSARQLTF
13	TRAV25	TRAJ16	CAGRSDGQKLLF	29	TRAV25	TRAJ47	CAGQIVGINKLVF
14	TRAV25	TRAJ16	CAGRSDGQKLLL	30	TRAV35	TRAJ52	CAGPRRLNAGGTSYGKLTF
15	TRAV14/DV4	TRAJ9	CAMKTGGFKTIF	31	TRAV14/DV4	TRAJ52	CAMRDPSNAGGTSYGKLTF
16	TRAV12-2	TRAJ9	CAVKWDGGFKTIF				

Table C.4: Mapping between representative IDs and α chain V genes, J genes and CDR3s used in Figure 2.18A for the SARS-CoV-2 single cell analysis comparison between Single Tag Decombinator and MiXCR. Blank CDR3s represent where the V and J gene have been annotated as directly joining one another with no insert.

ID	V	J	CDR3	ID	V	J	CDR3
1	TRBV5-1	TRBJ2-5	CASSLWAGGETQYF	24	TRBV4-3	TRBJ2-7	CASSQVSGSGYEQYF
2	TRBV13	TRBJ1-2	CASSLIGQGGYTF	25	TRBV7-3	TRBJ2-3	CASSLGGGSTDTQYF
3	TRBV13	TRBJ1-2	CASSLFGQGGYTF	26	TRBV20-1	TRBJ1-2	CSVTRTYPRCYTF
4	TRBV7-3	TRBJ1-2	CASSLIGQGGYTF	27	TRBV28	TRBJ2-7	CASSSVGSNEQYF
5	TRBV2	TRBJ2-6		28	TRBV27	TRBJ2-7	CASRELDGYEQYF
6	TRBV4-2	TRBJ2-1	CASSQTVGQGYEQFF	29	TRBV5-1	TRBJ2-7	CASSVWTGVGYEQYF
7	TRBV9	TRBJ2-3	CASSEGQGSTDTQYF	30	TRBV20-1	TRBJ1-2	
8	TRBV5-1	TRBJ1-2	CASSLIGQGGYTF	31	TRBV14	TRBJ2-1	CASSQDLYNEQFF
9	TRBV12-4	TRBJ1-5	CASSVRAPGQPQHF	32	TRBV20-1	TRBJ1-2	CSVTRTHPRYYTF
10	TRBV12-4	TRBJ1-5	CASSVRAPVQPQHF	33	TRBV28	TRBJ2-7	CVSSSVGSNEQYF
11	TRBV12-4	TRBJ1-5	CASSVRAPGQQQHF	34	TRBV2	TRBJ2-2	CASSDPTSGELFF
12	TRBV13	TRBJ1-5	CASSVRAPGQPQHF	35	TRBV7-3	TRBJ2-3	CASSLGGSSTDTQYF
13	TRBV9	TRBJ1-5	CASSVRAPGQPQHF	36	TRBV26	TRBJ2-3	YASSLGGSSTDTQYF
14	TRBV6-4	TRBJ1-5	CASSVRAPGQPQHF	37	TRBV13	TRBJ1-2	
15	TRBV13	TRBJ1-2	CASSLIGQGSYTF	38	TRBV6-5	TRBJ2-6	CASSLGQEGANVLTF
16	TRBV4-1	TRBJ2-3	CASSVGGGSDTQYF	39	TRBV13	TRBJ2-6	CASSLGQEGANVLTF
17	TRBV16	TRBJ2-1	CASSQDLYNEQFF	40	TRBV4-1	TRBJ2-7	CASSQVSGVGYEQYF
18	TRBV20-1	TRBJ1-2	CSVTRTHPRCYTF	41	TRBV4-3	TRBJ2-7	CASSQVSGVGYEQYF
19	TRBV7-9	TRBJ1-6	CASSLAGQGDGSPLHF	42	TRBV16	TRBJ1-5	CASSPDQGAPQHF
20	TRBV6-9	TRBJ2-7	CASSYTTSPHEQYF	43	TRBV23-1	TRBJ1-3	CASSHLSGTGHPLRRSGNTIYF
21	TRBV6-6	TRBJ2-7	CASSYTTSPHEQYF	44	TRBV16	TRBJ1-5	CANSPDQGAPQHF
22	TRBV27	TRBJ2-7	CASSYTTSPHEQYF	45	TRBV3-2	TRBJ1-5	CASSPDQGAPQHF
23	TRBV4-1	TRBJ2-7	CASSQVSGSGYEQYF	46	TRBV12-4	TRBJ2-5	CASSLLAGGLETQYF

Table C.5: Mapping between representative IDs and β chain V genes, J genes and CDR3s used in Figure 2.18B for the SARS-CoV-2 single cell analysis comparison between Single Tag Decombinator and MiXCR. Blank CDR3s represent where the V and J gene have been annotated as directly joining one another with no insert.

ID	Cell	ID	Cell	ID	Cell
1	P3_A01_S25	17	P3_B05_S81	33	P3_C09_S137
2	P3_A02_S27	18	P3_B06_S83	34	P3_C10_S139
3	P3_A03_S29	19	P3_B07_S85	35	P3_C11_S141
4	P3_A04_S31	20	P3_B08_S87	36	P3_C12_S143
5	P3_A05_S33	21	P3_B09_S89	37	P3_D01_S169
6	P3_A06_S35	22	P3_B10_S91	38	P3_D02_S171
7	P3_A07_S37	23	P3_B11_S93	39	P3_D03_S173
8	P3_A08_S39	24	P3_B12_S95	40	P3_D04_S175
9	P3_A09_S41	25	P3_C01_S121	41	P3_D05_S177
10	P3_A10_S43	26	P3_C02_S123	42	P3_D06_S179
11	P3_A11_S45	27	P3_C03_S125	43	P3_D07_S181
12	P3_A12_S47	28	P3_C04_S127	44	P3_D08_S183
13	P3_B01_S73	29	P3_C05_S129	45	P3_D09_S185
14	P3_B02_S75	30	P3_C06_S131	46	P3_D10_S187
15	P3_B03_S77	31	P3_C07_S133	47	P3_D11_S189
16	P3_B04_S79	32	P3_C08_S135	48	P3_D12_S191

Table C.6: Mapping between representative IDs and cell names used in Figure 2.18 and Figure C.1 for the SARS-CoV-2 single cell analysis comparison between Single Tag Decombinator and MiXCR.

ID	CDR3	ID	CDR3
1	CAVSGPQGSARQLTF	14	CAMKTGGFKTIF
2	CVVNPPSAGNNRKLIW	15	CAVKWDGGFKTIF
3	CAMREGTGFQKLVF	16	CAVTPLGSSASKIIF
4	CAVAVMALGRLYF	17	CAVNALYNQGGKLIF
5	CAMNNNLFF	18	CDVNALYNQGGKLIF
6		19	CATGGNNRLTF
7	CIVSALLRESYQLTF	20	CAVQAEGFGNVLHC
8	CAGWAEGWGKLQF	21	CAMRVQLNDMRF
9	CATGGNNRLAF	22	CVGPNQAGTALIF
10	CAGPNQAGTALIF	23	CAPRGPGSARQLTF
11	CAGQWAGGTSYGKLTF	24	CAGQIVGINKLVF
12	CAGRSDGQKLLF	25	CAGPRRLNAGGTSYGKLTF
13	CAGRSDGQKLLL	26	CAMRDPSNAGGTSYGKLTF

Table C.7: Mapping between representative IDs and α chain CDR3s used in Figure C.1A for the SARS-CoV-2 single cell analysis comparison between Single Tag Decombinator and MiXCR. Blank CDR3s represent where the V and J gene have been annotated as directly joining one another with no insert.

ID	CDR3	ID	CDR3
1	CASSLWAGGETQYF	17	CASSLGGGSTDTQYF
2	CASSLIGQGGYTF	18	CSVTRTYPRCYTF
3	CASSLFGQGGYTF	19	CASSSVGSNEQYF
4		20	CASRELDGYEQYF
5	CASSQTVGQGYEQFF	21	CASSVWTGVGYEQYF
6	CASSEGQGSTDTQYF	22	CSVTRTHPRYYTF
7	CASSVRAPGQPQHF	23	CVSSSVGSNEQYF
8	CASSVRAPVQPQHF	24	CASSDPTSGELFF
9	CASSVRAPGQQQHF	25	CASSLGGSSTDTQYF
10	CASSLIGQGSYTF	26	YASSLGGSSTDTQYF
11	CASSVGGGSDTQYF	27	CASSLGQEGANVLTF
12	CASSQDLYNEQFF	28	CASSQVSGVGYEQYF
13	CSVTRTHPRCYTF	29	CASSPDQGAPQHF
14	CASSLAGQGDGSPLHF	30	CASSHLSGTGHPLRRSGNTIYF
15	CASSYTTSPHEQYF	31	CANSPDQGAPQHF
16	CASSQVSGSGYEQYF	32	CASSLLAGGLETQYF

Table C.8: Mapping between representative IDs and β chain CDR3s used in Figure C.1B for the SARS-CoV-2 single cell analysis comparison between Single Tag Decombinator and MiXCR. Blank CDR3s represent where the V and J gene have been annotated as directly joining one another with no insert.

Appendix D

Decombinator Tools

The Decombinator-Tools repository (https://github.com/innate2adaptive/Decombinator-Tools) hosts a number of scripts that may be of used when working with Decombinator. An overview of these scripts is provided below, and a full README of operating instructions is provided within the repository.

collapsed_sample_overlap.R

This script can be used to measure the TCR overlap (using the DCR identifier) of samples produced using Collapsinator from Decombinator V4. The path to collapsed files should be supplied as input, and overlap heatmaps are generated as output. This script was used to generated the plots in Figure 2.10, and was written in R by Dr Tahel Ronel.

DCRtoGeneName.py

This script can be used to automatically convert the V and J tag indices in a file of DCR identifiers (used as a shorthand by the Decombinator pipeline scripts) to V and J gene names. For example, the identifier:

```
30, 37, 8, 7, CTGGGG
```

becomes:

```
TRAV38-2/DV8, TRAJ48, 8, 7, CTGGGG
```

This was a legacy script that has been recently upgraded to Python 3 and refactored to run from Decombinator-Tools as part of the Decombinator V4 release. The original code was written by Dr James Heather.

ExactSearch.py

This script performs searches for exact subsequences in FASTQ data. While it does not account for sequence mismatches, it can be useful as a rapid assessment that subsequences of interest are within frame for the reads produced by experiment.

The script takes a FASTQ file to be searched, and a file listing subsequences to search for, as input arguments. Additional arguments can be included to also search the data for the reverse, the complement, and the reverse complement of the supplied subsequences. An output FASTQ file is generated for each subsequence containing all reads of the original FASTQ file that feature that subsequence. This feature can be suppressed through an input argument. The output files are written to an output directory named after the input FASTQ file, which is in turn stored in the "ExactSearchResults" directory (which is created if it does not exist), alongside a log file summarising the total counts of each subsequence.

ExactSearchLogSummary.py

This script can be used to generate a summary of log files produced by the `ExactSearch` script. Log files are collated from each subdirectory in the "ExactSearchResults" directory. A table is constructed listing the file, line count, read count, average read length, and number of subsequence matches found, for each log file. This can be useful for the quick assessment that subsequences of interest feature in multiple FASTQ files.

LogSummary.py

This script collates log file data stored in the Logs directory produced during Decombinator and Collapsinator runs. An output comma-separated values (`.csv`) file is generated summarising quantitative information from the run, including total and unique DCR counts, average RNA duplication, and a measure of average sequence quality. This can be useful in quickly providing a broad overview of how many hits were found for various samples at once.

This was a legacy script that has been recently upgraded to Python 3 and refactored to run from Decombinator-Tools as part of the Decombinator V4 release. The original code was written by Dr Mazlina Ismail. Additionally, a file of sample names can be provided as an input argument to the script as an ordering for the final report. If not included, the report will contain samples containing "alpha" in their name, followed by those containing "beta", followed by any additional samples that contain neither, all sorted alphabetically.

RandomlySample.py

This script can be used to randomly sub-sample a file produced from the Decombinator pipeline down to a specified sample size.

This was a legacy script that has been recently upgraded to Python 3 and refactored to run from Decombinator-Tools as part of the Decombinator V4 release. The original code was written by Dr James Heather.

RunTestData.py

The Decombinator-Test-Data repository (`https://github.com/innate2adaptive/Decombinator-Test-Data`) contains a set of multiplexed data files produced by the Chain lab experimental protocol. It provides a convenient starting point for users who wish to test their installation of Decombinator, as well as an end-to-end test case for developers when modification is made to the pipeline.

The RunTestData.py script will attempt to automatically complete an end-to-end analysis of the test data using the Decombinator pipeline — that is to say, it will attempt to run the data set through the Demultiplexor, Decombinator, Collapsinator, and CDR3translator scripts in order. Users can choose to set start and stop checkpoints for the script — for example, running data only through Demultiplexor and Decombinator, or from only Collapsinator onwards, and so on.

SortSummary.py

This script can be used to sort the summary files generated by the `LogSummary` script. The script takes the log summary file and a file of sample names as input arguments, and outputs a new summary sorted according to the sample sheet.

TestDataGenerator.py

This script can be used to produce custom FASTQ files as test input data for the Decombinator script. It should be noted that this data is not simulated from biological principles (it does not simulate recombination of genes, or error through sequencing or PCR amplification), but creates data using the set of V and J tags used as part of the Decombinator algorithm, and according to user-specified information. V and J tags are chosen at random for each generated TCR read.

The test generator provides customisability allowing users to choose one of the two oligonucleotide protocols used by the Chain lab ("I8" or "M13"), one of two species (mouse or human), and whether the reads should be or not be (entirely or partially) reverse-complemented. Test data can be produced with or without barcodes, and can be run to use only a single V or J tag when building each read, to generate data for Single Tag Decombinator (Section 2.4). By default, the script will produce both an exclusive α and an exclusive β FASTQ file, but users may specify the output files to contain a mixture of α and β reads if they wish. Finally, users may also specify the read length (before barcoding), the barcode length, the total number of reads, the percentage of reads that contain TCRs, and the percentage of reads containing one sequencing error.

UMIHistogram.py

Users can choose to generate a data file containing the average size of the UMI cluster sizes associated with each DCR when running Collapsinator. This script can be used to generate a histogram plot from the data file, which can be useful when investigating the effects of PCR amplification on the sample. Input arguments make it easy to quickly customise the number of bins, and the colour and DPI (Dots Per Inch) of the resultant plot.

Recipes

The "recipes" directory contains a number of bash shell scripts that can be useful for running the pipeline scripts over multiple input files in parallel. Additionally, a submission script is provided for running the Decombinator test data set through the entire pipeline on cluster or grid computing systems that make use of job scheduling. Typical data analysis jobs using Decombinator in the Chain lab are run on the Myriad cluster at University College London. This script may serve as a template for running customised analysis using similar computing resources. These templates were written by the thesis author and by Professor Benny Chain.

Appendix E

Binding Residue Specification Formatting

For each of the benchmark docking cases, and for each docking scenario, the residues involved in the binding interface were determined. These residues were automatically formatted for use in the four docking platforms - ClusPro, HADDOCK, LightDock and ZDOCK.

ClusPro

Listing E.1 and Listing E.2 provide the ClusPro format for the receptor and ligand residues respectively for example case 1AO7 for Scenario 4. These can be immediately pasted into the two boxes in the `attraction` section in `Advanced options` on the ClusPro web server docking page.

```
D-2 D-26 D-27 D-28 D-29 D-37 D-38 D-57 D-58 D-63 D-82 D-107 D-108
   D-109 D-110 D-113 D-114 E-37 E-107 E-110 E-111 E-112 E-113 E-114
   E-115
```

Listing E.1: ClusPro format for receptor residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

```
A-58 A-65 A-66 A-68 A-69 A-72 A-73 A-149 A-150 A-151 A-154 A-155
   A-158 A-159 A-163 A-166 A-167 A-170 C-1 C-2 C-3 C-4 C-5 C-6 C-7
   C-8
```

Listing E.2: ClusPro format for ligand residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

HADDOCK

Listing E.3 and Listing E.4 provide the HADDOCK format for the receptor and ligand residues respectively for example case 1AO7 for Scenario 4. These can be immediately pasted into the `active residues` boxes for the receptor and ligand respectively in the `input parameters` tab on the HADDOCK web server docking submission page.

The digression from the usual numbering scheme should be noted here. The HADDOCK web server requires that each component in the docking features only a single chain ID. Consequently, a mask was applied to the input PDB files such that the chain ID of both α and β chains of the TCR were set to `A`, and the chain ID of the peptide and two MHC components were set to `B`. To avoid clashes in residue ID in a single component, an additional mask was applied. The second TCR chain has a value of 1000 temporarily added to each residue ID. The second and third pMHC chains had values of 1000 and 2000 added to each residue ID respectively. After modelling with HADDOCK, the reverse of these operations was performed to return to the standardised chain and residue labelling. All the benchmark cases can be found formatted for HADDOCK according to this description in the `haddock` directory at `https://github.com/innate2adaptive/ExpandedBenchmark`.

```
2, 26, 27, 28, 29, 37, 38, 57, 58, 63, 82, 107, 108, 109, 110, 113,
   114, 1037, 1107, 1110, 1111, 1112, 1113, 1114, 1115
```

Listing E.3: HADDOCK format for receptor residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

```
58, 65, 66, 68, 69, 72, 73, 149, 150, 151, 154, 155, 158, 159, 163,
    166, 167, 170, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008
```

Listing E.4: HADDOCK format for ligand residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

LightDock

Listing E.5 provides the LightDock format for the receptor and ligand residues for example case 1AO7 for Scenario 4. Receptor residues are given the prefix `R` and ligand residues the prefix `L`. Scenario 4 features only active residues. If passive residues are to be included, they should feature the additional `P` flag, e.g. `R D.LYS.2 P`. These formatted restraints are saved to a file which is provided to LightDock via an input argument.

```
R D.LYS.2
R D.SER.26
R D.ASP.27
R D.ARG.28
R D.GLY.29
R D.GLN.37
R D.SER.38
R D.TYR.57
R D.SER.58
R D.ASN.63
R D.LYS.82
R D.THR.107
R D.THR.108
R D.ASP.109
R D.SER.110
R D.TRP.113
R D.GLY.114
R E.GLU.37
```

```
R E.ARG.107

R E.LEU.110

R E.ALA.111

R E.GLY.112

R E.GLY.113

R E.ARG.114

R E.PRO.115

L A.GLU.58

L A.ARG.65

L A.LYS.66

L A.LYS.68

L A.ALA.69

L A.GLN.72

L A.THR.73

L A.ALA.149

L A.ALA.150

L A.HIS.151

L A.GLU.154

L A.GLN.155

L A.ALA.158

L A.TYR.159

L A.THR.163

L A.GLU.166

L A.TRP.167

L A.ARG.170

L C.LEU.1

L C.LEU.2

L C.PHE.3

L C.GLY.4

L C.TYR.5

L C.PRO.6

L C.VAL.7
```

```
L C.TYR.8
```

Listing E.5: LightDock format for residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

ZDOCK

Receptor and ligand residues involved in the binding interface are individually supplied to the ZDOCK web server using the residue selection screen. This is extremely cumbersome if performing a large number of docking runs. The manual selection problem was bypassed by generating custom JavaScript for each docking case that can be pasted into the developer tools console of the web browser to automatically select all relevant residues. This procedure was performed using Google Chrome, and is functional in the latest version 92.0.4515.107. Listing Listing E.6 provides the ZDOCK JavaScript for the receptor and ligand residues for example case 1AO7 for Scenario 4.

```
// receptor
var x = document.getElementsByName("nonContactRec[]")[0].options;
for(var i=0;i<x.length;i++){
   var res = x[i].text.split(" ");
   var res = res[0] + " " + res[1] + " " + res[2]
   if(['2 Chain D', '26 Chain D', '27 Chain D', '28 Chain D', '29
       Chain D', '37 Chain D', '38 Chain D', '57 Chain D', '58 Chain
       D', '63 Chain D', '82 Chain D', '107 Chain D', '108 Chain D',
       '109 Chain D', '110 Chain D', '113 Chain D', '114 Chain D', '37
       Chain E', '107 Chain E', '110 Chain E', '111 Chain E', '112
       Chain E', '113 Chain E', '114 Chain E', '115 Chain
       E'].includes(res)){
      x[i].selected=false;
      }
   else{
      x[i].selected=true;
   }
}
// ligand
var x = document.getElementsByName("nonContactLig[]")[0].options;
for(var i=0;i<x.length;i++){
   var res = x[i].text.split(" ");
   var res = res[0] + " " + res[1] + " " + res[2]
   if(['58 Chain A', '65 Chain A', '66 Chain A', '68 Chain A', '69
       Chain A', '72 Chain A', '73 Chain A', '149 Chain A', '150 Chain
       A', '151 Chain A', '154 Chain A', '155 Chain A', '158 Chain A',
       '159 Chain A', '163 Chain A', '166 Chain A', '167 Chain A',
       '170 Chain A', '1 Chain C', '2 Chain C', '3 Chain C', '4 Chain
       C', '5 Chain C', '6 Chain C', '7 Chain C', '8 Chain
       C'].includes(res)){
      x[i].selected=false;
      }
   else{
      x[i].selected=true;
   }
}
```

Listing E.6: Custom ZDOCK JavaScript for residues involved in the binding interface using Scenario 4 information for the benchmark case 1AO7.

Appendix F

CDR1 and CDR2 Modelling Performance

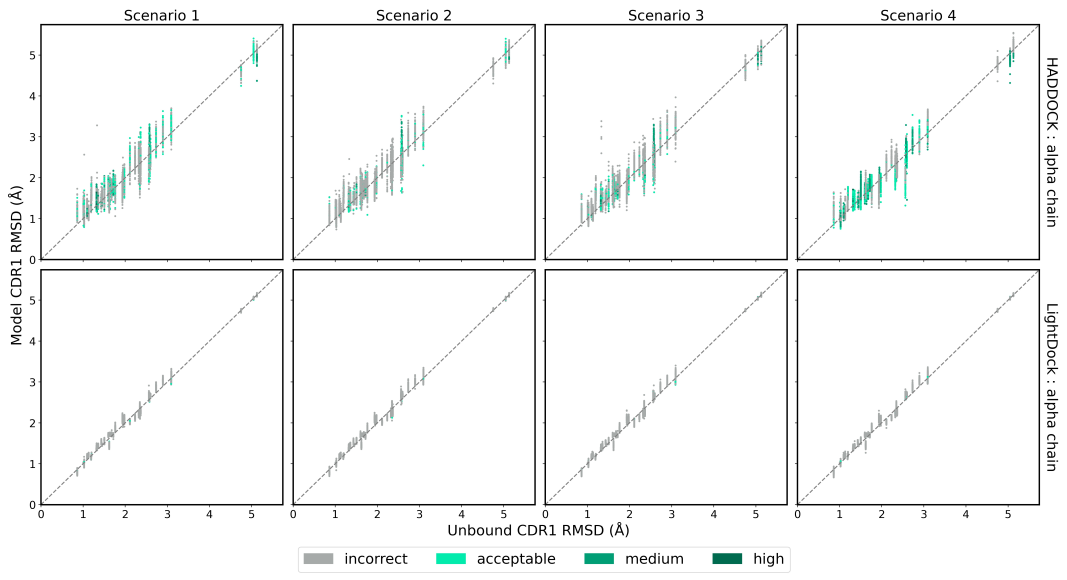


Figure F.1: The RMSD of the TCR α chain CDR1 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

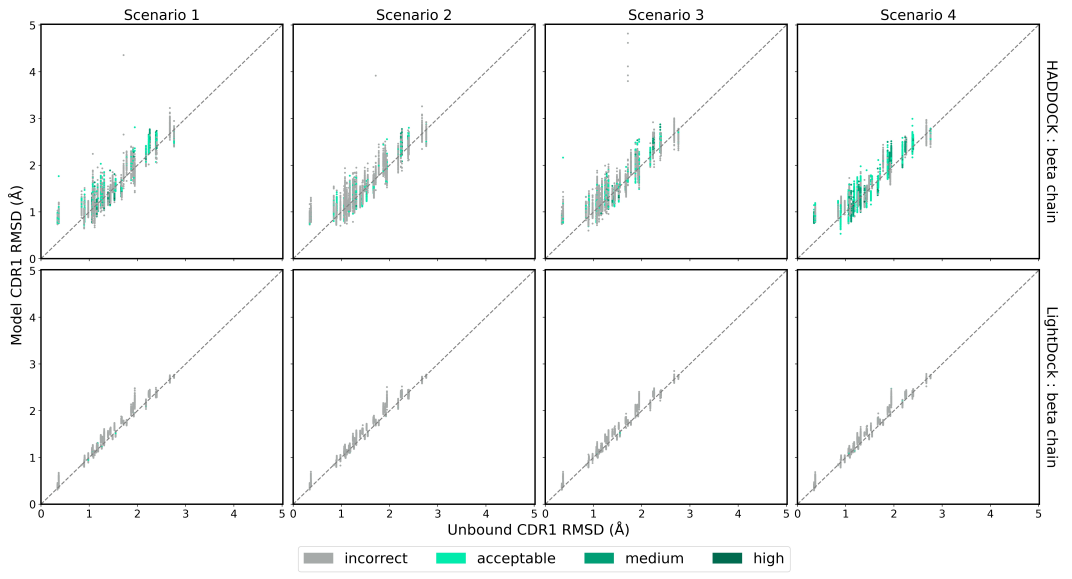


Figure F.2: The RMSD of the TCR β chain CDR1 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

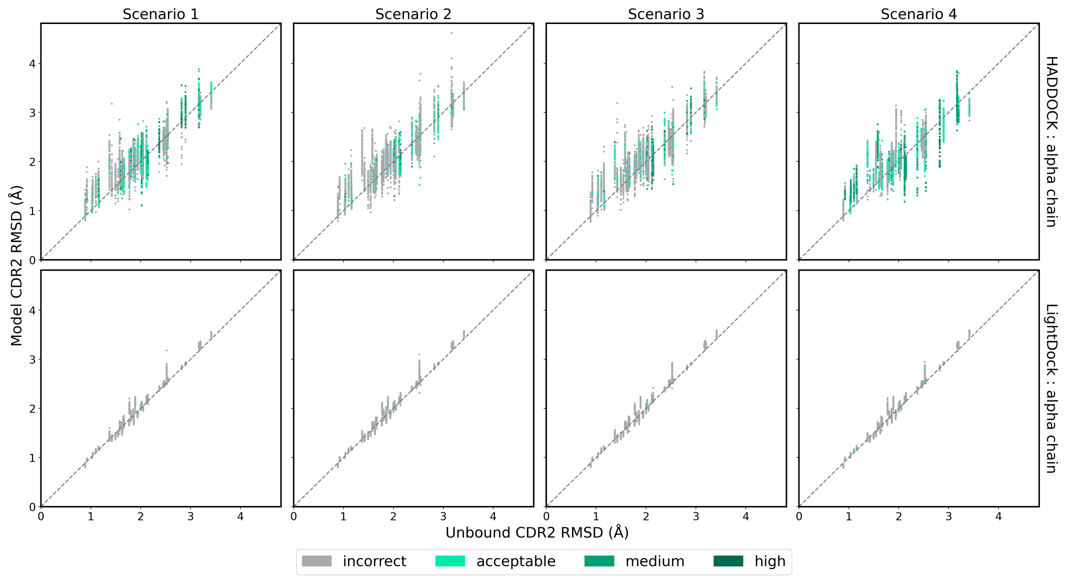


Figure F.3: The RMSD of the TCR α chain CDR2 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

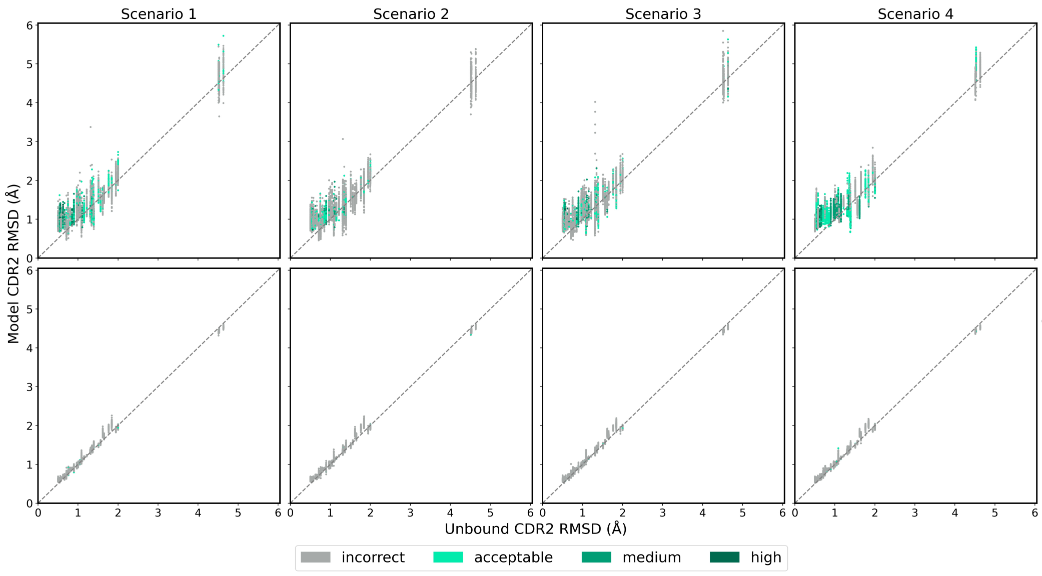


Figure F.4: The RMSD of the TCR β chain CDR2 loop between the unbound TCR and the reference structure versus that between each of the docked models and the reference structure, for each complex. Loop flexibility modelling by HADDOCK is shown in the top row and by LightDock in the bottom row. Models are coloured by their quality according to the CAPRI criteria.

Appendix G

Colophon

This document was set in the Times Roman typeface using LaTeX and BibTeX, drafted in the Sublime text editor, and composed with Overleaf.

Bibliography

K. R. Abhinandan and A. C. R. Martin. Analysis and prediction of VH/VL packing in antibodies. *Protein Engineering, Design and Selection*, 23(9):689–697, 2010.

J. J. Adams, S. Narayanan, B. Liu, M. E. Birnbaum, A. Kruse, N. A. Bowerman, W. Chen, A. M. Levin, J. M. Connolly, C. Zhu, D. M. Kranz, and K. C. Garcia. T cell receptor signaling is limited by docking geometry to peptide-Major Histocompatibility Complex. *Immunity*, 35(5):681–693, 2011.

P. Agrawal, H. Singh, H. K. Srivastava, S. Singh, G. Kishore, and G. P. S. Raghava. Benchmarking of different molecular docking methods for protein-peptide docking. *BMC Bioinformatics*, 19(13):426, 2019.

A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

R. Akbar, P. A. Robert, M. Pavlović, J. R. Jeliazkov, I. Snapkov, A. Slabodkin, C. R. Weber, L. Scheffer, E. Miho, I. H. Haff, D. T. T. Haug, F. Lund-Johansen, Y. Safonova, G. K. Sandve, and V. Greiff. A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding. *Cell Reports*, 34 (11):108856, 2021.

B. Al-Lazikani, A. M. Lesk, and C. Chothia. Canonical structures for the hypervariable regions of T cell $A\beta$ receptors. *Journal of Molecular Biology*, 295(4): 979–995, 2000.

R. Alli, Z. M. Zhang, P. Nguyen, J. J. Zheng, and T. L. Geiger. Rational Design of T

Cell Receptors with Enhanced Sensitivity for Antigen. *PLOS ONE*, 6(3):e18027, 2011.

M. AlQuraishi. Machine learning in protein structure prediction. *Current Opinion in Chemical Biology*, 65:1–8, 2021.

F. Ambrosetti, B. Jiménez-García, J. Roel-Touris, and A. M. J. J. Bonvin. Modeling Antibody-Antigen Complexes by Information-Driven Docking. *Structure*, 28(1): 119–129.e2, 2020a.

F. Ambrosetti, T. H. Olsen, P. P. Olimpieri, B. Jiménez-García, E. Milanetti, P. Marcatilli, and A. M. J. J. Bonvin. proABC-2: PRediction of AntiBody contacts v2 and its application to information-driven docking. *Bioinformatics*, 36(20):5107–5108, 2020b.

N. Andrusier, E. Mashiach, R. Nussinov, and H. J. Wolfson. Principles of Flexible Protein-Protein Docking. *Proteins*, 73(2):271–289, 2008.

I. Antes. DynaDock: A new molecular dynamics-based algorithm for protein–peptide docking including receptor flexibility. *Proteins*, 78(5):1084–1104, 2010.

A. R. Atilgan, S. R. Durell, R. L. Jernigan, M. C. Demirel, O. Keskin, and I. Bahar. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophysical Journal*, 80(1):505–515, 2001.

L. Au, E. Hatipoglu, M. R. de Massy, K. Litchfield, G. Beattie, A. Rowan, D. Schnidrig, R. Thompson, F. Byrne, S. Horswell, N. Fotiadis, S. Hazell, D. Nicol, S. T. C. Shepherd, A. Fendler, R. Mason, L. D. Rosario, K. Edmonds, K. Lingard, S. Sarker, M. Mangwende, E. Carlyle, J. Attig, K. Joshi, I. Uddin, P. D. Becker, M. W. Sunderland, A. Akarca, I. Puccio, W. W. Yang, T. Lund, K. Dhillon, M. D. Vasquez, E. Ghorani, H. Xu, C. Spencer, J. I. López, A. Green, U. Mahadeva, E. Borg, M. Mitchison, D. A. Moore, I. Proctor, M. Falzon, L. Pickering, A. J. S. Furness, J. L. Reading, R. Salgado, T. Marafioti,

M. Jamal-Hanjani, P. Consortium, G. Kassiotis, B. Chain, J. Larkin, C. Swanton, S. A. Quezada, S. Turajlic, and T. R. Consortium. Determinants of anti-PD-1 response and resistance in clear cell renal cell carcinoma. *Cancer Cell*, 39(11): 1497–1518.e11, 2021.

I. Bahar, T. R. Lezon, A. Bakan, and I. H. Shrivastava. Normal Mode Analysis of Biomolecular Structures: Functional Mechanisms of Membrane Proteins. *Chemical Reviews*, 110(3):1463–1497, 2010.

J. A. Bauer, J. Pavlović, and V. Bauerová-Hlinková. Normal Mode Analysis as a Routine Part of a Structural Investigation. *Molecules*, 24(18):3293, 2019.

B. Berger, M. S. Waterman, and Y. W. Yu. Levenshtein Distance, Sequence Comparison and Biological Database Search. *IEEE Transactions on Information Theory*, 67(6):3287–3294, 2021.

H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

K. Best, T. Oakes, J. M. Heather, J. Shawe-Taylor, and B. Chain. Computational analysis of stochastic heterogeneity in PCR amplification efficiency revealed by single molecule barcoding. *Scientific Reports*, 5(11):14629, 2015.

S. J. Blevins, B. G. Pierce, N. K. Singh, T. P. Riley, Y. Wang, T. T. Spear, M. I. Nishimura, Z. Weng, and B. M. Baker. How structural adaptability exists alongside HLA-A2 bias in the human $A\beta$ TCR repertoire. *Proceedings of the National Academy of Sciences of the United States of America*, 113(9):E1276–E1285, 2016.

D. A. Bolotin, S. Poslavsky, I. Mitrophanov, M. Shugay, I. Z. Mamedov, E. V. Putintseva, and D. M. Chudakov. MiXCR: Software for comprehensive adaptive immunity profiling. *Nature Methods*, 12(5):380–381, 2015.

R. Bonneau and D. Baker. Ab Initio Protein Structure Prediction: Progress and Prospects. *Annual Review of Biophysics and Biomolecular Structure*, 30(1):173–189, 2001.

T. Borrman, J. Cimons, M. Cosiano, M. Purcaro, B. G. Pierce, B. M. Baker, and Z. Weng. ATLAS: A database linking binding affinities with structures for wild-type and mutant TCR-pMHC complexes. *Proteins*, 85(5):908–916, 2017.

M. D. Brazeau and M. Friedman. The origin and early phylogenetic history of jawed vertebrates. *Nature*, 520(7548):490–497, 2015.

R. Brenke, D. R. Hall, G.-Y. Chuang, S. R. Comeau, T. Bohnuud, D. Beglov, O. Schueler-Furman, S. Vajda, and D. Kozakov. Application of asymmetric statistical potentials to antibody–protein docking. *Bioinformatics*, 28(20):2608–2614, 2012.

B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4(2):187–217, 1983.

K. Buchmann. Evolution of Innate Immunity: Clues from Invertebrates via Fish to Mammals. *Frontiers in Immunology*, 5:459, 2014.

D. Burnside, A. Schoenrock, H. Moteshareie, M. Hooshyar, P. Basra, M. Hajikarimlou, K. Dick, B. Barnes, T. Kazmirchuk, M. Jessulat, S. Pitre, B. Samanfar, M. Babu, J. R. Green, A. Wong, F. Dehne, K. K. Biggar, and A. Golshani. In Silico Engineering of Synthetic Binding Proteins from Random Amino Acid Sequences. *iScience*, 11:375–387, 2019.

S. R. Burrows, J. Rossjohn, and J. McCluskey. Have we cut ourselves too short in mapping CTL epitopes? *Trends in Immunology*, 27(1):11–16, 2006.

L. J. Carreño, S. M. Bueno, P. Bull, S. G. Nathenson, and A. M. Kalergis. The half-life of the T-cell receptor/peptide–major histocompatibility complex interaction can modulate T-cell activation in response to bacterial challenge. *Immunology*, 121(2):227–237, 2007.

D. A. Case. Normal mode analysis of protein dynamics. *Current Opinion in Structural Biology*, 4(2):285–290, 1994.

D. D. Chaplin. Overview of the Immune Response. *The Journal of allergy and clinical immunology*, 125(2 Suppl 2):S3–23, 2010.

S. Chaudhury, M. Berrondo, B. D. Weitzner, P. Muthu, H. Bergman, and J. J. Gray. Benchmarking and Analysis of Protein Docking Performance in Rosetta v3.2. *PLOS ONE*, 6(8):e22477, 2011.

J.-L. Chen, G. Stewart-Jones, G. Bossi, N. M. Lissin, L. Wooldridge, E. M. L. Choi, G. Held, P. R. Dunbar, R. M. Esnouf, M. Sami, J. M. Boulter, P. Rizkallah, C. Renner, A. Sewell, P. A. van der Merwe, B. K. Jakobsen, G. Griffiths, E. Y. Jones, and V. Cerundolo. Structural and kinetic basis for heightened immunogenicity of T cell vaccines. *Journal of Experimental Medicine*, 201(8):1243–1255, 2005.

R. Chen and Z. Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins*, 51(3):397–408, 2003.

R. Chen, J. Mintseris, J. Janin, and Z. Weng. A protein-protein docking benchmark. *Proteins*, 52(1):88–91, 2003.

S.-Y. Chen, C.-J. Liu, Q. Zhang, and A.-Y. Guo. An ultra-sensitive T-cell receptor detection method for TCR-Seq and RNA-Seq data. *Bioinformatics*, 36(15):4255–4262, 2020.

M. H. Cheng, S. Zhang, R. A. Porritt, M. N. Rivas, L. Paschold, E. Willscher, M. Binder, M. Arditi, and I. Bahar. Superantigenic character of an insert unique to SARS-CoV-2 spike supported by skewed TCR repertoire in patients with hyperinflammation. *Proceedings of the National Academy of Sciences of the United States of America*, 117(41):25254–25262, 2020.

T. M.-K. Cheng, T. L. Blundell, and J. Fernandez-Recio. pyDock: Electrostatics and desolvation for effective scoring of rigid-body protein–protein docking. *Proteins*, 68(2):503–515, 2007.

R. A. Chica, N. Doucet, and J. N. Pelletier. Semi-rational approaches to engineering enzyme activity: Combining the benefits of directed evolution and rational design. *Current Opinion in Biotechnology*, 16(4):378–384, 2005.

Y. Choi and C. M. Deane. FREAD revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, 78(6):1431–1440, 2010.

P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

S. R. Comeau, D. W. Gatchell, S. Vajda, and C. J. Camacho. ClusPro: A fully automated algorithm for protein–protein docking. *Nucleic Acids Research*, 32 (Web Server issue):W96–W99, 2004.

M. Correia-Neves, C. Waltzinger, D. Mathis, and C. Benoist. The Shaping of the T Cell Repertoire. *Immunity*, 14(1):21–32, 2001.

M. Costello, M. Fleharty, J. Abreu, Y. Farjoun, S. Ferriera, L. Holmes, B. Granger, L. Green, T. Howd, T. Mason, G. Vicente, M. Dasilva, W. Brodeur, T. DeSmet, S. Dodge, N. J. Lennon, and S. Gabriel. Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms. *BMC Genomics*, 19(1):332, 2018.

S. Daberdaku and C. Ferrari. Antibody interface prediction with 3D Zernike descriptors and SVM. *Bioinformatics*, 35(11):1870–1876, 2019.

M. De Simone, G. Rossetti, and M. Pagani. Single cell T cell receptor sequencing: Techniques and future challenges. *Frontiers in Immunology*, 9:1638, 2018.

S. J. de Vries and M. Zacharias. ATTRACT-EM: A New Method for the Computational Assembly of Large Molecular Machines Using Cryo-EM Maps. *PLOS ONE*, 7(12):e49733, 2012.

S. J. de Vries, A. D. J. van Dijk, M. Krzeminski, M. van Dijk, A. Thureau, V. Hsu, T. Wassenaar, and A. M. J. J. Bonvin. HADDOCK versus HADDOCK: New features and performance of HADDOCK2.0 on the CAPRI targets. *Proteins*, 69 (4):726–733, 2007.

S. J. de Vries, M. van Dijk, and A. M. J. J. Bonvin. The HADDOCK web server for data-driven biomolecular docking. *Nature Protocols*, 5(5):883–897, 2010.

A. Deac, P. Veličković, and P. Sormanni. Attentive Cross-Modal Paratope Prediction. *Journal of Computational Biology*, 26(6):536–545, 2019.

K. DeJong. Parameter Setting in EAs: A 30 Year Perspective. In *Parameter Setting in Evolutionary Algorithms*, 2007.

C. Dominguez, R. Boelens, and A. M. J. J. Bonvin. HADDOCK: A Protein-Protein Docking Approach Based on Biochemical or Biophysical Information. *Journal of the American Chemical Society*, 125(7):1731–1737, 2003.

P. Doruker, A. R. Atilgan, and I. Bahar. Dynamics of proteins predicted by molecular dynamics simulations and analytical approaches: Application to α-amylase inhibitor. *Proteins*, 40(3):512–524, 2000.

J. Dunbar and C. M. Deane. ANARCI: Antigen receptor numbering and receptor classification. *Bioinformatics*, 32(2):298–300, 2016.

J. Dunbar, B. Knapp, A. Fuchs, J. Shi, and C. M. Deane. Examining Variable Domain Orientations in Antigen Receptors Gives Insight into TCR-Like Antibody Design. *PLOS Computational Biology*, 10(9):e1003852, 2014.

S. M. Dunn, P. J. Rizkallah, E. Baston, T. Mahon, B. Cameron, R. Moysey, F. Gao, M. Sami, J. Boulter, Y. Li, and B. K. Jakobsen. Directed evolution of human

T cell receptor CDR2 residues by phage display dramatically enhances affinity for cognate peptide-MHC without increasing apparent cross-reactivity. *Protein Science*, 15(4):710–721, 2006.

S. Ebrahimi, H. Mohabatkar, and M. Behbahani. Predicting Promiscuous T Cell Epitopes for Designing a Vaccine Against Streptococcus pyogenes. *Applied biochemistry and biotechnology*, 187(1):90–100, 2019.

R. Evans, M. O'Neill, A. Pritzel, N. Antropova, A. Senior, T. Green, A. Žídek, R. Bates, S. Blackwell, J. Yim, O. Ronneberger, S. Bodenstein, M. Zielinski, A. Bridgland, A. Potapenko, A. Cowie, K. Tunyasuvunakool, R. Jain, E. Clancy, P. Kohli, J. Jumper, and D. Hassabis. Protein complex prediction with AlphaFold-Multimer. *bioRxiv*, 463034, 2021.

J. M. Faint, D. Pilling, A. N. Akbar, G. D. Kitas, P. A. Bacon, and M. Salmon. Quantitative flow cytometry for the analysis of T cell receptor Vβ chain expression. *Journal of Immunological Methods*, 225(1):53–60, 1999.

R. Farouni, H. Djambazian, L. E. Ferri, J. Ragoussis, and H. S. Najafabadi. Model-based analysis of sample index hopping reveals its widespread artifacts in multiplexed single-cell RNA-sequencing. *Nature Communications*, 11(1):2704, 2020.

M. L. Fernández-Quintero, N. D. Pomarici, J. R. Loeffler, C. A. Seidler, and K. R. Liedl. T-Cell Receptor CDR3 Loop Conformations in Solution Shift the Relative Vα-Vβ Domain Distributions. *Frontiers in Immunology*, 11:1440, 2020.

A. R. Fersht. From the first protein structures to our current knowledge of protein folding: Delights and scepticisms. *Nature Reviews Molecular Cell Biology*, 9(8): 650–654, 2008.

D. Fischer, O. Bachar, R. Nussinov, and H. Wolfson. An efficient automated computer vision based technique for detection of three dimensional structural motifs in proteins. *Journal of Biomolecular Structure and Dynamics*, 9(4):769–789, 1992.

J. P. Fisher, M. Yan, J. Heuijerjens, L. Carter, A. Abolhassani, J. Frosch, R. Wallace, B. Flutter, A. Capsomidis, M. Hubank, N. Klein, R. Callard, K. Gustafsson, and J. Anderson. Neuroblastoma killing properties of Vδ2 and Vδ2-negative $\gamma\delta$T cells following expansion by artificial antigen presenting cells. *Clinical Cancer Research*, 20(22):5720–5732, 2014.

C. D. Fjell, H. Jenssen, W. A. Cheung, R. E. W. Hancock, and A. Cherkasov. Optimization of Antibacterial Peptides by Genetic Algorithms and Cheminformatics. *Chemical Biology & Drug Design*, 77(1):48–56, 2011.

B. Fjukstad and L. A. Bongo. A review of scalable bioinformatics pipelines. *Data Science and Engineering*, 2(3):245–251, 2017.

M. F. Flajnik and M. Kasahara. Origin and evolution of the adaptive immune system: Genetic events and selective pressures. *Nature Reviews Genetics*, 11(1): 47–59, 2010.

D. R. Flower, K. Phadwal, I. K. Macdonald, P. V. Coveney, M. N. Davies, and S. Wan. T-cell epitope prediction and immune complex simulation using molecular dynamics: State of the art and persisting challenges. *Immunome Research*, 6 (Suppl 2):S4, 2010.

F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13: 2171–2175, 2012.

C. Fozza, F. Barraqueddu, G. Corda, S. Contini, P. Virdis, F. Dore, S. Bonfigli, and M. Longinotti. Study of the T-cell receptor repertoire by CDR3 spectratyping. *Journal of Immunological Methods*, 440:1–11, 2017.

H. A. Gabb, R. M. Jackson, and M. J. E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106–120, 1997.

E. J. Gardiner, P. Willett, and P. J. Artymiuk. Protein docking using a genetic algorithm. *Proteins*, 44(1):44–56, 2001.

V. Giudicelli, D. Chaume, and M.-P. Lefranc. IMGT/GENE-DB: a comprehensive database for human and mouse immunoglobulin and T cell receptor genes. *Nucleic Acids Research*, 33(suppl_1):D256–D261, 2005.

A. S. Gkazi, B. K. Margetts, T. Attenborough, L. Mhaldien, J. F. Standing, T. Oakes, J. M. Heather, J. Booth, M. Pasquet, R. Chiesa, P. Veys, N. Klein, B. Chain, R. Callard, and S. P. Adams. Clinical T Cell Receptor Repertoire Deep Sequencing and Analysis: An Application to Monitor Immune Reconstitution Following Cord Blood Transplantation. *Frontiers in Immunology*, 9:2547, 2018.

G. Glusman, L. Rowen, I. Lee, C. Boysen, J. C. Roach, A. F. A. Smit, K. Wang, B. F. Koop, and L. Hood. Comparative Genomics of the Human and Mouse T Cell Receptor Loci. *Immunity*, 15(3):337–349, 2001.

R. Gowthaman and B. G. Pierce. TCRmodel: High resolution modeling of T cell receptors from sequence. *Nucleic Acids Research*, 46(W1):W396–W401, 2018.

R. Gowthaman and B. G. Pierce. TCR3d: The T cell receptor structural repertoire database. *Bioinformatics*, 35(24):5323–5325, 2019.

S. Gras, J. Chadderton, C. M. Del Campo, C. Farenc, F. Wiede, T. M. Josephs, X. Y. X. Sng, M. Mirams, K. A. Watson, T. Tiganis, K. M. Quinn, J. Rossjohn, and N. L. La Gruta. Reversed T Cell Receptor Docking on a Major Histocompatibility Class I Complex Limits Involvement in the Immune Response. *Immunity*, 45(4):749–760, 2016.

J.-t. Guo, K. Ellrott, and Y. Xu. A Historical Perspective of Template-Based Protein Structure Prediction. In *Protein Structure Prediction*, Methods in Molecular Biology, pages 3–42. Humana Press, Totowa, NJ, 2008.

J. N. Haidar, B. Pierce, Y. Yu, W. Tong, M. Li, and Z. Weng. Structure-Based

Design of a T Cell Receptor Leads to Nearly 100-Fold Improvement in Binding Affinity for pepMHC. *Proteins*, 74(4):948–960, 2009.

T. Hameduh, Y. Haddad, V. Adam, and Z. Heger. Homology modeling in the time of collective and artificial intelligence. *Computational and Structural Biotechnology Journal*, 18:3494–3506, 2020.

T. Hamelryck and B. Manderick. PDB file parser and structure class implemented in Python. *Bioinformatics*, 19(17):2308–2310, 2003.

J. M. Heather and B. Chain. The sequence of sequencers: The history of sequencing DNA. *Genomics*, 107(1):1–8, 2016.

J. M. Heather, K. Best, T. Oakes, E. R. Gray, J. K. Roe, N. Thomas, N. Friedman, M. Noursadeghi, and B. Chain. Dynamic Perturbations of the T-Cell Receptor Repertoire in Chronic HIV Infection and following Antiretroviral Therapy. *Frontiers in Immunology*, 6:644, 2016.

T. Hoffmann, A. Marion, and I. Antes. DynaDom: Structure-based prediction of T cell receptor inter-domain and T cell receptor-peptide-MHC (class I) association angles. *BMC Structural Biology*, 17(1):2, 2017.

J. H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–73, 1992.

P. D. Holler, P. O. Holman, E. V. Shusta, S. O'Herrin, K. D. Wittrup, and D. M. Kranz. In vitro evolution of a T cell receptor with high affinity for peptide/MHC. *Proceedings of the National Academy of Sciences of the United States of America*, 97(10):5387–5392, 2000.

W. Huang, L. Li, J. R. Myers, and G. T. Marth. ART: A next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012.

J. Hughes, S. Rees, S. Kalindjian, and K. Philpott. Principles of early drug discovery. *British Journal of Pharmacology*, 162(6):1239–1249, 2011.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

B. Hwang, J. H. Lee, and D. Bang. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & Molecular Medicine*, 50(8):1–14, 2018.

H. Hwang, T. Vreven, J. Janin, and Z. Weng. Protein-Protein Docking Benchmark Version 4.0. *Proteins*, 78(15):3111–3114, 2010.

M. Jamal-Hanjani, G. A. Wilson, N. McGranahan, N. J. Birkbak, T. B. Watkins, S. Veeriah, S. Shafi, D. H. Johnson, R. Mitter, R. Rosenthal, M. Salm, S. Horswell, M. Escudero, N. Matthews, A. Rowan, T. Chambers, D. A. Moore, S. Turajlic, H. Xu, S.-M. Lee, M. D. Forster, T. Ahmad, C. T. Hiley, C. Abbosh, M. Falzon, E. Borg, T. Marafioti, D. Lawrence, M. Hayward, S. Kolvekar, N. Panagiotopoulos, S. M. Janes, R. Thakrar, A. Ahmed, F. Blackhall, Y. Summers, R. Shah, L. Joseph, A. M. Quinn, P. A. Crosbie, B. Naidu, G. Middleton, G. Langman, S. Trotter, M. Nicolson, H. Remmen, K. Kerr, M. Chetty, L. Gomersall, D. A. Fennell, A. Nakas, S. Rathinam, G. Anand, S. Khan, P. Russell, V. Ezhil, B. Ismail, M. Irvin-Sellers, V. Prakash, J. F. Lester, M. Kornaszewska, R. Attanoos, H. Adams, H. Davies, S. Dentro, P. Taniere, B. O'Sullivan, H. L. Lowe, J. A. Hartley, N. Iles, H. Bell, Y. Ngai, J. A. Shaw, J. Herrero, Z. Szallasi, R. F. Schwarz, A. Stewart, S. A. Quezada, J. Le Quesne, P. Van Loo, C. Dive, A. Hackshaw, C. Swanton, and T. Consortium. Tracking the Evolution of Non–Small-Cell Lung Cancer. *New England Journal of Medicine*, 376(22):2109–2121, 2017.

J. Janin, K. Henrick, J. Moult, L. T. Eyck, M. J. E. Sternberg, S. Vajda, I. Vakser, and S. J. Wodak. CAPRI: A Critical Assessment of PRedicted Interactions. *Proteins*, 52(1):2–9, 2003.

K. K. Jensen, V. Rantos, E. C. Jappe, T. H. Olsen, M. C. Jespersen, V. Jurtz, L. E. Jessen, E. Lanzarotti, S. Mahajan, B. Peters, M. Nielsen, and P. Marcatili.

TCRpMHCmodels: Structural modelling of TCR-pMHC class I complexes. *Scientific Reports*, 9(1):14530, 2019.

B. Jiménez-García, J. Roel-Touris, M. Romero-Durana, M. Vidal, D. Jiménez-González, and J. Fernández-Recio. LightDock: A new multi-scale approach to protein–protein docking. *Bioinformatics*, 34(1):49–55, 2018.

K. Joshi, M. R. de Massy, M. Ismail, J. L. Reading, I. Uddin, A. Woolston, E. Hatipoglu, T. Oakes, R. Rosenthal, T. Peacock, T. Ronel, M. Noursadeghi, V. Turati, A. J. S. Furness, A. Georgiou, Y. N. S. Wong, A. Ben Aissa, M. W. Sunderland, M. Jamal-Hanjani, S. Veeriah, N. J. Birkbak, G. A. Wilson, C. T. Hiley, E. Ghorani, J. A. Guerra-Assunção, J. Herrero, T. Enver, S. R. Hadrup, A. Hackshaw, K. S. Peggs, N. McGranahan, C. Swanton, T. consortium, S. A. Quezada, and B. Chain. Spatial heterogeneity of the T cell receptor repertoire reflects the mutational landscape in lung cancer. *Nature Medicine*, 25(1010):1549–1559, 2019.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

E. Karaca and A. M. J. J. Bonvin. On the usefulness of ion-mobility mass spectrometry and SAXS data in scoring docking decoys. *Acta Crystallographica Section D Biological Crystallography*, 69:683–694, 2013.

P. L. Kastritis, J. P. G. L. M. Rodrigues, G. E. Folkers, R. Boelens, and A. M. J. J. Bonvin. Proteins feel more than they see: Fine-tuning of binding affinity by properties of the non-interacting surface. *Journal of Molecular Biology*, 426(14): 2632–2652, 2014.

E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A. A. Friesem, C. Aflalo, and I. A.

Vakser. Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, 1992.

Z. Kato, J. N. H. Stern, H. K. Nakamura, N. Miyashita, K. Kuwata, N. Kondo, and J. L. Strominger. The autoimmune TCR-Ob.2F3 can bind to MBP85–99/HLA-DR2 having an unconventional mode as in TCR-Ob.1A12. *Molecular Immunology*, 48(1):314–320, 2010.

M. S. Klausen, M. V. Anderson, M. C. Jespersen, M. Nielsen, and P. Marcatili. LYRA, a webserver for lymphocyte receptor structural modeling. *Nucleic Acids Research*, 43(W1):W349–W355, 2015.

B. Knapp, U. Omasits, and W. Schreiner. Side chain substitution benchmark for peptide/MHC interaction. *Protein Science*, 17(6):977–982, 2008.

B. Knapp, V. Giczi, R. Ribarics, and W. Schreiner. PeptX: Using genetic algorithms to optimize peptides for MHC binding. *BMC bioinformatics*, 12(1):241, 2011.

B. Knapp, S. Demharter, R. Esmaielbeiki, and C. M. Deane. Current status and future challenges in T-cell receptor/peptide/MHC molecular dynamics simulations. *Briefings in Bioinformatics*, 16(6):1035–1044, 2015.

R. A. Koup and D. C. Douek. Vaccine Design for CD8 T Lymphocyte Responses. *Cold Spring Harbor Perspectives in Medicine*, 1(1):a007252, 2011.

D. Kozakov, D. R. Hall, B. Xia, K. A. Porter, D. Padhorny, C. Yueh, D. Beglov, and S. Vajda. The ClusPro web server for protein–protein docking. *Nature Protocols*, 12(2):255–278, 2017.

K. Krawczyk, T. Baker, J. Shi, and C. M. Deane. Antibody i-Patch prediction of the antibody binding site improves rigid local antibody–antigen docking. *Protein Engineering, Design and Selection*, 26(10):621–629, 2013.

E. Krieger, S. B. Nabuurs, and G. Vriend. Homology Modeling. In *Structural Bioinformatics*, pages 509–523. John Wiley & Sons, Ltd, 2003.

K. N. Krishnanand and D. Ghose. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence*, 3 (2):87–124, 2009.

G. G. Krivov, M. V. Shapovalov, and R. L. Dunbrack. Improved prediction of protein side-chain conformations with SCWRL4. *Proteins*, 77(4):778–795, 2009.

V. Kunik, S. Ashkenazi, and Y. Ofran. Paratome: An online tool for systematic identification of antigen-binding regions in antibodies based on sequence or structure. *Nucleic Acids Research*, 40(W1):W521–W524, 2012.

D. Kuroda, H. Shirai, M. P. Jacobson, and H. Nakamura. Computer-aided antibody design. *Protein Engineering, Design and Selection*, 25(10):507–522, 2012.

B. Lamoree and R. E. Hubbard. Current perspectives in fragment-based lead discovery (FBLD). *Essays in Biochemistry*, 61(5):453–464, 2017.

A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. Kaufman, P. D. Renfrew, C. A. Smith, W. Sheffler, I. W. Davis, S. Cooper, A. Treuille, D. J. Mandell, F. Richter, Y.-E. A. Ban, S. J. Fleishman, J. E. Corn, D. E. Kim, S. Lyskov, M. Berrondo, S. Mentzer, Z. Popović, J. J. Havranek, J. Karanicolas, R. Das, J. Meiler, T. Kortemme, J. J. Gray, B. Kuhlman, D. Baker, and P. Bradley. Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. *Methods in Enzymology*, 487:545–574, 2011.

L. Lee, N. Alrasheed, G. Khandelwal, E. Fitzsimons, H. Richards, W. Wilson, S. J. Chavda, J. Henry, L. Conde, M. R. De Massy, M. Chin, D. Galas-Filipowicz, J. Herrero, B. Chain, S. A. Quezada, and K. Yong. Increased Immune-Regulatory Receptor Expression on Effector T Cells as Early Indicators of Relapse Following Autologous Stem Cell Transplantation for Multiple Myeloma. *Frontiers in Immunology*, 12:618610, 2021.

J. Leem, S. H. P. de Oliveira, K. Krawczyk, and C. M. Deane. STCRDab: The structural T-cell receptor database. *Nucleic Acids Research*, 46(D1):D406–D412, 2018.

A. Leimgruber, M. Ferber, M. Irving, H. Hussain-Kahn, S. Wieckowski, L. Derré, N. Rufer, V. Zoete, and O. Michielin. TCRep 3D: An Automated In Silico Approach to Study the Structural Properties of TCR Repertoires. *PLOS ONE*, 6(10): e26301, 2011.

M. Levitt and A. Warshel. Computer simulation of protein folding. *Nature*, 253 (5494):694–698, 1975.

M. Levitt, M. Hirshberg, R. Sharon, and V. Daggett. Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Computer Physics Communications*, 91(1):215–231, 1995.

G.-B. Li, S. Ji, L.-L. Yang, R.-J. Zhang, K. Chen, L. Zhong, S. Ma, and S.-Y. Yang. LEADOPT: An automatic tool for structure-based lead optimization, and its application in structural optimizations of VEGFR2 and SYK inhibitors. *European Journal of Medicinal Chemistry*, 93:523–538, 2015.

H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

S. Li, J. Wilamowski, S. Teraguchi, F. J. van Eerden, J. Rozewicki, A. Davila, Z. Xu, K. Katoh, and D. M. Standley. Structural Modeling of Lymphocyte Receptors and Their Antigens. In *In Vitro Differentiation of T-Cells: Methods and Protocols*, pages 207–229. Springer, New York, NY, 2019.

Y. Li, R. Moysey, P. E. Molloy, A.-L. Vuidepot, T. Mahon, E. Baston, S. Dunn, N. Liddy, J. Jacob, B. K. Jakobsen, and J. M. Boulter. Directed evolution of human T-cell receptors with picomolar affinities by phage display. *Nature Biotechnology*, 23(3):349–354, 2005.

E. Liberis, P. Veličković, P. Sormanni, M. Vendruscolo, and P. Liò. Parapred: Antibody paratope prediction using convolutional and recurrent neural networks. *Bioinformatics*, 34(17):2944–2950, 2018.

G. P. Linette, E. A. Stadtmauer, M. V. Maus, A. P. Rapoport, B. L. Levine, L. Emery, L. Litzky, A. Bagg, B. M. Carreno, and P. J. Cimino. Cardiovascular toxicity and titin cross-reactivity of affinity-enhanced T cells in myeloma and melanoma. *Blood*, 122(6):863–871, 2013.

J. Ma. Usefulness and limitations of normal mode analysis in modeling dynamics of biomolecular complexes. *Structure*, 13(3):373–380, 2005.

L. E. MacConaill, R. T. Burns, A. Nag, H. A. Coleman, M. K. Slevin, K. Giorda, M. Light, K. Lai, M. Jarosz, M. S. McNeill, M. D. Ducar, M. Meyerson, and A. R. Thorner. Unique, dual-indexed sequencing adapters with UMIs effectively eliminate index cross-talk and significantly improve sensitivity of massively parallel sequencing. *BMC Genomics*, 19(1):30, 2018.

G. Macindoe, L. Mavridis, V. Venkatraman, M.-D. Devignes, and D. W. Ritchie. HexServer: An FFT-based protein docking server powered by graphics processors. *Nucleic Acids Research*, 38(Web Server issue):W445–W449, 2010.

I. Z. Mamedov, O. V. Britanova, I. V. Zvyagin, M. A. Turchaninova, D. A. Bolotin, E. V. Putintseva, Y. B. Lebedev, and D. M. Chudakov. Preparing unbiased T-cell receptor and antibody cDNA libraries for the deep next generation sequencing profiling. *Frontiers in Immunology*, 4:456, 2013.

J. N. Mandl and R. N. Germain. Focusing in on T Cell Cross-Reactivity. *Cell*, 157 (5):1006–1008, 2014.

Q. Marcou, T. Mora, and A. M. Walczak. High-throughput immune repertoire analysis with IGoR. *Nature Communications*, 9(1):561, 2018.

F. Markowetz. All biology is computational biology. *PLOS Biology*, 15(3): e2002050, 2017.

C. Marks, J. Nowak, S. Klostermann, G. Georges, J. Dunbar, J. Shi, S. Kelm, and C. M. Deane. Sphinx: Merging knowledge-based and ab initio approaches to improve protein loop prediction. *Bioinformatics*, 33(9):1346–1353, 2017.

J. S. Marshall, R. Warrington, W. Watson, and H. L. Kim. An introduction to immunology and immunopathology. *Allergy, Asthma & Clinical Immunology*, 14(Suppl 2):49, 2018.

E. Mashiach, D. Schneidman-Duhovny, A. Peri, Y. Shavit, R. Nussinov, and H. J. Wolfson. An Integrated Suite of Fast Docking Algorithms. *Proteins*, 78(15): 3197–3204, 2010.

A. May and M. Zacharias. Energy minimization in low-frequency normal modes to efficiently allow for global flexibility during systematic protein–protein docking. *Proteins*, 70(3):794–809, 2008.

C. McBeth, A. Seamons, J. C. Pizarro, S. J. Fleishman, D. Baker, T. Kortemme, J. M. Goverman, and R. K. Strong. A new twist in TCR diversity revealed by a forbidden $\alpha\beta$ TCR. *Journal of Molecular Biology*, 375(5):1306–1319, 2008.

S. McGinn and I. G. Gut. DNA sequencing – spanning the generations. *New Biotechnology*, 30(4):366–372, 2013.

T. W. McKeithan. Kinetic proofreading in T-cell receptor signal transduction. *Proceedings of the National Academy of Sciences of the United States of America*, 92(11):5042–5046, 1995.

R. Méndez, R. Leplae, L. D. Maria, and S. J. Wodak. Assessment of blind predictions of protein-protein interactions: Current status of docking methods. *Proteins*, 52(1):51–67, 2003.

X.-Y. Meng, H.-X. Zhang, M. Mezei, and M. Cui. Molecular Docking: A powerful approach for structure-based drug discovery. *Current Computer-Aided Drug Design*, 7(2):146–157, 2011.

M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1996.

I. H. Moal and P. A. Bates. SwarmDock and the Use of Normal Modes in Protein-Protein Docking. *International Journal of Molecular Sciences*, 11(10):3623–3648, 2010.

J. H. Moore, F. W. Asselbergs, and S. M. Williams. Bioinformatics challenges for genome-wide association studies. *Bioinformatics*, 26(4):445–455, 2010.

T. Mora and A. M. Walczak. Quantifying lymphocyte receptor diversity. In *Systems Immunology*, pages 183–198. CRC Press, 2019.

L. E. Mose, S. R. Selitsky, L. M. Bixby, D. L. Marron, M. D. Iglesia, J. S. Serody, C. M. Perou, B. G. Vincent, and J. S. Parker. Assembly-based inference of B-cell receptor repertoires from short read RNA sequencing data with V'DJer. *Bioinformatics*, 32(24):3729–3734, 2016.

J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano. Critical Assessment of Methods of Protein Structure Prediction (CASP) – Round XII. *Proteins*, 86(Suppl 1):7–15, 2018.

M. T. Muhammed and E. Aki-Yalcin. Homology modeling in drug discovery: Overview, current applications, and future perspectives. *Chemical Biology & Drug Design*, 93(1):12–20, 2019.

K. Murphy, P. Travers, M. Walport, and C. Janeway. *Janeway's Immunobiology*. Garland Science, New York, seventh edition, 2008.

G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

T. Oakes, J. M. Heather, K. Best, R. Byng-Maddick, C. Husovsky, M. Ismail, K. Joshi, G. Maxwell, M. Noursadeghi, N. Riddell, T. Ruehl, C. T. Turner, I. Uddin, and B. Chain. Quantitative characterization of the T cell receptor repertoire of naïve and memory subsets using an integrated experimental and computational pipeline which is robust, economical, and versatile. *Frontiers in Immunology*, 8: 1267, 2017.

R. Oliva, A. Vangone, and L. Cavallo. Ranking multiple docking solutions based on the conservation of inter-residue contacts. *Proteins*, 81(9):1571–1584, 2013.

Y. Ouchi, A. Patil, Y. Tamura, H. Nishimasu, A. Negishi, S. K. Paul, N. Takemura, T. Satoh, Y. Kimura, M. Kurachi, O. Nureki, K. Nakai, H. Kiyono, and S. Uematsu. Generation of tumor antigen-specific murine CD8+ T cells with enhanced anti-tumor activity via highly efficient CRISPR/Cas9 genome editing. *International Immunology*, 30(4):141–154, 2018.

N. S. Pagadala, K. Syed, and J. Tuszynski. Software for molecular docking: A review. *Biophysical Reviews*, 9(2):91–102, 2017.

T. Pantsar and A. Poso. Binding Affinity via Docking: Fact and Fiction. *Molecules*, 23(8):1899, 2018.

T. Peacock and B. Chain. Information-driven docking for TCR-pMHC complex prediction. *Frontiers in Immunology*, 12:1952, 2021.

T. Peacock, J. M. Heather, T. Ronel, and B. Chain. Decombinator V4 – an improved AIRR-compliant software package for T cell receptor sequence annotation. *Bioinformatics*, 37(6):876–878, 2021.

N. D. Pennock, J. T. White, E. W. Cross, E. E. Cheney, B. A. Tamburini, and R. M. Kedl. T cell responses: Naïve to memory and everything in between. *Advances in Physiology Education*, 37(4):273–283, 2013.

S. Pérez and I. Tvaroška. Carbohydrate–Protein Interactions: Molecular Modeling Insights. *Advances in Carbohydrate Chemistry and Biochemistry*, 71:9–136, 2014.

B. G. Pierce and Z. Weng. A flexible docking approach for prediction of T cell receptor-peptide-MHC complexes. *Protein Science*, 22(1):35–46, 2013.

B. G. Pierce, L. M. Hellman, M. Hossain, N. K. Singh, C. W. Vander Kooi, Z. Weng, and B. M. Baker. Computational Design of the Affinity and Specificity of a

Therapeutic T Cell Receptor. *PLOS Computational Biology*, 10(2):e1003478, 2014a.

B. G. Pierce, K. Wiehe, H. Hwang, B.-H. Kim, T. Vreven, and Z. Weng. ZDOCK server: Interactive docking prediction of protein–protein complexes and symmetric multimers. *Bioinformatics*, 30(12):1771–1773, 2014b.

C. Pons, M. D'Abramo, D. I. Svergun, M. Orozco, P. Bernadó, and J. Fernández-Recio. Structural Characterization of Protein–Protein Complexes by Integrating Computational Docking with Small-angle Scattering Data. *Journal of Molecular Biology*, 403(2):217–230, 2010.

K. A. Porter, I. Desta, D. Kozakov, and S. Vajda. What method to use for protein–protein docking? *Current Opinion in Structural Biology*, 55:1–7, 2019.

Protein Data Bank. Crystallography: Protein Data Bank. *Nature New Biology*, 233 (42):223–223, 1971.

J. D. Rabinowitz, C. Beeson, D. S. Lyons, M. M. Davis, and H. M. McConnell. Kinetic discrimination in T-cell activation. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1401–1405, 1996.

W. Ratajczak, P. Niedźwiedzka-Rystwej, B. Tokarz-Deptuła, and W. Deptuła. Immunological memory cells. *Central European Journal of Immunology*, 43(2): 194–203, 2018.

D. W. Ritchie and G. J. L. Kemp. Protein docking using spherical polar Fourier correlations. *Proteins*, 39(2):178–194, 2000.

D. W. Ritchie and V. Venkatraman. Ultra-fast FFT protein docking on graphics processors. *Bioinformatics*, 26(19):2398–2405, 2010.

N. Röckendorf, M. Borschbach, and A. Frey. Molecular Evolution of Peptide Ligands with Custom-Tailored Characteristics for Targeting of Glycostructures. *PLOS Computational Biology*, 8(12):e1002800, 2012.

J. P. G. L. M. Rodrigues and A. M. J. J. Bonvin. Integrative computational modeling of protein interactions. *The FEBS Journal*, 281(8):1988–2003, 2014.

J. P. G. L. M. Rodrigues, M. Trellet, C. Schmitz, P. Kastritis, E. Karaca, A. S. J. Melquiond, and A. M. J. J. Bonvin. Clustering biomolecular complexes by residue contacts similarity. *Proteins*, 80(7):1810–1817, 2012.

J. Roel-Touris, A. M. J. J. Bonvin, and B. Jiménez-García. LightDock goes information-driven. *Bioinformatics*, 36(3):950–952, 2020.

Z. A. Rollins, M. B. Curtis, R. Faller, and S. C. George. Automated protein-protein structure prediction of the T cell receptor-peptide major histocompatibility complex. *bioRxiv*, 494331, 2022.

P. A. Romero and F. H. Arnold. Exploring protein fitness landscapes by directed evolution. *Nature Reviews Molecular Cell Biology*, 10(12):866–876, 2009.

M. R. Romo, D. Pérez-Martínez, and C. C. Ferrer. Innate immunity in vertebrates: An overview. *Immunology*, 148(2):125–139, 2016.

T. Ronel, M. Harries, K. Wicks, T. Oakes, H. Singleton, R. Dearman, G. Maxwell, and B. Chain. The clonal structure and dynamics of the human T cell response to an organic chemical hapten. *eLife*, 10:e54747, 2021.

M. C. D. Rosa, B. Giardina, C. Bianchi, C. C. Alinovi, D. Pirolli, G. Ferraccioli, M. D. Santis, G. D. Sante, and F. Ria. Modeling the Ternary Complex TCR-Vβ/CollagenII(261–273)/HLA-DR4 Associated with Rheumatoid Arthritis. *PLOS ONE*, 5(7):e11550, 2010.

S. A. Rosenberg, N. P. Restifo, J. C. Yang, R. A. Morgan, and M. E. Dudley. Adoptive cell transfer: A clinical path to effective cancer immunotherapy. *Nature Reviews Cancer*, 8(4):299–308, 2008.

J. Rossjohn, S. Gras, J. J. Miles, S. J. Turner, D. I. Godfrey, and J. McCluskey. T Cell Antigen Receptor Recognition of Antigen-Presenting Molecules. *Annual Review of Immunology*, 33(1):169–200, 2015.

F. Rubelt, C. E. Busse, S. A. C. Bukhari, J.-P. Bürckert, E. Mariotti-Ferrandiz, L. G. Cowell, C. T. Watson, N. Marthandan, W. J. Faison, U. Hershberg, U. Laserson, B. D. Corrie, M. M. Davis, B. Peters, M.-P. Lefranc, J. K. Scott, F. Breden, E. T. Luning Prak, and S. H. Kleinstein. Adaptive Immune Receptor Repertoire Community recommendations for sharing immune-repertoire sequencing data. *Nature Immunology*, 18(12):1274–1278, 2017.

M. G. Rudolph, R. L. Stanfield, and I. A. Wilson. How TCRs bind MHCs, peptides, and coreceptors. *Annual Review of Immunology*, 24(1):419–466, 2006.

A. Šali and T. Blundell. Comparative Protein Modelling by Satisfaction of Spatial Restraints. *Journal of Molecular Biology*, 234(3):779–815, 1993.

D. B. Sant'Angelo, B. Lucas, P. G. Waterbury, B. Cohen, T. Brabb, J. Goverman, R. N. Germain, and C. A. Janeway. A Molecular Map of T Cell Development. *Immunity*, 9(2):179–186, 1998.

D. Schneidman-Duhovny, Y. Inbar, R. Nussinov, and H. J. Wolfson. PatchDock and SymmDock: Servers for rigid and symmetric docking. *Nucleic Acids Research*, 33(Web Server issue):W363–W367, 2005.

D. Schritt, S. Li, J. Rozewicki, K. Katoh, K. Yamashita, W. Volkmuth, G. Cavet, and D. M. Standley. Repertoire Builder: High-throughput structural modeling of B and T cell receptors. *Molecular Systems Design & Engineering*, 4(4):761–768, 2019.

N. J. Schuldt and B. A. Binstadt. Dual TCR T Cells: Identity Crisis or Multitaskers? *The Journal of Immunology*, 202(3):637–644, 2019.

R. Sender, S. Fuchs, and R. Milo. Revised Estimates for the Number of Human and Bacteria Cells in the Body. *PLOS Biology*, 14(8):e1002533, 2016.

M. Shugay, O. V. Britanova, E. M. Merzlyak, M. A. Turchaninova, I. Z. Mamedov, T. R. Tuganbaev, D. A. Bolotin, D. B. Staroverov, E. V. Putintseva, K. Plevova, and et al. Towards error-free profiling of immune repertoires. *Nature Methods*, 11(66):653–655, 2014.

J. Singh, M. A. Ator, E. P. Jaeger, M. P. Allen, D. A. Whipple, J. E. Soloweij, S. Chowdhary, and A. M. Treasurywala. Application of Genetic Algorithms to Combinatorial Synthesis: A Computational Approach to Lead Identification and Lead Optimization,. *Journal of the American Chemical Society*, 118(7):1669–1676, 1996.

N. K. Singh, T. P. Riley, S. C. B. Baker, T. Borrman, Z. Weng, and B. M. Baker. Emerging concepts in T cell receptor specificity: Rationalizing and (maybe) predicting outcomes. *Journal of Immunology*, 199(7):2203–2213, 2017.

N. K. Singh, E. T. Abualrous, C. M. Ayres, F. Noé, R. Gowthaman, B. G. Pierce, and B. M. Baker. Geometrical characterization of T cell receptor binding modes reveals class-specific binding to maximize access to antigen. *Proteins*, 88(3): 503–513, 2020.

R. Sinha, G. Stanley, G. S. Gulati, C. Ezran, K. J. Travaglini, E. Wei, C. K. F. Chan, A. N. Nabhan, T. Su, R. M. Morganti, S. D. Conley, H. Chaib, K. Red-Horse, M. T. Longaker, M. P. Snyder, M. A. Krasnow, and I. L. Weissman. Index switching causes "spreading-of-signal" among multiplexed samples in Illumina HiSeq 4000 DNA sequencing. *bioRxiv*, 125724, 2017.

M. Sipper, W. Fu, K. Ahuja, and J. H. Moore. Investigating the parameter space of evolutionary algorithms. *BioData Mining*, 11(1):2, 2018.

A. Sircar, S. Chaudhury, K. P. Kilambi, M. Berrondo, and J. J. Gray. A generalized approach to sampling backbone conformations with RosettaDock for CAPRI rounds 13–19. *Proteins*, 78(15):3115–3123, 2010.

A. Six, E. Mariotti-Ferrandiz, W. Chaara, S. Magadan, H.-P. Pham, M.-P. Lefranc, T. Mora, V. Thomas-Vaslin, A. Walczak, and P. Boudinot. The Past, Present, and Future of Immune Repertoire Biology – The Rise of Next-Generation Repertoire Analysis. *Frontiers in Immunology*, 4:413, 2013.

L. Skjaerven, S. M. Hollup, and N. Reuter. Normal mode analysis for proteins. *Computational and Theoretical Chemistry*, 898(1):42–48, 2009.

G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe. Computational Methods in Drug Discovery. *Pharmacological Reviews*, 66(1):334–395, 2014.

S. Smit and A. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *2009 IEEE Congress on Evolutionary Computation*, pages 399–406, 2009.

N. C. Smith, M. L. Rise, and S. L. Christian. A Comparison of the Innate and Adaptive Immune Systems in Cartilaginous Fish, Ray-Finned Fish, and Lobe-Finned Fish. *Frontiers in Immunology*, 10:2292, 2019.

L. Song, D. Cohen, Z. Ouyang, Y. Cao, X. Hu, and X. S. Liu. TRUST4: Immune repertoire reconstruction from bulk and single-cell RNA-seq data. *Nature Methods*, 18(6):627–630, 2021.

J. O. Spiegel and J. D. Durrant. AutoGrow4: An open-source genetic algorithm for de novo drug design and lead optimization. *Journal of Cheminformatics*, 12(1): 25, 2020.

M. J. Stubbington, T. Lönnberg, V. Proserpio, S. Clare, A. O. Speak, G. Dougan, and S. A. Teichmann. T cell fate and clonality inference from single cell transcriptomes. *Nature Methods*, 13(4):329–332, 2016.

C. T. Su, C. Schönbach, and C.-K. Kwoh. Molecular docking analysis of 2009-H1N1 and 2004-H5N1 influenza virus HLA-B*4405-restricted HA epitope candidates: Implications for TCR cross-recognition and vaccine development. *BMC Bioinformatics*, 14(2):S21, 2013.

S. Teraguchi, D. S. Saputri, M. A. Llamas-Covarrubias, A. Davila, D. Diez, S. A. Nazlica, J. Rozewicki, H. S. Ismanto, J. Wilamowski, J. Xie, Z. Xu, M. d. J. Loza-Lopez, F. J. van Eerden, S. Li, and D. M. Standley. Methods for sequence and structural analysis of B and T cell receptor repertoires. *Computational and Structural Biotechnology Journal*, 18:2000–2011, 2020.

F. Tessaro and L. Scapozza. How 'Protein-Docking' Translates into the New Emerging Field of Docking Small Molecules to Nucleic Acids? *Molecules*, 25(12): 2749, 2020.

N. Thomas, J. Heather, W. Ndifon, J. Shawe-Taylor, and B. Chain. Decombinator: a tool for fast, efficient gene assignment in T-cell receptor sequences using a finite state machine. *Bioinformatics*, 29(5):542–550, 2013.

N. Thomas, K. Best, M. Cinelli, S. Reich-Zeliger, H. Gal, E. Shifrut, A. Madi, N. Friedman, J. Shawe-Taylor, and B. Chain. Tracking global changes induced in the CD4 T-cell receptor repertoire by immunization with a complex antigen using short stretches of CDR3 protein sequence. *Bioinformatics*, 30(22):3181–3188, 2014.

M. K. Tiwari, R. Singh, R. K. Singh, I.-W. Kim, and J.-K. Lee. Computational Approaches For Rational Design Of Proteins With Novel Functionalities. *Computational and Structural Biotechnology Journal*, 2(3):e201204002, 2012.

N. Tomar and R. K. De. Immunoinformatics: A Brief Review. In *Immunoinformatics*, Methods in Molecular Biology, pages 23–55. Springer, New York, NY, 2014.

A. Tovchigrechko and I. A. Vakser. GRAMM-X public web server for protein–protein docking. *Nucleic Acids Research*, 34(Web Server issue):W310–W314, 2006.

I. Uddin, A. Woolston, T. Peacock, K. Joshi, M. Ismail, T. Ronel, C. Husovsky, and B. Chain. Quantitative analysis of the T cell receptor repertoire. *Methods in Enzymology*, 629:465–492, 2019.

R. Unger. The genetic algorithm approach to protein structure prediction. *Structure and Bonding*, 110:2697–2699, 2004.

I. A. Vakser. Protein-Protein Docking: From Interaction to Interactome. *Biophysical journal*, 107(8):1785–1793, 2014.

P. A. van der Merwe and S. J. Davis. Molecular Interactions Mediating T Cell Antigen Recognition. *Annual Review of Immunology*, 21(1):659–684, 2003.

G. van Zundert, J. Rodrigues, M. Trellet, C. Schmitz, P. Kastritis, E. Karaca, A. Melquiond, M. van Dijk, S. de Vries, and A. Bonvin. The HADDOCK2.2 Web Server: User-Friendly Integrative Modeling of Biomolecular Complexes. *Journal of Molecular Biology*, 428(4):720–725, 2016.

J. A. Vander Heiden, S. Marquez, N. Marthandan, S. A. C. Bukhari, C. E. Busse, B. Corrie, U. Hershberg, S. H. Kleinstein, F. A. Matsen IV, D. K. Ralph, A. M. Rosenfeld, C. A. Schramm, T. A. C. , S. Christley, and U. Laserson. AIRR Community Standardized Representations for Annotated Immune Repertoires. *Frontiers in Immunology*, 9:2206, 2018.

A. Vangone and A. M. Bonvin. Contacts-based prediction of binding affinity in protein–protein complexes. *eLife*, 4:e07454, 2015.

K. V. Voelkerding, S. A. Dames, and J. D. Durtschi. Next-Generation Sequencing: From Basic Research to Diagnostics. *Clinical Chemistry*, 55(4):641–658, 2009.

T. Vreven, I. H. Moal, A. Vangone, B. G. Pierce, P. L. Kastritis, M. Torchala, R. Chaleil, B. Jiménez-García, P. A. Bates, J. Fernandez-Recio, A. M. J. J. Bonvin, and Z. Weng. Updates to the Integrated Protein–Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2. *Journal of Molecular Biology*, 427(19):3031–3041, 2015.

M. L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

T. A. Wassenaar, M. van Dijk, N. Loureiro-Ferreira, G. van der Schot, S. J. de Vries, C. Schmitz, J. van der Zwan, R. Boelens, A. Giachetti, L. Ferella, A. Rosato, I. Bertini, T. Herrmann, H. R. A. Jonker, A. Bagaria, V. Jaravine,

P. Güntert, H. Schwalbe, W. F. Vranken, J. F. Doreleijers, G. Vriend, G. W. Vuister, D. Franke, A. Kikhney, D. I. Svergun, R. H. Fogh, J. Ionides, E. D. Laue, C. Spronk, S. Jurkša, M. Verlato, S. Badoer, S. Dal Pra, M. Mazzucato, E. Frizziero, and A. M. J. J. Bonvin. WeNMR: Structural Biology on the Grid. *Journal of Grid Computing*, 10(4):743–767, 2012.

B. Webb and A. Sali. Comparative Protein Structure Modeling Using MODELLER. *Current Protocols in Bioinformatics*, 54:5.6.1–5.6.37, 2016.

J. A. Weinstein, N. Jiang, R. A. White, D. S. Fisher, and S. R. Quake. High-throughput sequencing of the zebrafish antibody repertoire. *Science*, 324(5928): 807–810, 2009.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, New York, 2016.

M. Wieczorek, E. T. Abualrous, J. Sticht, M. Álvaro-Benito, S. Stolzenberg, F. Noé, and C. Freund. Major Histocompatibility Complex (MHC) Class I and MHC Class II Proteins: Conformational Plasticity in Antigen Presentation. *Frontiers in Immunology*, 8:292, 2017.

W. K. Wong, J. Leem, and C. M. Deane. Comparative Analysis of the CDR Loops of Antigen Receptors. *Frontiers in Immunology*, 10:2454, 2019.

W. K. Wong, C. Marks, J. Leem, A. P. Lewis, J. Shi, and C. M. Deane. TCRBuilder: Multi-state T-cell receptor structure prediction. *Bioinformatics*, 36(11):3580–3581, 2020.

K. W. Wucherpfennig, M. J. Call, L. Deng, and R. Mariuzza. Structural Alterations in peptide–MHC Recognition by Self-reactive T cell Receptors. *Current Opinion in Immunology*, 21(6):590–595, 2009.

wwPDB consortium. Protein Data Bank: The single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, 47(D1):D520–D528, 2019.

Z. Xu, A. Davila, J. Wiamowski, S. Teraguchi, and D. M. Standley. Improved antibody-specific epitope prediction using AlphaFold and AbAdapt. *bioRxiv*, 492907, 2022.

R. Yin, B. Y. Feng, A. Varshney, and B. G. Pierce. Benchmarking AlphaFold for protein complex modeling reveals accuracy determinants. *bioRxiv*, 465575, 2021.

M. Zacharias. Protein–protein docking with a reduced protein model accounting for side-chain flexibility. *Protein Science*, 12(6):1271–1282, 2003.

V. Zarnitsyna, B. Evavold, L. Schoettle, J. Blattman, and R. Antia. Estimating the Diversity, Completeness, and Cross-Reactivity of the T Cell Repertoire. *Frontiers in Immunology*, 4:485, 2013.

Y. Zhang, X. Yang, Y. Zhang, Y. Zhang, M. Wang, J. X. Ou, Y. Zhu, H. Zeng, J. Wu, C. Lan, H.-W. Zhou, W. Yang, and Z. Zhang. Tools for fundamental analysis functions of TCR repertoires: A systematic comparison. *Briefings in Bioinformatics*, 21(5):1706–1716, 2020.

H. Zhou and Y. Zhou. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Science*, 11(11):2714–2726, 2002.

V. Zoete, M. Irving, M. Ferber, M. A. Cuendet, and O. Michielin. Structure-Based, Rational Design of T Cell Receptors. *Frontiers in Immunology*, 4:268, 2013.