immuneML: an ecosystem for machine learning analysis of adaptive immune receptor repertoires

Milena Pavlović[1,2,3,#], Lonneke Scheffer[1,2,#], Keshav Motwani[4], Chakravarthi Kanduri[2], Radmila Kompova[2], Nikolay Vazov[6], Knut Waagan[6], Fabian L. M. Bernal[6], Alexandre Almeida Costa[7], Brian Corrie[8], Rahmad Akbar[9], Ghadi S. Al Hajj[1], Gabriel Balaban[1,2], Todd M. Brusko[4, 5], Maria Chernigovskaya[9], Scott Christley[10], Lindsay G. Cowell[11], Robert Frank[9], Ivar Grytten[1,2], Sveinung Gundersen[2], Ingrid Hobæk Haff[11], Sepp Hochreiter[12,13], Eivind Hovig[1,2,14], Ping-Han Hsieh[15], Günter Klambauer[12], Marieke L. Kuijjer[15,16], Christin Lund-Andersen[14,17], Antonio Martini[1], Thomas Minotto[11], Johan Pensar[11], Knut Rand[1,2], Enrico Riccardi[1,2], Philippe A. Robert[9], Artur Rocha[7], Andrei Slabodkin[9], Igor Snapkov[9], Ludvig M. Sollid[3,9], Dmytro Titov[2], Cédric R. Weber[18], Michael Widrich[12], Gur Yaari[19], Victor Greiff[9◒], Geir Kjetil Sandve[1,2,3◒]

[1] Department of Informatics, University of Oslo, Norway
[2] Centre for Bioinformatics, University of Oslo, Norway
[3] K.G. Jebsen Centre for Coeliac Disease Research, Institute of Clinical Medicine, University of Oslo, Oslo, Norway
[4] Department of Pathology, Immunology and Laboratory Medicine, College of Medicine, University of Florida Diabetes Institute, USA
[5] Department of Pediatrics, College of Medicine, University of Florida Diabetes Institute, USA
[6] University Center for Information Technology, University of Oslo, Norway
[7] Institute for Systems and Computer Engineering, Technology and Science, Portugal
[8] Biological Sciences, Simon Fraser University, Canada
[9] Department of Immunology, University of Oslo, Norway
[10] Department of Population and Data Sciences, UT Southwestern Medical Center, USA
[11] Department of Mathematics, University of Oslo, Norway
[12] ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria
[13] Institute of Advanced Research in Artificial Intelligence (IARAI)
[14] Department of Tumor Biology, Institute for Cancer Research, Oslo University Hospital, The Norwegian Radium Hospital, Norway
[15] Centre for Molecular Medicine Norway (NCMM), Nordic EMBL Partnership, University of Oslo, Norway
[16] Department of Pathology, Leiden University Medical Center, the Netherlands
[17] Faculty of Medicine, Institute of Clinical Medicine, University of Oslo, Norway
[18] Department of Biosystems Science and Engineering, ETH Zürich, Switzerland
[19] Faculty of Engineering, Bar Ilan University, Israel

[#] Equal contribution, ◒ Joint supervision

Abstract

Adaptive immune receptor repertoires (AIRR) are key targets for biomedical research as they record past and ongoing adaptive immune responses. The capacity of machine learning (ML) to identify complex discriminative sequence patterns renders it an ideal approach for AIRR-based diagnostic and therapeutic discovery. To date, widespread adoption of AIRR ML has been inhibited by a lack of reproducibility, transparency, and interoperability. immuneML (immuneml.uio.no) addresses these concerns by implementing each step of the AIRR ML process in an extensible, open-source software ecosystem that is based on fully specified and shareable workflows. To facilitate widespread user adoption, immuneML is available as a command-line tool and through an intuitive Galaxy web interface, and extensive documentation of workflows is provided. We demonstrate the broad applicability of immuneML by (i) reproducing a large-scale study on immune state prediction, (ii) developing, integrating, and applying a novel method for antigen specificity prediction, and (iii) showcasing streamlined interpretability-focused benchmarking of AIRR ML.

Introduction

T-cell receptors (TCRs) and B-cell receptors (BCRs), that are collectively known as adaptive immune receptor (AIR) repertoires (AIRRs), recognize antigens and record information on past and ongoing immune responses[1–4]. AIRR-encoded information is particularly useful for the *repertoire*-based prediction and analysis of immune states (e.g., health, disease, infection, vaccination) in relation to other metadata such as major histocompatibility complex (MHC)[5–7], age[7,8], and sex[9]. Together this information shapes the foundation for AIRR-based diagnostics[6,10–13]. Similarly, s*equence*-based prediction of antigen and epitope binding is of fundamental importance for AIR-based therapeutics discovery and engineering[14–24]. In this manuscript, the term *AIRR* signifies both AIRs and AIRRs (a collection of AIRs) if not specified otherwise.

Machine learning (ML) has recently entered center stage in the biological sciences because it allows detection, recovery, and re-creation of high-complexity biological information from large-scale biological data[25–28]. AIRRs have complex biology with specialized research questions, such as immune state and receptor specificity prediction, that warrant domain-specific ML analysis[14]. Briefly, (i) ~10^8–10^{10} distinct AIRs exist in a given individual at any one time[29–31], with little overlap among individuals, necessitating encodings that allow detection of predictive patterns. These shared patterns may correspond to full-length AIRs[6] or subsequences[15] alternative representations thereof[11,12,16,17,21,32–34]. (ii) In repertoire-based ML, the patterns relevant to any immune state may be as rare as one antigen-binding AIR per million lymphocytes in a repertoire[35] translating into a very low rate of relevant sequences per repertoire (low witness rate)[11,36,37]. (iii) In sequence-based ML, the enormous diversity of antigen recognition combined with polyreactivity points to complex high-order statistical dependencies in the short sequence known to be the main determinant of antigen recognition (complementarity-determining region 3, CDR3)[1,15].

Tailored ML frameworks and platforms that account for the idiosyncrasies of the underlying data have been published for applications in genomics[38,39], proteomics[40,41], biomedicine[42], and chemistry[43]. Their creation recognizes the infeasibility to define, implement, and train appropriate ML models by relying solely on generic ML frameworks such as scikit-learn[44] or PyTorch[45]. The lack of a standardized framework for AIRR ML has led to heterogeneity in terms of technical solutions, domain assumptions, and user interaction options, hampering transparent comparative evaluation and the ability to explore and select the ML methodology most appropriate for a given study[14].

Results

immuneML overview

Here, we present immuneML, an open-source collaborative ecosystem for AIRR ML (Figure 1). immuneML enables the ML study of both experimental and synthetic AIRR-seq data that are labeled on the repertoire-level (e.g., immune state, sex, age, or any other metadata) or sequence-level (e.g., antigen binding), all the way from preprocessing to model training and model interpretation. It natively implements model selection and assessment procedures like nested cross-validation to ensure robustness in selecting the ML model. immuneML may be operated either via the command line or the Galaxy web interface, which offers an intuitive user interface that promotes collaboration and reusability through shareable analysis histories. To expedite analyses, immuneML may also be deployed to cloud services such as Amazon Web Services (AWS) and Google Cloud, or on a local server for data privacy concerns. Reproducibility and transparency are achieved by shareable specification files, which include all analysis details (Supplementary Figure 1). immuneML's compliance with AIRR community software and sequence annotation standards[46,47] ensures straightforward integration with third-party tools for AIRR data preprocessing and AIRR ML results' downstream analysis. For example, immuneML is fully compatible with the sequencing read processing and annotation suite MiXCR[48] and the Immcantation[49,50] and immunarch[51] frameworks for AIRR data analysis. AIRR data from the AIRR Data Commons[52] through the iReceptor Gateway[53], as well as the epitope-specific TCR database VDJdb[54] may be directly downloaded into the immuneML Galaxy environment. Additionally, immuneML is integrated with the AIRR-specific attention-based multiple-instance learning ML method DeepRC[37], the TCR-specific clustering method TCRdist[16], and is compatible with GLIPH2[55].

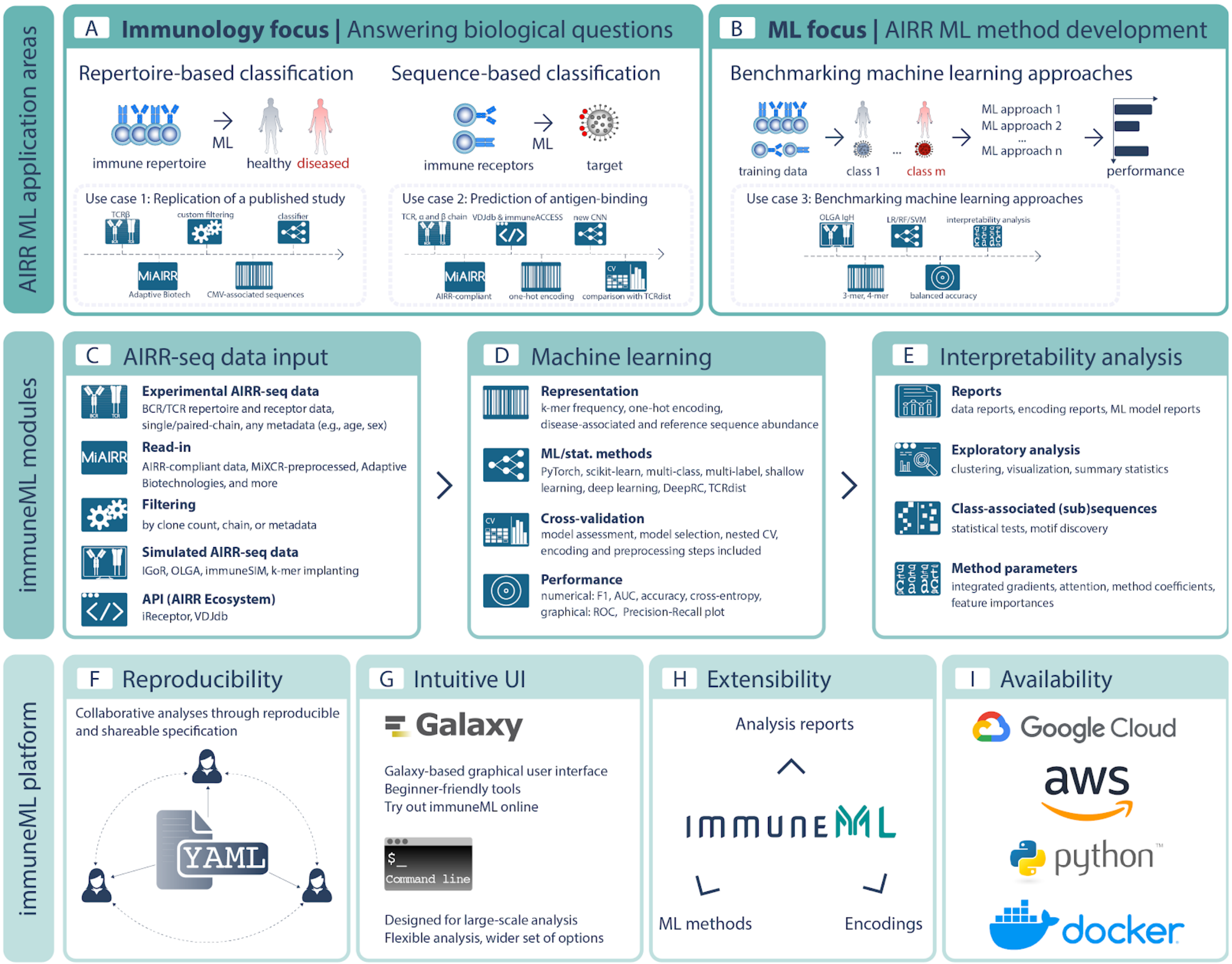

AIRR ML application areas
A Immunology focus | Answering biological questions
Repertoire-based classification
immune repertoire
ML
healthy diseased
Sequence-based classification
immune receptors
ML
target
Use case 1: Replication of a published study
TCRβ
custom filtering
classifier
MiAIRR
Adaptive Biotech
CMV-associated sequences
Use case 2: Prediction of antigen-binding
TCR, α and β chain
VDJdb & immuneACCESS
new CNN
MiAIRR
AIRR-compliant
one-hot encoding
CV
comparison with TCRdist
B ML focus | AIRR ML method development
Benchmarking machine learning approaches
training data
class 1
...
class m
ML approach 1
ML approach 2
...
ML approach n
performance
Use case 3: Benchmarking machine learning approaches
OLGA IgH
LR/RF/SVM
interpretability analysis
3-mer, 4-mer
balanced accuracy
immuneML modules
C AIRR-seq data input
Experimental AIRR-seq data
BCR/TCR repertoire and receptor data, single/paired-chain, any metadata (e.g., age, sex)
MiAIRR
Read-in
AIRR-compliant data, MiXCR-preprocessed, Adaptive Biotechnologies, and more
Filtering
by clone count, chain, or metadata
Simulated AIRR-seq data
IGoR, OLGA, immuneSIM, k-mer implanting
API (AIRR Ecosystem)
iReceptor, VDJdb
D Machine learning
Representation
k-mer frequency, one-hot encoding, disease-associated and reference sequence abundance
ML/stat. methods
PyTorch, scikit-learn, multi-class, multi-label, shallow learning, deep learning, DeepRC, TCRdist
CV
Cross-validation
model assessment, model selection, nested CV, encoding and preprocessing steps included
Performance
numerical: F1, AUC, accuracy, cross-entropy, graphical: ROC, Precision-Recall plot
E Interpretability analysis
Reports
data reports, encoding reports, ML model reports
Exploratory analysis
clustering, visualization, summary statistics
Class-associated (sub)sequences
statistical tests, motif discovery
Method parameters
integrated gradients, attention, method coefficients, feature importances
immuneML platform
F Reproducibility
Collaborative analyses through reproducible and shareable specification
YAML
G Intuitive UI
Galaxy
Galaxy-based graphical user interface
Beginner-friendly tools
Try out immuneML online
$_
Command line
Designed for large-scale analysis
Flexible analysis, wider set of options
H Extensibility
Analysis reports
immuneML
ML methods
Encodings
I Availability
Google Cloud
aws
python
docker

Figure 1 | Overview of immuneML. (A) The main immuneML application areas are sequence- and repertoire-based prediction of AIRR with application to (**A**) immunodiagnostics and therapeutics research, as well as to (**B**) develop AIRR-based methods. We show three use cases belonging to these application areas. Use case 1: reproduction of the study by Emerson et al.[6] on repertoire classification, use case 2: extending the platform with a novel convolutional neural network (CNN) classifier for prediction of TCR-pMHC binding that allows paired-chain input, use case 3: benchmarking ML methods with respect to their ability to recover a sequence-implanted immune signal. The immuneML core is composed of three pillars, which are (**C**)

AIRR-seq data input and filtering, (**D**) ML, and (**E**) Interpretability analysis. Each of these pillars has different modules that may be interconnected to build an immuneML workflow. (**F**) immuneML uses a specification file (YAML), which is customizable and allows full reproducibility and shareability with collaborators or the broader research community. An overview of how immuneML analyses can be specified is given in Supplementary Figure 1. **(G)** immuneML may be operated via the Galaxy web interface or the command line. (**H**) All immuneML modules are extendable. Documentation for developers is available online. (**I**) immuneML is available as a Python package, a Docker image, and may be deployed to cloud frameworks (e.g., AWS, Google Cloud). Abbreviations: CMV (cytomegalovirus).

To get started with immuneML, we refer the reader to **Focus Box 1**. To demonstrate immuneML's capabilities for performing AIRR ML, we provide an overview of the main features of the platform, and then highlight three orthogonal use cases: (i) we reproduce the cytomegalovirus (CMV) serostatus prediction study of Emerson et al.[6] inside immuneML and examine the robustness of the approach showing one way of using immuneML for repertoire-based immune state prediction, (ii) we apply a new custom convolutional neural network (CNN) for the sequence-based task of antigen-binding prediction based on paired-chain TCR data and (iii) we show the use of immuneML for benchmarking AIRR ML methods.

immuneML allows read-in of experimental single- and paired-chain data from offline and online sources as well as the generation of synthetic data for ML benchmarking

Experimental data may be read-in directly if it complies with the major formats used for AIRR-seq data V(D)J annotation: AIRR-C standard-conforming[46], MIXCR[48], 10x Genomics[56], Adaptive Biotechnologies ImmunoSEQ[6,57] or VDJdb formats[54]. The AIRR-C format compatibility ensures that also synthetic data as generated by immuneSIM[58] can be imported. Importing synthetic data as generated by IGoR[59] and OLGA[60] is also supported. Moreover, immuneML can be configured to read in data from any custom tabular format. To facilitate access to large-scale AIRR-seq data repositories, we provide Galaxy tools to download data from the AIRR Data Commons[52] via the iReceptor Gateway[53] and from VDJdb[54] into the Galaxy environment for subsequent ML analysis. Furthermore, immuneML includes built-in capacities for complex synthetic AIRR data generation to satisfy the need for ground-truth data in the context of ML method benchmarking. Finally, read-in data may be filtered by clone count, metadata, and chain.

immuneML supports multiple ML frameworks and allows for interpretation of ML models

immuneML supports two major ML platforms to ensure flexibility: scikit-learn[44] and PyTorch[45] and, therefore, is compliant with all ML methods inside these platforms. immuneML features scikit-learn implementations such as logistic regression, support vector machine, and random forest. In addition, we provide AIRR-adapted ML methods. Specifically, for repertoire classification, immuneML includes a custom implementation of the method published by Emerson et al.[6], as well as the attention-based deep learning method DeepRC[37]. For paired-chain sequence-based prediction, immuneML includes a custom-implemented CNN-based deep learning method, integrates with TCRdist[16], and is compatible with GLIPH2[55]. immuneML also includes several encodings that are commonly used for AIRR data such as k-mer frequency decomposition, one-hot encoding where each position in the sequence is represented by a vector of zeros except one entry containing 1 denoting appropriate amino acid or nucleotide, encodings by the presence of disease-associated sequences, and repertoire distances. For the full overview of analysis components, see Supplementary Table 1.

A variety of tabular and graphical analysis reports may be automatically generated as part of an analysis, providing details about the encoded data (e.g., feature value distributions), the ML model (e.g., interpretability reports), and the prediction accuracy (a variety of performance metrics across training, validation, and test sets). Additionally, the trained models may be exported and used in future analyses.

immuneML facilitates reproducibility, interoperability, and transparency of ML models

immuneML draws on a broad range of techniques and design choices to ensure that it meets the latest expectations with regard to usability, reproducibility, interoperability, extensibility, and transparency[61–64] (Figure 1).

Usability is achieved by a range of installation and usage options, catered to novices and experts, and to small and large-scale analyses. A Galaxy web interface allows users to run analyses without the need for any installation and without requiring any skills in programming or command-line operations. Availability through GitHub, pip, and Docker streamlines usage at scales ranging from laptops to high-performance infrastructures such as Google Cloud and AWS (docs.immuneml.uio.no/installation/cloud.html).

Reproducibility is ensured by leveraging the Galaxy framework that enables sharing of users' analysis histories, including the data and parameters, so that they can be independently reproduced. If working outside Galaxy, reproducibility is ensured by shareable analysis specification (YAML) files. YAML specification files produced in the Galaxy web interface can also be downloaded to seamlessly switch between Galaxy and command-line operation.

Interoperability is ensured by supporting the import from multiple data sources and export into AIRR-C format (MiAIRR standard) for post-analysis by third-party tools that are AIRR-compliant[46].

Extensibility of immuneML, signifying straightforward inclusion of new ML methods, encodings, reports, and preprocessing, is ensured by its modular design (Supplementary Figure 2). The code is open-source and available on GitHub (**Focus Box 2**). The documentation details step-by-step developer tutorials for immuneML extension (docs.immuneml.uio.no/developer_docs.html).

Transparency is established by (i) a YAML analysis specification in which the assumptions of the AIRR ML analysis are explicitly defined, and default parameter settings are exported, (ii) separate immunologist-centric Galaxy user interfaces that translate parameters and assumptions of the ML process to aspects of immune receptors that immunologists may better relate to (Supplementary Figure 3) and (iii) for each analysis report, the availability of underlying data for further user inspection.

Focus Box 1: Getting started with immuneML

➔ Visit the project website at immuneml.uio.no. immuneML may be used (i) online via the Galaxy web interface (galaxy.immuneml.uio.no), (ii) through a Docker container, or (iii) from the command line by installing and running immuneML as a Python package. Detailed instructions for each of these options are available in the immuneML documentation: docs.immuneml.uio.no/installation.html.

Getting started: web interface

- ➔ For immunologists, we recommend the Quickstart guide based on simplified interfaces for training ML models: docs.immuneml.uio.no/quickstart/galaxy_simple.html. Explanations of the relevant ML concepts can be found in the documentation (sequence classification docs.immuneml.uio.no/galaxy/galaxy_simple_receptors.html and repertoire classification docs.immuneml.uio.no/galaxy/galaxy_simple_repertoires.html)

- ➔ Alternatively, to have full control over all details of the analysis, see the YAML-based Galaxy Quickstart guide: docs.immuneml.uio.no/quickstart/galaxy_yaml.html.

- ➔ For guidance on how to use each immuneML Galaxy tool, see the immuneML & Galaxy documentation (docs.immuneml.uio.no/galaxy.html) and the list of published example Galaxy histories (galaxy.immuneml.uio.no/histories/list_published).

Getting started: command-line interface

- ➔ For the command-line Quickstart guide, see docs.immuneml.uio.no/quickstart/cli_yaml.html

- ➔ For detailed examples of analyses that can be performed with immuneML, see the tutorials (docs.immuneml.uio.no/tutorials.html), use case examples (docs.immuneml.uio.no/usecases.html), and see all supported analysis options in the YAML specification documentation (docs.immuneml.uio.no/specification.html).

For any questions, contact us at contact@immuneml.uio.no, visit the troubleshooting page in the documentation (docs.immuneml.uio.no/troubleshooting.html), or open an issue on our GitHub repository (github.com/uio-bmi/immuneML/issues).

Focus Box 2: How to contribute to immuneML

There exist multiple avenues for contributing and extending immuneML:

- ➔ ML workflows for specific research questions can be shared on galaxy.immuneml.uio.no, which allows other researchers to use them directly in their own data analysis.

- ➔ Questions, enhancements, or encountered bugs may be reported on the immuneML GitHub under “Issues” (github.com/uio-bmi/immuneML/issues).

- ➔ To improve or extend the immuneML platform, obtain the source code from GitHub at github.com/uio-bmi/immuneML. The immuneML codebase is described in the immuneML developer documentation docs.immuneml.uio.no/developer_docs.html, along with tutorials on how to add new ML methods, encodings, and report components to the platform.

➔ We encourage developers to contribute their improvements and extensions back to the community, either by making their own versions public or by submitting their contributions as GitHub “pull requests” to the main immuneML codebase.

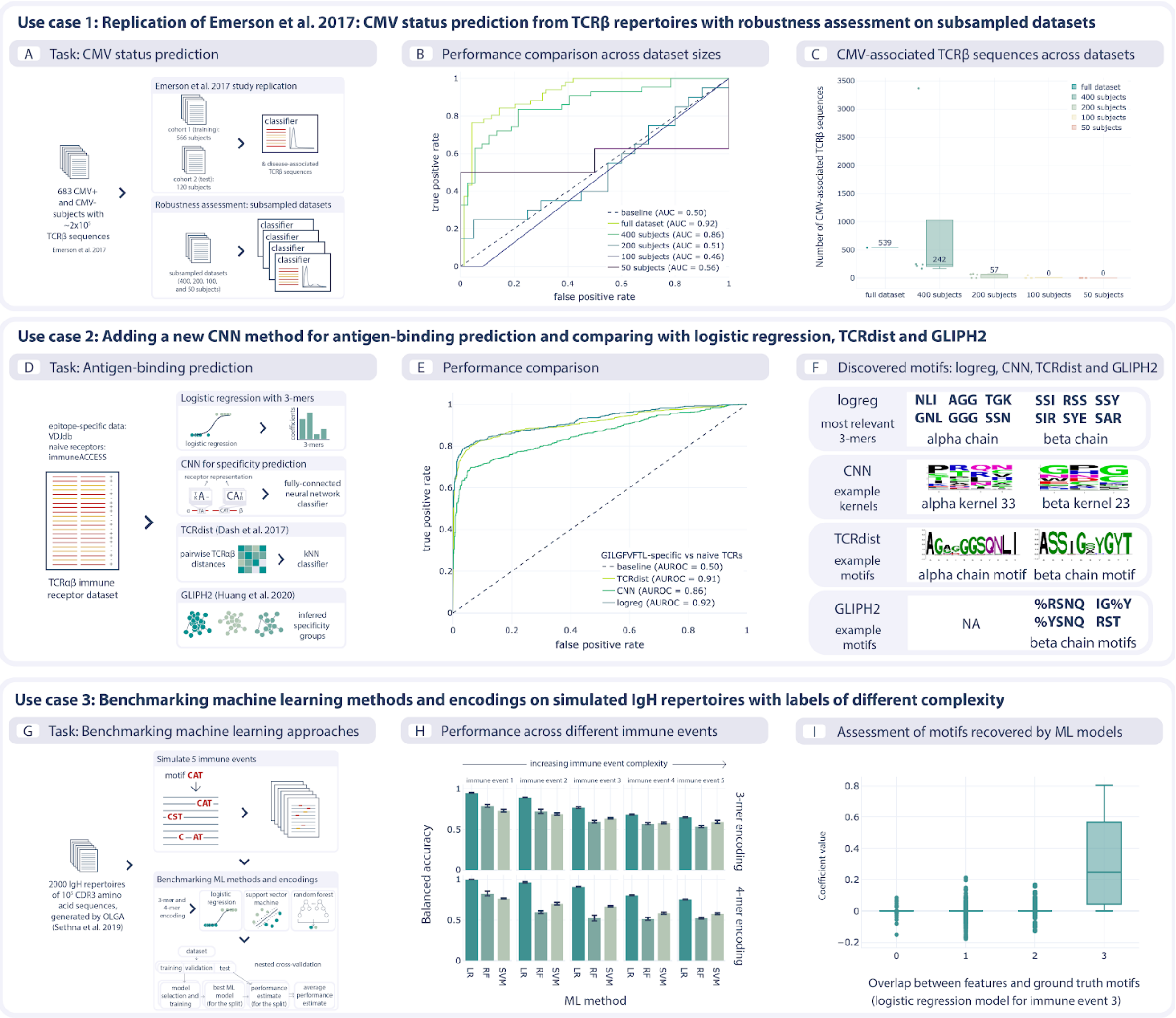


Figure 2 | Use cases demonstrating ML model training, benchmarking, and platform extension. We showcase three use cases to exemplify immuneML usage. **(A–C)** Use case 1: Reproduction of a published study[6] where the task consisted in distinguishing between TCRβ repertoires from CMV (cytomegalovirus) positive and negative individuals, as well as the identification of TCRβ sequences that are associated with CMV status. In addition, we assessed the robustness of the respective statistical approach, measured by the predictive performance, as a function of decreasing dataset size. We show how a lower number of repertoires (400, 200, 100, and 50) leads to decreased prediction accuracy (AUROC: 0.86–0.46) and a lower number of CMV-associated TCRβ sequences (with almost none found in datasets of 100 and 50 subjects). **(D–F)** Use case 2: We developed a new ML method for antigen-specificity prediction on paired-chain T-cell receptor data using a convolutional neural network (CNN) architecture. The method separately detects motifs in paired chains and combines the motif scores corresponding to kernel activations to obtain the receptor representation which is then used as input to a classifier. We compared the CNN method with the TCRdist-based k-nearest neighbor classifier and logistic regression on a dataset consisting of epitope-specific and naive TCRαβ sequences (assumed to be non-epitope-specific). For epitope-specific sequences, we used Epstein-Barr-virus-specific TCRαβ sequences binding to the GILGFVFTL epitope. We also show the motifs recovered by CNN, TCRdist, and GLIPH2 among the epitope-specific sequences. **(G–I)** Use case 3: We show how ground-truth synthetic data may be used to benchmark AIRR ML methods. The dataset consists of 2000 immune repertoires generated by OLGA[60]. Using immuneML, five immune events of increasing complexity are simulated by implanting synthetic signals into the OLGA-generated repertoires. This dataset is subsequently used to benchmark three different ML methods (logistic regression

(LR), support vector machine (SVM), and random forest (RF)) in combination with two encodings (3-mer and 4-mer encoding) inside immuneML, showing that the classification performance drops as the immune event complexity increases. The quality of the ML models was further assessed by comparing the feature coefficient sizes with how well these features represent the ground-truth signals. This revealed that models with a good classification performance were indeed able to recover the ground-truth signals.

Use case 1: Reproduction of a published study inside immuneML

To show how a typical AIRR ML analysis may be performed within immuneML, we reproduced a previously published study by Emerson et al. on the TCRβ-repertoire-based classification of individuals into CMV seropositive and seronegative[6] (Figure 2A). Using the standard interface of immuneML, we set up a repertoire classification analysis using 10-fold cross-validation on cohort 1 of 563 patients to choose optimal hyperparameters for immuneML's native implementation of the statistical classifier introduced by Emerson and colleagues. We then retrained the classifier on the complete cohort 1 and tested it on a second cohort (cohort 2) of 120 patients, as described in the original publication (see Methods).

immuneML exports classifier details, such as a list of immune-status-associated sequences for each classifier created during cross-validation, as well as a performance overview using the metrics of choice. We replicated the predictive performance achieved by Emerson et al.[6], finding 143 of the same CMV-associated TCRs (out of 164) reported in the original study.

We further used built-in robustness analysis of immuneML to explore how classification accuracy and the set of immune-status-associated sequences varied when learning classifiers based on smaller subsets of repertoires (Figure 2 A and B). While the exact set of learned immune-status-associated sequences varied across subsampled data of sizes close to the full dataset, the classification accuracy was nonetheless consistently high (>0.85) as long as the number of training repertoires was 400 or higher (below this, classification accuracy on the separate test sets deteriorated sharply) (Figure 2 B and C). The YAML specification files and detailed results are provided in the immuneML use case documentation (docs.immuneml.uio.no/usecases/emerson_reproduction.html).

Use case 2: Extending immuneML with a deep learning component for antigen specificity prediction based on paired-chain (single immune cell) data

To illustrate the extensibility of the immuneML platform, we added a new CNN component for predicting antigen specificity based on paired-chain AIR data. The ML task is to discover motifs in the two receptor chains (sequences) and to exploit the presence of these motifs to predict if the receptor will bind the antigen. As the immuneML platform provides comprehensive functionality for parsing and encoding paired-chain data, for hyperparameter optimization, and for presenting results, the only development step needed was to add the code for the CNN-based method itself (Supplementary Figure 5). Briefly, the added CNN consists of a set of kernels for each chain that act as motif detectors, a vector representation of the receptor obtained by combining all kernel activations, and a fully-connected layer that predicts if the receptor will bind the antigen or not. Furthermore, we show how to run analyses with the added component and compare its results with those of alternative models, such as a logistic regression model based on 3-mer frequencies and a k-nearest neighbor classifier relying on TCRdist[16] as the distance metric (available directly from immuneML through the tcrdist3 package[65]). We also show that the motifs can be recovered from the CNN model, the logistic regression, TCRdist, and GLIPH2[55] (Figure 2 D).

Use case 3: ML methods benchmarking on ground-truth synthetic data

Given the current rise in AIRR ML applications, the ability for method developers and practitioners to efficiently benchmark the variety of available approaches is becoming crucial[1,14,58]. Due to the limited current availability of high-resolution, labeled experimental data, rigorous benchmarking relies on a combination of experimental and simulated ground-truth data. The immuneML platform natively supports both the generation of synthetic data for benchmarking purposes and the efficient comparative benchmarking of multiple methodologies based on synthetic as well as experimental data. To exhibit the efficiency with which such benchmarking can be performed within the immuneML framework, we simulated, using the OLGA framework[60], 2000 human IgH repertoires consisting of 10^5 CDR3 amino acid sequences each, and implanted sequence motifs reflecting five different immune events of varying complexity (Figure 2 G, Supplementary Table 2). We examined the classification accuracy of three assessed ML methods (Figure 2 H) and used a native immuneML report to examine the overlap between ground truth implanted motifs and learned model features (Figure 2 I, Supplementary Figure 6).

Discussion

We have presented immuneML, a collaborative and open-source platform for transparent AIRR ML, accessible both via the command line and via an intuitive Galaxy web interface. immuneML supports the analysis of both BCR and TCR repertoires, with single or paired chains, at the sequence (receptor) and repertoire level. It accepts experimental data in a variety of formats and includes native support for generating synthetic AIRR data to benchmark the performance of AIRR ML approaches. As a flexible platform for tailoring AIRR ML analyses, immuneML features a broad selection of modular software components for data import, feature encoding, ML, and performance assessment (Supplementary Table 1). The platform can be easily extended with new encodings, ML methods, and analytical reports by the research community. immuneML supports all major standards in the AIRR field, uses YAML analysis specification files for transparency, and scales from local machines to the cloud. Extensive documentation for both users and contributors is available (docs.immuneml.uio.no).

immuneML caters to a variety of user groups and usage contexts. The Galaxy web tools make sophisticated ML-based receptor specificity and repertoire immune state prediction accessible to immunologists and clinicians through intuitive, graphical interfaces. The diversity of custom preprocessing and encoding used in published AIRR ML studies hinders their comparison and reproducibility. In contrast, the YAML-based specification of analyses on the command line or through Galaxy improves the collaboration, transparency, and reproducibility of AIRR ML for experienced bioinformaticians and data scientists. The integrated support for AIRR data simulation and systematic ML method benchmarking helps method *users* to select those approaches most appropriate to their analytical setting, and to assists method *developers* to effectively evaluate ML-related methodological ideas.

From a developer perspective, the impressive sophistication of generic ML frameworks such as TensorFlow[66] and PyTorch[45] may suggest that these frameworks would suffice as a starting point for AIRR ML method development. However, the fact that the immuneML architecture builds strictly on top of frameworks such as PyTorch underlines the breadth of additional functionality needed for robust ML development and execution in the AIRR domain. For ML researchers, the rich support for integrating novel ML components within existing code for data processing, hyper-parameter optimization, and performance assessment can greatly accelerate method development.

The current version of immuneML includes a set of components mainly focused on supervised ML, but the platform is also suitable for the community to extend it with components for settings such as unsupervised learning[67] or generative receptor modeling[14,19,68]. We also aim to improve the general support for model introspection, in particular in the direction of supporting causal interpretations for discovering and alleviating technical biases or challenges related to the study design[69].

In conclusion, immuneML enables the transition of AIRR ML method setup representing a bona fide research project to being at the fingertips of immunologists and clinicians. Complementally, AIRR ML method developers can focus on the implementation of components reflecting their unique research contribution, relying on existing immuneML functionality for the entire remaining computational process. immuneML facilitates the increased adoption of AIRR-based diagnostics and therapeutics discovery by supporting the accelerated development of AIRR ML methods.

Methods

immuneML availability: immuneML can be used (i) as a web tool through the Galaxy web interface (galaxy.immuneml.uio.no), (ii) from a command-line interface (CLI), (iii) through Docker (hub.docker.com/repository/docker/milenapavlovic/immuneml), (iv) via cloud services such as Google Cloud (cloud.google.com) through Docker integration, or (v) as a Python library (pypi.org/project/immuneML).

immuneML analysis specification: immuneML analyses are specified using a YAML specification file (Supplementary Figure 1), which describes analysis components as well as instructions to perform analyses using these components. When using Galaxy, the user may choose to provide a specification file directly or use a graphical interface that compiles the specification for the user. When used as a CLI tool, locally or in the cloud, with or without Docker, the specification file is provided by the user. Examples of specification files and detailed documentation on how to create them are available at docs.immuneml.uio.no/tutorials/how_to_specify_an_analysis_with_yaml.html.

immuneML supports different types of instructions: (i) training and assessment of ML models, (ii) applications of trained ML models, (iii) exploratory data analysis, and (iv) generation of synthetic AIRR datasets. Tutorials detailing these instructions are available at docs.immuneml.uio.no/tutorials.html.

immuneML public instance: the immuneML Galaxy web interface is available at galaxy.immuneml.uio.no. In addition to core immuneML components, the Galaxy instance includes interfaces towards the VDJdb[54] database and the iReceptor Gateway[53]. The documentation for the Galaxy immuneML tools is available at docs.immuneml.uio.no/galaxy.html.

immuneML architecture: immuneML has a modular architecture that can easily be extended (Supplementary Figure 2). In particular, we have implemented glass-box extensibility mechanisms[70], which enable the creation of customized code to implement new functionalities (encodings, ML methods, reports) that might be needed by the users. Such extensibility mechanisms allow the users to adapt immuneML to their specific cases without the need to understand the complexity of the immuneML code. For tutorials on how to add a new ML method, encoding, or an analysis report, see the developer documentation: docs.immuneml.uio.no/developer_docs.html.

Use cases:

Use case 1: Reproduction of a published study inside immuneML

We reproduced the study by Emerson and colleagues using a custom implementation of the encoding and classifier described in the original publication[6]. Out of the 786 subjects listed in the original study, we removed 103 subjects (1 with missing repertoire data, 25 with unknown CMV status, 3 with negative template counts for some of the sequences, and the rest with no template count information, all of which occurred in cohort 1), and performed the analysis on the remaining 683 subjects. We achieved comparable results to the original publication, as shown in Supplementary Figure 4. Supplementary Table 3 shows TCRβ receptor sequences inferred to be CMV-associated, comparing them to those published by Emerson et al.

In addition to reproducing the Emerson et al. study, we retrained the classifier on datasets consisting of 400, 200, 100, and 50 TCRβ repertoires randomly subsampled from cohort 1 and cohort 2. We show how the performance and the overlap of CMV-associated sequences changes with such reductions of dataset size (Figure 2 B and C).

The YAML specification files for this use case are available in the immuneML documentation under use case examples: docs.immuneml.uio.no/usecases/emerson_reproduction.html. The complete collection of results produced by immuneML, as well as the subsampled datasets, can be found in the NIRD research data archive[71].

Use case 2: Extending immuneML with a deep learning component for antigen specificity prediction based on paired-chain (single immune cell) data

To demonstrate the ease of extensibility for the platform, we added a CNN-based receptor specificity prediction ML method to the platform (Supplementary Figure 5), detailing the steps needed to add this as a new component under use case examples in the immuneML documentation: docs.immuneml.uio.no/usecases/extendability_use_case.html. Subsequently, we ran the added component through the standard immuneML model training interface, comparing its predictive performance with TCRdist[16,65] and logistic regression across three datasets. Additionally, we recovered motifs from the kernels of the neural network by limiting the values of the kernels similar to Ploenzke and Irizarry[72], and from the hierarchical clustering based on TCRdist distance, and compare these recovered motifs with the motifs extracted by GLIPH2[55] on the same datasets. Each dataset includes a set of epitope-specific TCRαβ receptors downloaded from VDJdb and a set of naive, randomly paired TCR$\alpha\beta$ receptors from the peripheral blood samples of 4 healthy donors[73]. Epitope-specific datasets are specific to cytomegalovirus (KLGGALQAK epitope, with 13000 paired TCRαβ receptors), Influenza A (GILGFVFTL epitope, with 2000 paired TCRαβ receptors), and Epstein-Barr virus (AVFDRKSDAK epitope, with 1700 paired TCRαβ receptors). Dataset details are summarized in Supplementary Table 4. The code for creating the datasets and YAML specifications describing the analysis can be found in the immuneML documentation:

docs.immuneml.uio.no/usecases/extendability_use_case.html. The three datasets of epitope-specific receptors, the complete collection of kernel visualizations produced by immuneML, as well as the results produced by GLIPH2, have been stored in the NIRD research data archive[74].

Use case 3: ML methods benchmarking on ground-truth synthetic data

To show immuneML's utility for benchmarking AIRR ML methods, we constructed a synthetic AIR dataset with known implanted ground-truth signals and performed a benchmarking of ML methods and encodings inside immuneML. To create the dataset for this use case, 2000 human IgH repertoires of 10^5 CDR3 amino acid sequences were generated using OLGA[60]. Subsequently, immuneML was used to simulate five different immune events of varying complexity by implanting signals containing probabilistic 3-mer motifs (Supplementary Table 2). The signals of each immune event were implanted in 50% of the repertoires, without correlating the occurrence of different immune events. Signals were implanted in 0.1% of the CDRH3 sequences of the repertoires selected for immune event simulation.

Using immuneML, three different ML methods (logistic regression, random forest, support vector machine) combined with two encodings (3-mer and 4-mer frequency encoding) were benchmarked. Hyperparameter optimization was done through nested cross-validation. For the model assessment (outer) cross-validation loop, the 2000 repertoires were randomly split into 70% training and 30% testing data, and this was repeated three times. In the model selection (inner) cross-validation loop, 3-fold cross-validation was used. The test set classification performances of the trained classifiers for each immune event are shown in Figure 2 H.

The immune signals implanted in this dataset can be used to examine the ability of the ML methods to recover ground-truth motifs by comparing the coefficient value (logistic regression, support vector machine) or feature importance (random forest) of a given feature with the overlap between that feature and an implanted signal (Figure 2 I, Supplementary Figure 6).

The bash script for generating the OLGA sequences, as well as the YAML specification files describing the simulation of immune events and benchmarking of ML methods are available in the immuneML documentation under use case examples: docs.immuneml.uio.no/usecases/benchmarking_use_case.html. The benchmarking dataset with simulated immune events as well as the complete collection of figures (for all cross-validation splits, immune events, ML methods, and encodings) can be downloaded from the NIRD research data archive[75].

Acknowledgements

We acknowledge generous support by The Leona M. and Harry B. Helmsley Charitable Trust (#2019PG-T1D011, to VG and TMB), UiO World-Leading Research Community (to VG and LMS), UiO:LifeScience Convergence Environment Immunolingo (to VG and GKS), EU Horizon 2020 iReceptorplus (#825821) (to VG), a Research Council of Norway FRIPRO project (#300740, to VG), a Research Council of Norway IKTPLUSS project (#311341, to VG and GKS), the National Institutes of Health (P01 AI042288 and HIRN UG3 DK122638 to TMB) and Stiftelsen Kristian Gerhard Jebsen (K.G.

Jebsen Coeliac Disease Research Centre) (to LMS and GKS). We acknowledge support from ELIXIR Norway in recognizing immuneML as a national node service.

Author contributions

MP, VG, GKS conceived the study. MP and GKS designed the overall software architecture. MP, LS, and KM developed the main platform code. MP and LS performed all analyses. MP, LS, CK, FLMB, RA, GSAH, GB, MC, RF, IG, SG, PHH, KR, ER, PAR, AS, DT, CW, and MW created software or documentation content. RK, NV, KW, LS, MP, AAC, and BC designed and developed the Galaxy tools. CK, RA, TB, MC, SC, LGC, IHH, EH, SH, GK, MLK, CLA, AM, TM, JP, KR, PAR, AR, IS, LMS and GY provided critical feedback. MP, LS, VG, GKS drafted the manuscript. VG and GKS supervised the project. All authors read and approved the final manuscript and are personally accountable for its content.

Competing Interests

VG declares advisory board positions in aiNET GmbH and Enpicom B.V.

References

1. Brown, A. J. *et al.* Augmenting adaptive immunity: progress and challenges in the quantitative engineering and analysis of adaptive immune receptor repertoires. *Mol. Syst. Des. Eng.* **4**, 701–736 (2019).
2. Georgiou, G. *et al.* The promise and challenge of high-throughput sequencing of the antibody repertoire. *Nat. Biotechnol.* **32**, 158–168 (2014).
3. Yaari, G. & Kleinstein, S. H. Practical guidelines for B-cell receptor repertoire sequencing analysis. *Genome Med.* **7**, 121 (2015).
4. Csepregi, L., Ehling, R. A., Wagner, B. & Reddy, S. T. Immune Literacy: Reading, Writing, and Editing Adaptive Immunity. *iScience* **23**, 101519 (2020).
5. DeWitt, W. S., III *et al.* Human T cell receptor occurrence patterns encode immune history, genetic background, and receptor specificity. *eLife* **7**, e38358 (2018).
6. Emerson, R. O. *et al.* Immunosequencing identifies signatures of cytomegalovirus exposure history and HLA-mediated effects on the T cell repertoire. *Nat. Genet.* **49**, 659–665 (2017).
7. Krishna, C., Chowell, D., Gönen, M., Elhanati, Y. & Chan, T. A. Genetic and environmental determinants of human TCR repertoire diversity. *Immun. Ageing* **17**, 26 (2020).
8. Britanova, O. V. *et al.* Age-Related Decrease in TCR Repertoire Diversity Measured with Deep and Normalized Sequence Profiling. *J. Immunol.* **192**, 2689–2698 (2014).
9. Schneider-Hohendorf, T. *et al.* Sex bias in MHC I-associated shaping of the adaptive immune system. *Proc. Natl. Acad. Sci.* **115**, 2168–2173 (2018).
10. Shemesh, O., Polak, P., Lundin, K. E. A., Sollid, L. M. & Yaari, G. Machine learning analysis of naïve B-cell receptor repertoires stratifies celiac disease patients and controls. *bioRxiv* 2020.11.09.371336 (2020) doi:10.1101/2020.11.09.371336.
11. Ostmeyer, J., Christley, S., Toby, I. T. & Cowell, L. G. Biophysicochemical motifs in T cell receptor sequences distinguish repertoires from tumor-infiltrating lymphocytes and adjacent healthy tissue. *Cancer Res.* canres.2292.2018 (2019) doi:10.1158/0008-5472.CAN-18-2292.

12. Beshnova, D. *et al.* De novo prediction of cancer-associated T cell receptors for noninvasive cancer detection. *Sci. Transl. Med.* **12**, (2020).
13. Liu, X. *et al.* T cell receptor β repertoires as novel diagnostic markers for systemic lupus erythematosus and rheumatoid arthritis. *Ann. Rheum. Dis.* **78**, 1070–1078 (2019).
14. Greiff, V., Yaari, G. & Cowell, L. Mining adaptive immune receptor repertoires for biological and clinical information using machine learning. *Curr. Opin. Syst. Biol.* (2020) doi:10.1016/j.coisb.2020.10.010.
15. Akbar, R. *et al.* A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding. *bioRxiv* 759498 (2019) doi:10.1101/759498.
16. Dash, P. *et al.* Quantifiable predictive features define epitope-specific T cell receptor repertoires. *Nature* **547**, 89–93 (2017).
17. Glanville, J. *et al.* Identifying specificity groups in the T cell receptor repertoire. *Nature* **547**, 94–98 (2017).
18. Springer, I., Besser, H., Tickotsky-Moskovitz, N., Dvorkin, S. & Louzoun, Y. Prediction of Specific TCR-Peptide Binding From Large Dictionaries of TCR-Peptide Pairs. *Front. Immunol.* **11**, (2020).
19. Friedensohn, S. *et al.* Convergent selection in antibody repertoires is revealed by deep learning. *bioRxiv* 2020.02.25.965673 (2020) doi:10.1101/2020.02.25.965673.
20. Mason, D. M. *et al.* Deep learning enables therapeutic antibody optimization in mammalian cells by deciphering high-dimensional protein sequence space. *bioRxiv* 617860 (2019) doi:10.1101/617860.
21. Moris, P. *et al.* Current challenges for unseen-epitope TCR interaction prediction and a new perspective derived from image classification. *Brief. Bioinform.* (2020) doi:10.1093/bib/bbaa318.
22. Graves, J. *et al.* A Review of Deep Learning Methods for Antibodies. *Antibodies* **9**, (2020).
23. Narayanan, H. *et al.* Machine Learning for Biologics: Opportunities for Protein Engineering, Developability, and Formulation. *Trends Pharmacol. Sci.* **42**, 151–165 (2021).
24. Fischer, D. S., Wu, Y., Schubert, B. & Theis, F. J. Predicting antigen specificity of single T cells based on TCR CDR3 regions. *Mol. Syst. Biol.* **16**, e9416 (2020).
25. Eraslan, G., Avsec, Ž., Gagneur, J. & Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nat. Rev. Genet.* **20**, 389–403 (2019).
26. Esteva, A. *et al.* A guide to deep learning in healthcare. *Nat. Med.* **25**, 24–29 (2019).
27. Vamathevan, J. *et al.* Applications of machine learning in drug discovery and development. *Nat. Rev. Drug Discov.* 1 (2019) doi:10.1038/s41573-019-0024-5.
28. Wainberg, M., Merico, D., Delong, A. & Frey, B. J. Deep learning in biomedicine. *Nat. Biotechnol.* **36**, 829–838 (2018).
29. Lythe, G., Callard, R. E., Hoare, R. L. & Molina-París, C. How many TCR clonotypes does a body maintain? *J. Theor. Biol.* **389**, 214–224 (2016).
30. Mora, T. & Walczak, A. M. How many different clonotypes do immune repertoires contain? *Curr. Opin. Syst. Biol.* **18**, 104–110 (2019).
31. Briney, B., Inderbitzin, A., Joyce, C. & Burton, D. R. Commonality despite exceptional diversity in the baseline human antibody repertoire. *Nature* **566**, 393–397 (2019).
32. Greiff, V. *et al.* Learning the High-Dimensional Immunogenomic Features That Predict Public and Private Antibody Repertoires. *J. Immunol.* ji1700594 (2017) doi:10.4049/jimmunol.1700594.
33. Parameswaran, P. *et al.* Convergent Antibody Signatures in Human Dengue. *Cell Host Microbe* **13**, 691–700 (2013).

34. Thomas, N. *et al.* 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**, 3181–3188 (2014).
35. Christophersen, A. *et al.* Tetramer-visualized gluten-specific CD4+ T cells in blood as a potential diagnostic marker for coeliac disease without oral gluten challenge. *United Eur. Gastroenterol. J.* **2**, 268–278 (2014).
36. Sidhom, J.-W. *et al.* DeepTCR: a deep learning framework for understanding T-cell receptor sequence signatures within complex T-cell repertoires. *bioRxiv* 464107 (2019) doi:10.1101/464107.
37. Widrich, M. *et al.* Modern Hopfield Networks and Attention for Immune Repertoire Classification. *Adv. Neural Inf. Process. Syst.* **33**, (2020).
38. Chen, K. M., Cofer, E. M., Zhou, J. & Troyanskaya, O. G. Selene: a PyTorch-based deep learning library for sequence data. *Nat. Methods* **16**, 315 (2019).
39. Kopp, W., Monti, R., Tamburrini, A., Ohler, U. & Akalin, A. Deep learning for genomics using Janggu. *Nat. Commun.* **11**, 3488 (2020).
40. Feng, J. *et al.* Firmiana: towards a one-stop proteomic cloud platform for data processing and analysis. *Nat. Biotechnol.* **35**, 409–412 (2017).
41. Gessulat, S. *et al.* Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning. *Nat. Methods* **16**, 509–518 (2019).
42. Tomic, A. *et al.* SIMON: Open-Source Knowledge Discovery Platform. *Patterns* **2**, (2021).
43. Wu, Z. *et al.* MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
44. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
45. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. in *Advances in Neural Information Processing Systems 32* (eds. Wallach, H. et al.) 8026–8037 (Curran Associates, Inc., 2019).
46. Rubelt, F. *et al.* Adaptive Immune Receptor Repertoire Community recommendations for sharing immune-repertoire sequencing data. *Nat. Immunol.* **18**, 1274–1278 (2017).
47. Vander Heiden, J. A. *et al.* AIRR Community Standardized Representations for Annotated Immune Repertoires. *Front. Immunol.* **9**, (2018).
48. Bolotin, D. A. *et al.* MiXCR: software for comprehensive adaptive immunity profiling. *Nat. Methods* **12**, 380–381 (2015).
49. Gupta, N. T. *et al.* Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. *Bioinformatics* **31**, 3356–3358 (2015).
50. Vander Heiden, J. A. *et al.* pRESTO: a toolkit for processing high-throughput sequencing raw reads of lymphocyte receptor repertoires. *Bioinformatics* **30**, 1930–1932 (2014).
51. Vadim Nazarov, immunarch.bot & Eugene Rumynskiy. *immunomind/immunarch: 0.6.5: Basic single-cell support*. (Zenodo, 2020). doi:10.5281/zenodo.3893991.
52. Christley, S. *et al.* The ADC API: A Web API for the Programmatic Query of the AIRR Data Commons. *Front. Big Data* **3**, (2020).
53. Corrie, B. D. *et al.* iReceptor: A platform for querying and analyzing antibody/B-cell and T-cell receptor repertoire data across federated repositories. *Immunol. Rev.* **284**, 24–41 (2018).
54. Bagaev, D. V. *et al.* VDJdb in 2019: database extension, new analysis infrastructure and a T-cell

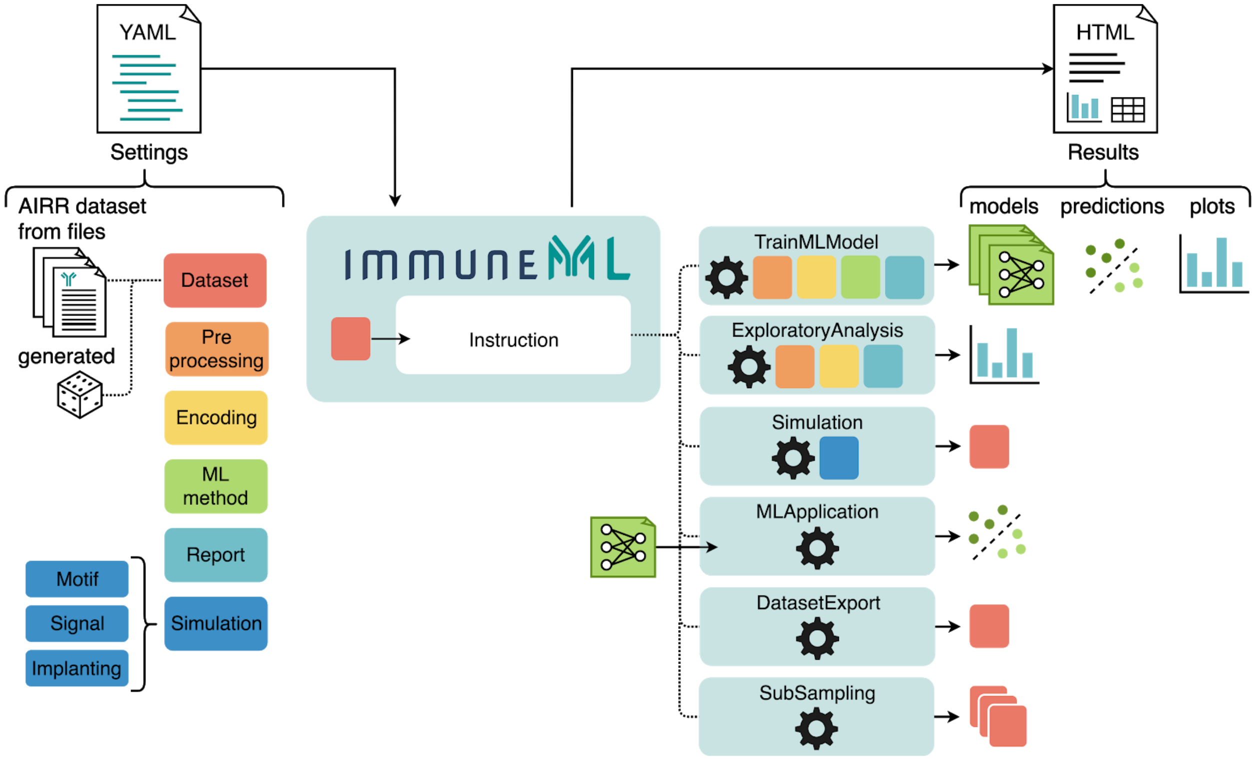
receptor motif compendium. *Nucleic Acids Res.* **48**, D1057–D1062 (2020).

55. Huang, H., Wang, C., Rubelt, F., Scriba, T. J. & Davis, M. M. Analyzing the Mycobacterium tuberculosis immune response by T-cell receptor clustering with GLIPH2 and genome-wide antigen screening. *Nat. Biotechnol.* 1–9 (2020) doi:10.1038/s41587-020-0505-4.
56. Zheng, G. X. Y. *et al.* Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* **8**, 14049 (2017).
57. Nolan, S. *et al.* A large-scale database of T-cell receptor beta (TCRβ) sequences and binding associations from natural and synthetic exposure to SARS-CoV-2. *Res. Sq.* (2020) doi:10.21203/rs.3.rs-51964/v1.
58. Weber, C. R. *et al.* immuneSIM: tunable multi-feature simulation of B- and T-cell receptor repertoires for immunoinformatics benchmarking. *Bioinformatics* **36**, 3594–3596 (2020).
59. Marcou, Q., Mora, T. & Walczak, A. M. High-throughput immune repertoire analysis with IGoR. *Nat. Commun.* **9**, 1–10 (2018).
60. Sethna, Z., Elhanati, Y., Callan, C. G., Walczak, A. M. & Mora, T. OLGA: fast computation of generation probabilities of B- and T-cell receptor amino acid sequences and motifs. *Bioinformatics* **35**, 2974–2981 (2019).
61. FAIR principles for data stewardship. *Nat. Genet.* **48**, 343–343 (2016).
62. Scott, J. K. & Breden, F. The adaptive immune receptor repertoire community as a model for FAIR stewardship of big immunology data. *Curr. Opin. Syst. Biol.* **24**, 71–77 (2020).
63. Breden, F. *et al.* Reproducibility and Reuse of Adaptive Immune Receptor Repertoire Data. *Front. Immunol.* **8**, (2017).
64. Software with impact. *Nat. Methods* **11**, 211–211 (2014).
65. Mayer-Blackwell, K. *et al.* TCR meta-clonotypes for biomarker discovery with tcrdist3: quantification of public, HLA-restricted TCR biomarkers of SARS-CoV-2 infection. *bioRxiv* 2020.12.24.424260 (2020) doi:10.1101/2020.12.24.424260.
66. Abadi, M. *et al.* TensorFlow: a system for large-scale machine learning. in *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* 265–283 (USENIX Association, 2016).
67. Vujovic, M. *et al.* T cell receptor sequence clustering and antigen specificity. *Comput. Struct. Biotechnol. J.* **18**, 2166–2173 (2020).
68. Davidsen, K. *et al.* Deep generative models for T cell receptor protein sequences. *eLife* **8**, e46935 (2019).
69. Bareinboim, E. & Pearl, J. Causal inference and the data-fusion problem. *Proc. Natl. Acad. Sci.* **113**, 7345–7352 (2016).
70. Zenger, M. '1.2 Characteristics of Extensibility Mechanisms.' Programming Language Abstractions for Extensible Software Components. (Lausanne: Swiss Federal Institute of Technology, 2004).
71. Pavlović, M. immuneML use case 1: Replication of a published study inside immuneML. (2021) doi:10.11582/2021.00008.
72. Ploenzke, M. S. & Irizarry, R. A. Interpretable Convolution Methods for Learning Genomic Sequence Motifs. *bioRxiv* 411934 (2018) doi:10.1101/411934.
73. Heikkilä, N. *et al.* Human thymic T cell repertoire is imprinted with strong convergence to shared sequences. *Mol. Immunol.* **127**, 112–123 (2020).
74. Pavlović, M. immuneML use case 2: Extending immuneML with a deep learning component for

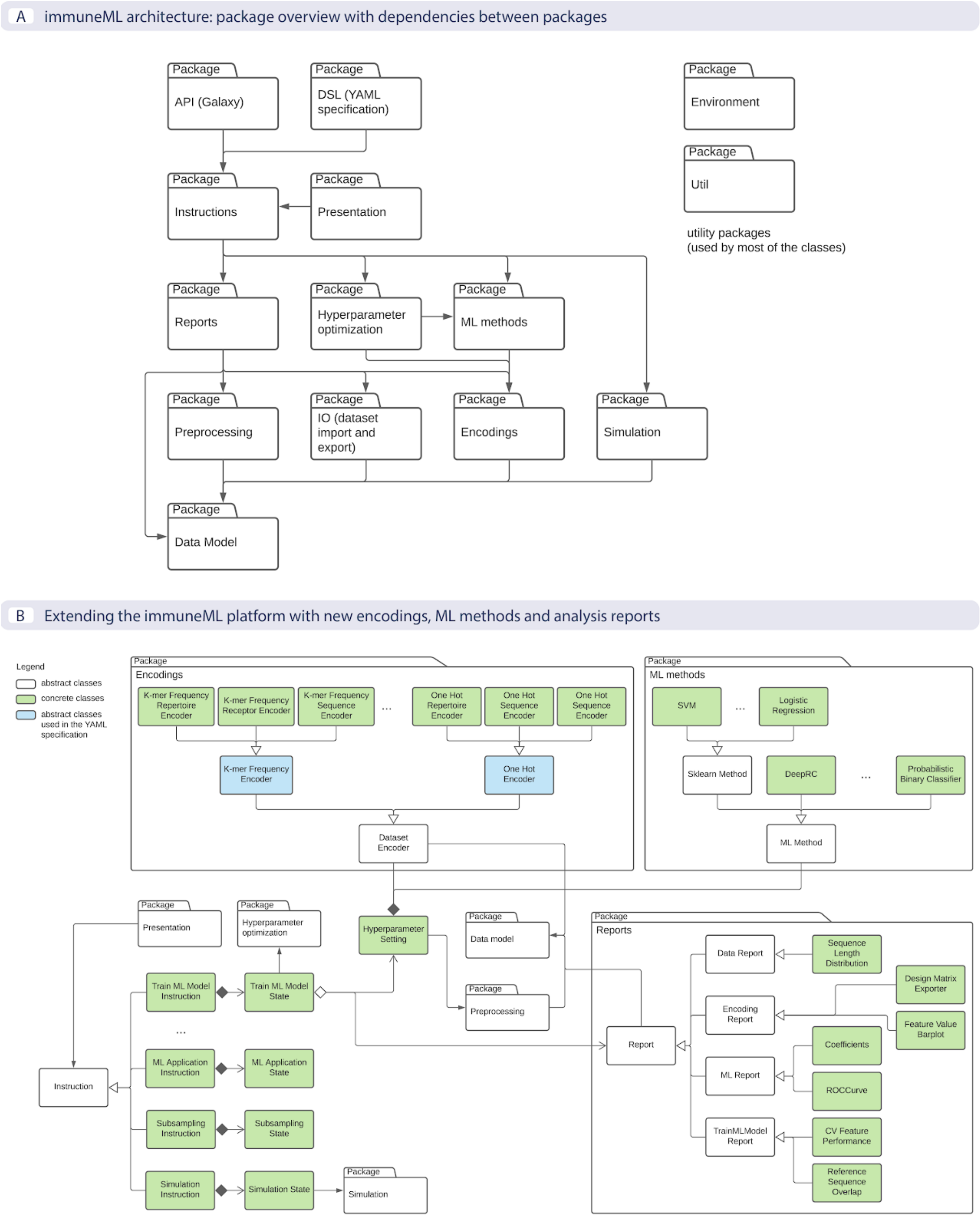
predicting antigen specificity of paired receptor data. (2021) doi:10.11582/2021.00009.

75. Scheffer, L. immuneML use case 3: Benchmarking ML methods for AIRR classification on ground-truth synthetic data. (2021) doi:10.11582/2021.00005.
76. Berman, H. M. *et al.* The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242 (2000).
77. Schrodinger, LLC. *The PyMOL Molecular Graphics System*. (2015).
78. immunoSEQ Analyzer | From Sequencing Data to Insights. *immunoseq.com* https://www.immunoseq.com/analyzer/.
79. Greiff, V. *et al.* A bioinformatic framework for immune repertoire diversity profiling enables detection of immunological status. *Genome Med.* **7**, (2015).
80. Řehůřek, R. & Sojka, P. Software Framework for Topic Modelling with Large Corpora. in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* 45–50 (ELRA, 2010).
81. Shannon, P. *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.* **13**, 2498–2504 (2003).
82. 10x Genomics. *A New Way of Exploring Immunity - Linking Highly Multiplexed Antigen Recognition to Immune Repertoire and Phenotype*. https://www.10xgenomics.com/resources/application-notes/a-new-way-of-exploring-immunity-linking-highly-multiplexed-antigen-recognition-to-immune-repertoire-and-phenotype/.
83. Campos-Lima, P. O., Levitsky, V., Imreh, M. P., Gavioli, R. & Masucci, M. G. Epitope-dependent selection of highly restricted or diverse T cell receptor repertoires in response to persistent infection by Epstein-Barr virus. *J. Exp. Med.* **186**, 83–89 (1997).
84. Chen, G. *et al.* Sequence and Structural Analyses Reveal Distinct and Highly Diverse Human CD8+ TCR Repertoires to Immunodominant Viral Antigens. *Cell Rep.* **19**, 569–583 (2017).
85. Grant, E. J. *et al.* Lack of Heterologous Cross-reactivity toward HLA-A*02:01 Restricted Viral Epitopes Is Underpinned by Distinct αβT Cell Receptor Signatures. *J. Biol. Chem.* **291**, 24335–24351 (2016).
86. Wang, Z. *et al.* Clonally diverse CD38+HLA-DR+CD8+ T cells persist during fatal H7N9 disease. *Nat. Commun.* **9**, 824 (2018).
87. Sant, S. *et al.* Single-Cell Approach to Influenza-Specific CD8+ T Cell Receptor Repertoires Across Different Age Groups, Tissues, and Following Influenza Virus Infection. *Front. Immunol.* **9**, 1453 (2018).
88. Lehner, P. J. *et al.* Human HLA-A0201-restricted cytotoxic T lymphocyte recognition of influenza A is dominated by T cells bearing the V beta 17 gene segment. *J. Exp. Med.* **181**, 79–91 (1995).

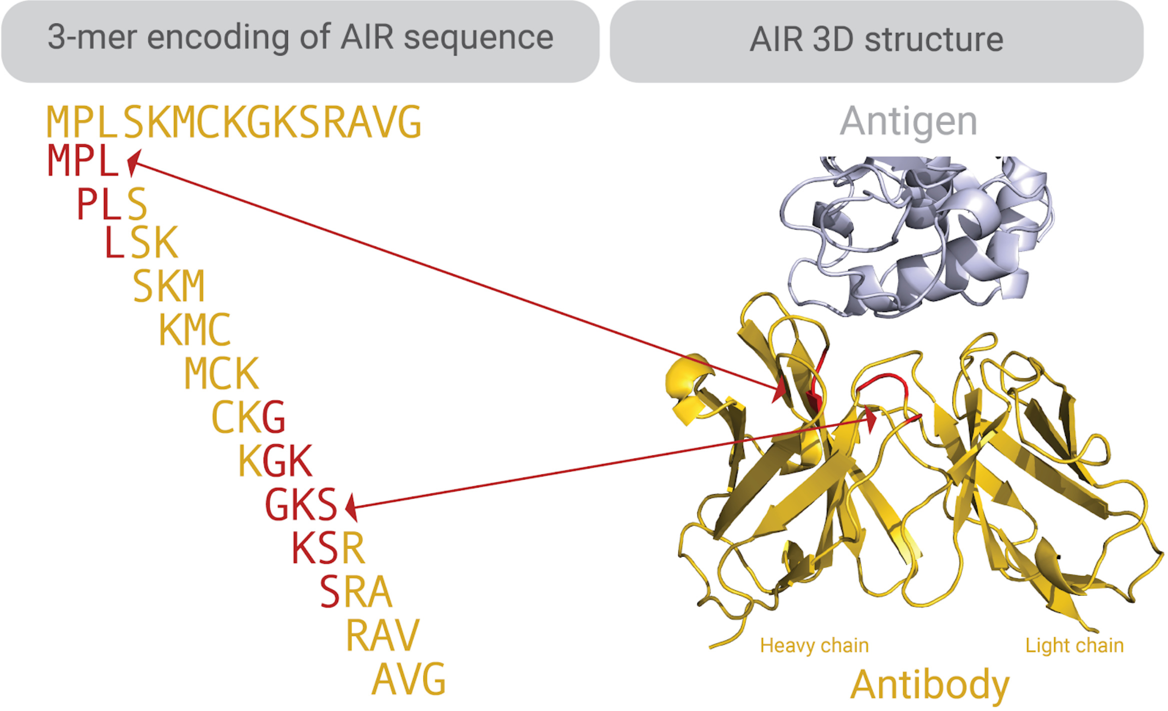
Supplementary Figures



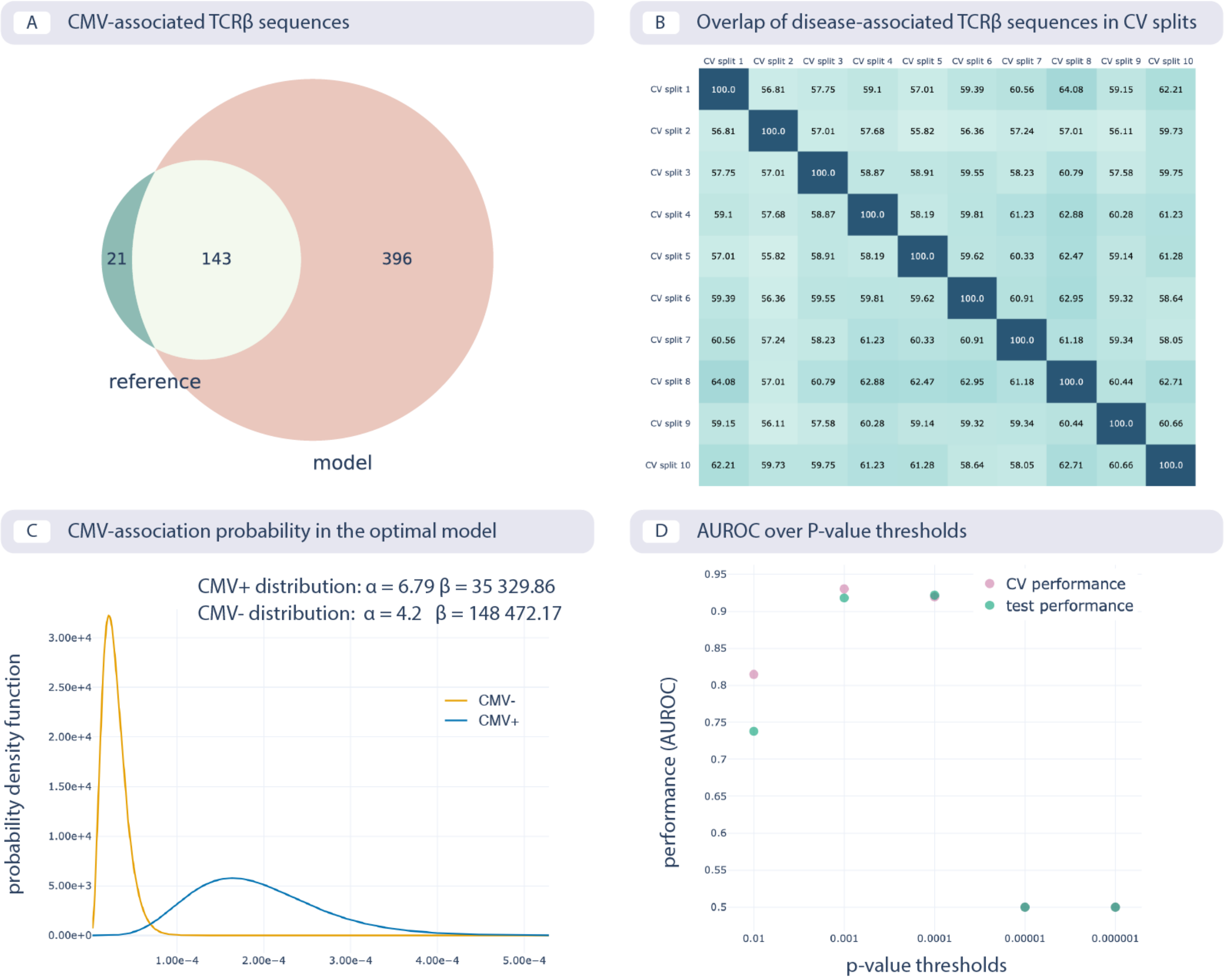
Supplementary Figure 1 | Overview of how immuneML analyses are specified. The YAML specification file describes the analysis components, and what instructions should be performed using these components. The analysis components are datasets, preprocessing, encoding, ML methods, analysis reports, and simulation-related components. Supplementary Table 1 contains a complete list of all components that can be specified. Instructions include training and applying ML models, exploratory analysis, and simulation of synthetic datasets. The results produced by the instructions can be navigated through an HTML summary page generated by immuneML.



Supplementary Figure 2 | immuneML architecture overview as UML (Unified Modeling Language) diagrams. **A.** High-level overview of the immuneML architecture showing the most important packages and their dependencies. The application programming interface (API) and domain-specific language (DSL) packages represent how the user can interact with immuneML, either through the Galaxy web interface (API package) or when constructing YAML specification files (DSL package). These packages invoke instructions, which map to different analyses that can be performed with immuneML, such as training an ML model or simulating an AIRR dataset. In turn, instructions depend on specific components to perform the analysis. **B.** To extend the platform with new encodings, ML methods, or reports, users may look into the corresponding package and implement the functionalities as described by the appropriate abstract class. The added components could then be used in different instructions according to their purpose. Developer tutorials are available at docs.immuneml.uio.no/developer_documentation.html.

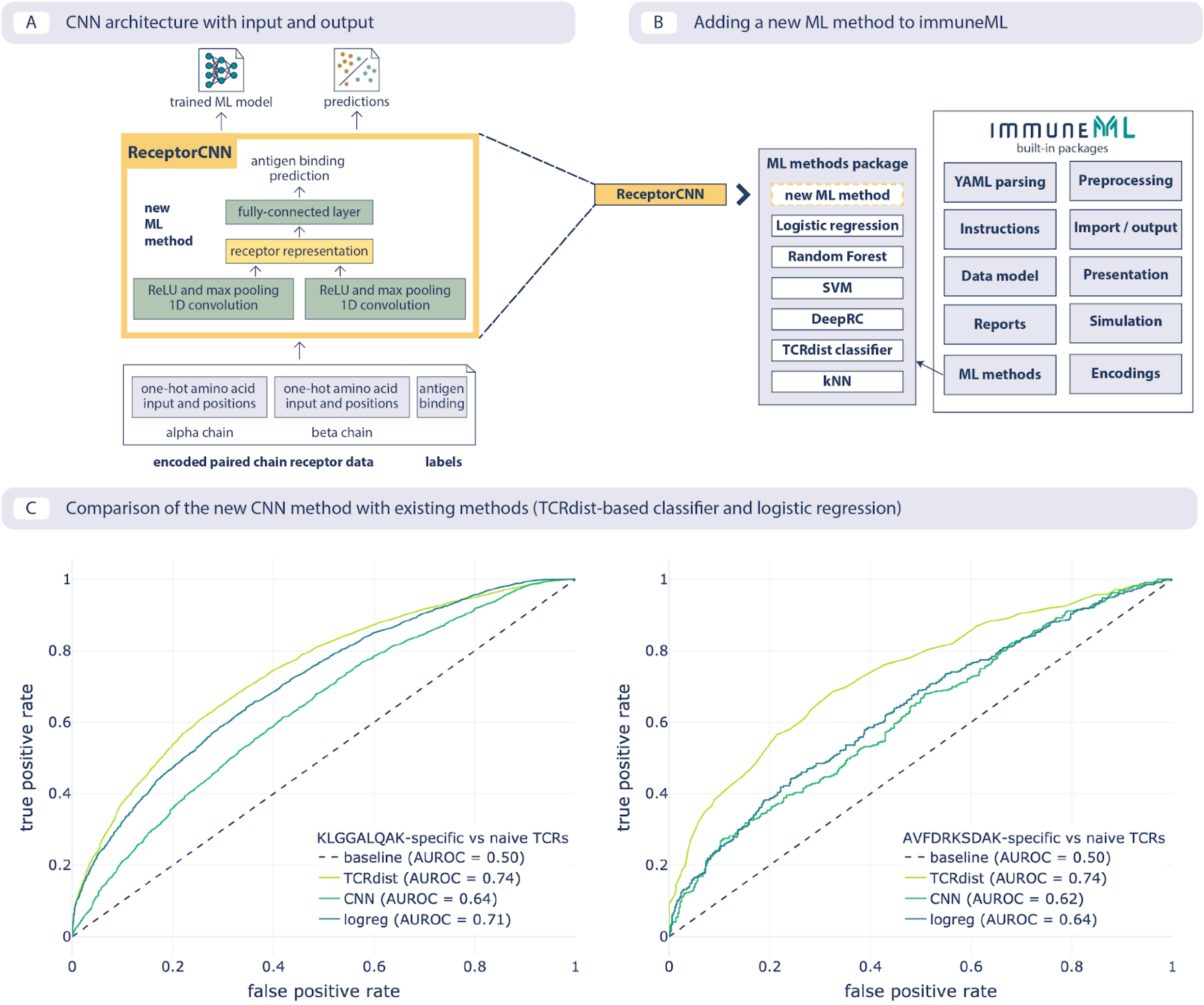


Supplementary Figure 3 | An example of how a k-mer encoded AIR (here antibody) sequence may map the antibody 3D-structure (Protein Data Bank (PDB) ID: 2DQC[76]). K-mers are subsequences of length *k*. Through ML, we can learn, for example, which k-mers are important for determining antigen specificity (color-coded in red). These k-mers may map to regions of the (CDR3) sequence that are in contact with the antigen.[15] 3D visualization of the antibody-antigen structure was carried out in Pymol[77].

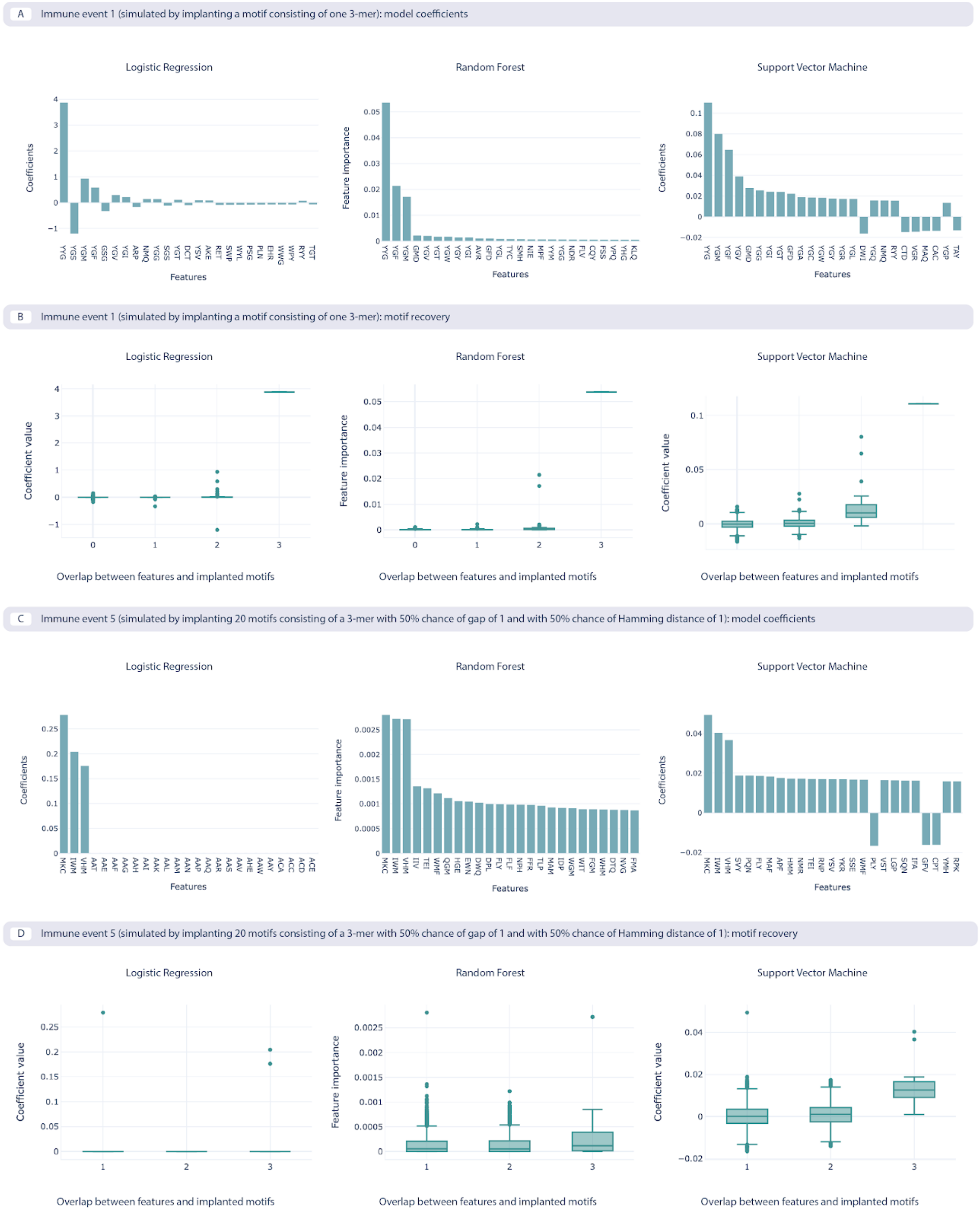


	CV split 1	CV split 2	CV split 3	CV split 4	CV split 5	CV split 6	CV split 7	CV split 8	CV split 9	CV split 10
CV split 1	100.0	56.81	57.75	59.1	57.01	59.39	60.56	64.08	59.15	62.21
CV split 2	56.81	100.0	57.01	57.68	55.82	56.36	57.24	57.01	56.11	59.73
CV split 3	57.75	57.01	100.0	58.87	58.91	59.55	58.23	60.79	57.58	59.75
CV split 4	59.1	57.68	58.87	100.0	58.19	59.81	61.23	62.88	60.28	61.23
CV split 5	57.01	55.82	58.91	58.19	100.0	59.62	60.33	62.47	59.14	61.28
CV split 6	59.39	56.36	59.55	59.81	59.62	100.0	60.91	62.95	59.32	58.64
CV split 7	60.56	57.24	58.23	61.23	60.33	60.91	100.0	61.18	59.34	58.05
CV split 8	64.08	57.01	60.79	62.88	62.47	62.95	61.18	100.0	60.44	62.71
CV split 9	59.15	56.11	57.58	60.28	59.14	59.32	59.34	60.44	100.0	60.66
CV split 10	62.21	59.73	59.75	61.23	61.28	58.64	58.05	62.71	60.66	100.0

Supplementary Figure 4 | Reproducing the CMV status prediction study by Emerson et al.[6] **A**. The overlap of the 164 disease-associated TCRβ sequences (V-TCRβaa-J) determined in the original study by Emerson et al., labeled "reference", with those determined by the optimal model as reproduced here with a p-value threshold of 0.001 (labeled "model"). **B**. The overlap percentage of disease-associated TCRβ sequences for the optimal model with the p-value threshold of 0.001 between different data splits in 10-fold cross-validation (between 50% and 65% overlap). **C.** The probability that a TCRβ sequence is CMV-associated follows a beta distribution estimated separately for CMV positive and negative subjects, which is then used for CMV status prediction of new subjects. **D.** Area under the ROC curve (AUROC) over p-value thresholds in training data (average AUROC over 10 cross-validation splits) and test data (AUROC in cohort 2).



Supplementary Figure 5 | Extending immuneML with a new ML method **A.** The architecture of the new convolutional neural network (CNN) is shown together with expected input in the format of encoded paired-chain receptors and predictions and trained model as output. **B.** Adding a new method requires implementing an interface for the method and then it can reuse the infrastructure (data model, encodings, nested cross-validation, visualizations) without any additional changes. **C**. An example usage where the method can be readily compared with other methods already available within the platform. Here, the area under the ROC curve (AUROC) is shown on two datasets: CMV-specific (epitope: KLGGALQAK, left), and EBV-specific (epitope: AVFDRKSDAK, right), for the CNN that was added (cnn), TCRdist-based k-nearest neighbors classifier (tcrdist) and logistic regression on 3-mer frequencies (logreg).



Supplementary Figure 6 | The benchmarking use case model coefficients and motif recovery, where the repertoire data is represented by 3-mer amino acid frequencies. Two immune events are shown. Immune event 1 (A, B) is the simplest event simulated by implanting a single 3-mer, while the immune event 5 (C, D) is the most complex one simulated by implanting 20 motifs consisting of a 3-mer with a 50% chance of having a gap and 50% chance of having a Hamming distance of 1. **A.** The 25 largest coefficients of the logistic regression model, feature importances on random forest model, and coefficients of the support vector machine (SVM) model with a linear kernel. The highest value of the coefficients corresponds to the implanted motif. **B.** Coefficient values for the features depending on the overlap between the recovered features that overlap with the implanted motif, measuring how well the recovered motifs correspond to the implanted motif, shown across the three ML models. **C.** The 25 largest coefficients and feature importances for the ML models trained on immune event 5. **D.** Overlap of recovered and implanted motifs for the ML models trained on immune event 5. Motif recovery for immune event 5 is less effective than for immune event 1.

Supplementary Tables

Supplementary Table 1 | Components for data import, encoding, ML methods, and reports included in immuneML.

Component type	**Name**	**Description**
Data import	AIRR	Imports data in AIRR format.
Data import	Generic	Fully configurable import class which can import data from any tabular file.
Data import	IGoR	Imports CDR3 sequences exported by IGoR[59] when running IGoR with the –CDR3 option specified
Data import	IReceptor	Imports datasets from the AIRR Data Commons[52] retrieved through the iReceptor Gateway[53]. This importer automatically parses repertoire metadata as exported by the iReceptor Gateway.
Data import	ImmunoSEQRearrangement	Imports data from Adaptive Biotechnologies immunoSEQ Analyzer[78] rearrangement-level .tsv files.
Data import	ImmunoSEQSample	Imports data from Adaptive Biotechnologies immunoSEQ Analyzer[78] sample-level .tsv files.
Data import	MiXCR	Imports data exported by MiXCR[48].
Data import	OLGA	Imports data generated by OLGA[60].
Data import	Pickle	Imports data from pickle files previously exported by immuneML.
Data import	RandomReceptorDataset	Automatically generate a paired receptor dataset consisting of random amino acids during runtime. This can be used for testing or benchmarking.
Data import	RandomRepertoireDataset	Automatically generate a repertoire dataset consisting of random amino acids during runtime. This can be used for testing or benchmarking.
Data import	RandomSequenceDataset	Automatically generate a sequence dataset consisting of random amino acids during runtime. This can be used for testing or benchmarking.

Data import	TenxGenomics	Imports files produced by 10x Genomics Cell Ranger analysis pipeline[56] named 'Clonotype consensus annotations (CSV)'.
Data import	VDJdb	Imports data from the VDJdb[54]
Preprocessing	ChainRepertoireFilter	Removes repertoires from the dataset containing sequences with illegal chain types, based on user-specified chain types (e.g., TRB, IGH).
Preprocessing	ClonesPerRepertoireFilter	Removes all repertoires from the dataset which contain too many or too few clonotypes based on user-specified thresholds.
Preprocessing	CountPerSequenceFilter	Removes all sequences from a repertoire which have a sequence count value below the user-specified threshold, or missing count values.
Preprocessing	DuplicateSequenceFilter	Collapses all sequences in a repertoire which have the same amino acid or nucleotide sequence and V and J genes.
Preprocessing	MetadataRepertoireFilter	Removes all repertoires from the dataset based on their specified metadata (e.g., remove repertoires when age is greater than a given threshold).
Preprocessing	SubjectRepertoireCollector	Merges all repertoires in the dataset that have the same subject identifier.
Encoding	AtchleyKmer	Represents a repertoire through Atchley factors and relative abundance of k-mers, as done by Ostmeyer and colleagues [11].
Encoding	Distance	Encodes a given repertoire dataset as a distance matrix, where the pairwise distance between each of the repertoires is calculated using a user-configurable distance metric.
Encoding	DeepRC	Prepare a repertoire dataset to be used with the DeepRC classifier[37].
Encoding	EvennessProfile	Encodes a repertoire as its evenness profile described by Greiff and colleagues[79].
Encoding	KmerFrequency	Encodes repertoires, sequences, or paired receptors by frequencies of k-mers it contains. A k-mer is a sequence of letters of length k into which an immune receptor sequence can be decomposed.
Encoding	MatchedReceptors	Takes a collection of paired reference receptors and counts how often they occur in a repertoire dataset.
Encoding	MatchedRegex	Takes a collection of regular expressions and counts how often they occur in a repertoire dataset.

Encoding	MatchedSequences	Takes a collection of reference sequences and counts how often they occur in a repertoire dataset.
Encoding	OneHot	One-hot encoding for repertoires, sequences or paired receptors. Positional information of the characters in the sequences may be included.
Encoding	SequenceAbundance	Represents the repertoires as vectors where the first element corresponds to the number of label-associated clonotypes, and the second element is the total number of unique clonotypes, as done by Emerson and colleagues[6].
Encoding	TCRdist	Encodes a paired receptor dataset as a distance matrix between all receptors, where the distance is computed using tcrdist3[65].
Encoding	Word2Vec	Encodes a repertoire dataset based on the vector representations of k-mers in the sequences in a repertoire from the context the k-mers appear in, using the Word2Vec implementation from the gensim package[80].
ML method	AtchleyKmerMILClassifier	A multiple instance learning classifier using AtchleyKmer encoding, replicating Ostmeyer et al.[11].
ML method	DeepRC	Internally uses the DeepRC method for repertoire classification[37].
ML method	KNN	A wrapper for the scikit-learn KNeighborsClassifier class[44].
ML method	LogisticRegression	A wrapper for the scikit-learn LogisticRegression class[44].
ML method	ProbabilisticBinaryClassifier	A classifier that predicts the class based on examples encoded by the number of successful trials and the total number of trials (SequenceAbundance encoding). It models this ratio by one beta distribution per class and predicts the class of the new examples using log-posterior odds ratio with the threshold at 0, replicating Emerson et al[6].
ML method	RandomForestClassifier	A wrapper for the scikit-learn RandomForestClassifier class[44].
ML method	ReceptorCNN	A CNN that detects motifs using CNN kernels in each chain of one-hot encoded paired receptors, combines the kernel activations into a unique representation of the receptor and uses this representation to predict the antigen-binding.
ML method	SVM	A wrapper for scikit-learn SVC[44].

ML method	TCRdistClassifier	Implementation of a nearest neighbors classifier based on TCR distances as presented in [16]. This method is implemented using the scikit-learn KNeighborsClassifier with k determined at runtime from the training dataset size and weights linearly scaled to decrease with the distance of examples
Data report	CytoscapeNetworkExporter	Exports paired receptors to .sif format such that they can directly be imported as a network in Cytoscape[81], to visualize chain sharing between the different receptors in a dataset.
Data report	GLIPH2Exporter	Exports paired receptor data to GLIPH2 format so that it can be directly used in GLIPH2[55].
Data report	ReceptorDatasetOverview	Plots a histogram of the lengths of AIRs per chain in a Receptor Dataset.
Data report	SequenceLengthDistribution	Plots a histogram of the lengths of the AIR sequences in an AIRR dataset.
Encoding report	DesignMatrixExporter	Exports the design matrix and related information of an encoded dataset to csv files.
Encoding report	FeatureDistribution	Plots the distribution of feature values for an encoded dataset. Distributions may be grouped based on metadata labels.
Encoding report	FeatureValueBarplot	Plots a barplot of feature values for an encoded dataset. Bars may be grouped based on metadata labels.
Encoding report	Matches	Reports the number of matches found when encoding the data using MatchedSequences, MatchedReceptors or MatchedRegex.
Encoding report	RelevantSequenceExporter	Exports the sequences that are extracted as label-associated by the SequenceAbundance encoder.
ML model report	Coefficients	Plot the coefficients or feature importance for a given ML model.
ML model report	ConfounderAnalysis	Plots the number of false positive and false negative predictions made for the examples (repertoires or receptors), grouped by the metadata.
ML model report	DeepRCMotifDiscovery	Plots the contributions of input sequences and kernels to trained DeepRC model[37] with respect to the test dataset. Contributions are computed using integrated gradients.
ML model report	KernelSequenceLogo	Plots the kernels of the ReceptorCNN model as sequence logos, as well as the weights in the final fully-connected layer of the network associated with kernel outputs.
ML model report	MotifSeedRecovery	Shows how well implanted motifs (simulated immune events) are recovered by ML methods using

		KmerFrequency encoding. This report plots the number of amino acids that overlap between the feature and implanted motif against the coefficient size of the respective feature.
ML model report	ROCCurve	Plot the ROC curve of a trained ML model.
ML model report	SequenceAssociationLikelihood	Plots the beta distribution used as a prior for a class assignment in ProbabilisticBinaryClassifier. The distribution plotted shows the probability that a sequence is associated with a given class for a label.
ML model report	TCRdistMotifDiscovery	Discovers motifs in the clusters created by the TCRdist classifier and creates logo plots for these motifs. This report internally uses the tcrdist3[65] library.
Train ML model report	CVFeaturePerformance	Plots the performance on the training and test datasets as a function of a specific parameter (feature), where there are multiple training or test datasets, it plots the average performance for the given feature value.
Train ML model report	DiseaseAssociatedSequenceCVOverlap	For disease-associated sequences discovered by SequenceAbundanceEncoder and ProbabilisticBinaryClassifier (implementing the CMV study[6]), shows the overlap of the sequences across cross-validation folds as a heatmap.
Train ML model report	MLSettingsPerformance	Plots the performance of each ML setting (a combination of ML method, encoding, and optionally preprocessing).
Train ML model report	ReferenceSequenceOverlap	Compares a list of disease-associated sequences produced by the SequenceAbundance encoder to a list of reference sequences, and creates a Venn diagram representing the overlap.
Multi dataset report	DiseaseAssociatedSequenceOverlap	Creates a heatmap showing the overlap of disease-associated sequences produced by SequenceAbundance encoder between multiple datasets of different sizes (different number of repertoires per dataset).
Multi dataset report	PerformanceOverview	Creates a ROC plot and precision-recall plot for optimal trained ML models on multiple datasets.

Supplementary Table 2 | An overview of the settings for the five different simulated immune signals (Figure 2G). One signal can be composed of one or more k-mers (here: 3-mers). For the gapped k-mers, a gap of length 1 was introduced with a 50% chance immediately before or after the middle amino acid of the 3-mer. In the two most complex signals, there was a 50% chance that one of the amino acids in the 3-mer was exchanged for a random different amino acid when implanting it in a sequence. Each immune signal is implanted in a different subset of the repertoires.

Immune event	3-mers count	Gap length	Hamming distance	Repertoires with immune events
Immune event 1	1	0	0	1–1000
Immune event 2	20	0	0	1–500, 1001–1500
Immune event 3	20	0 or 1 (50% chance)	0	1–250, 501–750, 1001–1250, 1501–1750
Immune event 4	20	0	0 or 1 (50% chance)	501–1500
Immune event 5	20	0 or 1 (50% chance)	0 or 1 (50% chance)	251–750, 1251–1750

Supplementary Table 3 | The list of CMV-associated CDR3β sequences with V and J gene determined by training the predictive model on cohort 1 in use case 1 when reproducing the study by Emerson and colleagues[6] that were also found in the original study. The full list of CMV-associated sequences (including the non-overlapping ones) is available in the NIRD research data archive[71].

CDR3β amino acid sequence	TRBV gene	TRBJ gene
ASSYPGETQY	TRBV6-6	TRBJ2-5
ASSQVPGQGDNEQF	TRBV14	TRBJ2-1
ASSSPGRSGANVLT	TRBV28	TRBJ2-6
ASSLEGQQPQH	TRBV28	TRBJ1-5
AWRGTGNSPLH	TRBV30	TRBJ1-6
ASSGDRLYEQY	TRBV2	TRBJ2-7
ASSPDRVGQETQY	TRBV5-1	TRBJ2-5
ASRRGSSYEQY	TRBV28	TRBJ2-7
ASSLIGVSSYNEQF	TRBV7-9	TRBJ2-1
ASSISAGEAF	TRBV19	TRBJ1-1
ASSTGTSGSYEQY	TRBV6-1	TRBJ2-7
ASSPGDEQF	TRBV25-1	TRBJ2-1
ASSLQGADTQY	TRBV7-8	TRBJ2-3
ASSPAGLNTEAF	TRBV19	TRBJ1-1
ASSPLSDTQY	TRBV7-9	TRBJ2-3
ASSLVGDGYT	TRBV7-8	TRBJ1-2
ASSSDRVGQETQY	TRBV5-1	TRBJ2-5
ASSPNQETQY	TRBV5-4	TRBJ2-5
ASSRGTGATDTQY	TRBV19	TRBJ2-3
ASSALGGAGTGELF	TRBV9	TRBJ2-2
ASMGGASYEQY	TRBV27	TRBJ2-7
ASSAQGAYEQY	TRBV9	TRBJ2-7
ASSPPSGLTDTQY	TRBV28	TRBJ2-3
ASSQNRGQETQY	TRBV14	TRBJ2-5

SVRDNYNQPQH	TRBV29-1	TRBJ1-5
ASSLPSGLTDTQY	TRBV28	TRBJ2-3
ASSVTGGTDTQY	TRBV9	TRBJ2-3
ASSDRGNTGELF	TRBV4-1	TRBJ2-2
ASSLTDTGELF	TRBV11-2	TRBJ2-2
ASSGQGAYEQY	TRBV9	TRBJ2-7
ASSLQGINQPQH	TRBV5-6	TRBJ1-5
ASSLAGVDYEQY	TRBV7-9	TRBJ2-7
ASSLVAGGRETQY	TRBV5-6	TRBJ2-5
ASSPSTGTEAF	TRBV5-6	TRBJ1-1
ASSLRREKLF	TRBV5-6	TRBJ1-4
ASSLQGYSNQPQH	TRBV5-8	TRBJ1-5
ASSRNRGQETQY	TRBV14	TRBJ2-5
ASSLGHRDSSYEQY	TRBV5-1	TRBJ2-7
ASSTSGNTIY	TRBV6-5	TRBJ1-3
ASSYGGEGYT	TRBV6-5	TRBJ1-2
ASSPGSGANVLT	TRBV19	TRBJ2-6
ASSLGAGNQPQH	TRBV28	TRBJ1-5
ATSRGTVSYEQY	TRBV15	TRBJ2-7
ASSRLAGGTDTQY	TRBV7-3	TRBJ2-3
ASSIGPLEHNEQF	TRBV19	TRBJ2-1
ASSAGQGVTYEQY	TRBV9	TRBJ2-7
ASSLGDRAYNEQF	TRBV5-6	TRBJ2-1
ASSPLGGTTEAF	TRBV18	TRBJ1-1
ASRPTGYEQY	TRBV6-1	TRBJ2-7
ASSLLWDQPQH	TRBV5-5	TRBJ1-5
ASSTTGGDGYT	TRBV19	TRBJ1-2
ASSLAPGATNEKLF	TRBV7-6	TRBJ1-4
ASSSGQVQETQY	TRBV11-2	TRBJ2-5
ASSFPGGETQY	TRBV11-1	TRBJ2-5
AWSVSDLAKNIQY	TRBV30	TRBJ2-4
ASSTGGAQPQH	TRBV19	TRBJ1-5
ASSLGQGLAEAF	TRBV5-1	TRBJ1-1
ASSVDGGRGTEAF	TRBV9	TRBJ1-1
ASSPQRNTEAF	TRBV4-3	TRBJ1-1
ASSFPTSGQETQY	TRBV7-9	TRBJ2-5
ASSYNPYSNQPQH	TRBV6-6	TRBJ1-5

ASSLNRGQETQY	TRBV14	TRBJ2-5
ASTPGDEQF	TRBV25-1	TRBJ2-1
ASSLGVGPYNEQF	TRBV7-2	TRBJ2-1
ASSQNRAQETQY	TRBV14	TRBJ2-5
ASSIEGNQPQH	TRBV28	TRBJ1-5
SASDHEQY	TRBV20-1	TRBJ2-7
ASSRLAASTDTQY	TRBV7-3	TRBJ2-3
ASSPGDEQY	TRBV25-1	TRBJ2-7
ASGRDTYEQY	TRBV2	TRBJ2-7
ASSEEGIQPQH	TRBV2	TRBJ1-5
ASSRDRNYGYT	TRBV6-4	TRBJ1-2
ASSRGRQETQY	TRBV7-6	TRBJ2-5
ASSEARGGVEKLF	TRBV6-1	TRBJ1-4
ASSEIPNTEAF	TRBV6-4	TRBJ1-1
ASRGQGAGELF	TRBV2	TRBJ2-2
ATSREGSGYEQY	TRBV15	TRBJ2-7
ASSLEAEYEQY	TRBV7-2	TRBJ2-7
ASSLRGSSYNEQF	TRBV5-8	TRBJ2-1
ASSRNRAQETQY	TRBV14	TRBJ2-5
ASSLGWTEAF	TRBV5-1	TRBJ1-1
ASSYVRTGGNYGYT	TRBV6-5	TRBJ1-2
ATSRDTQGSYGYT	TRBV15	TRBJ1-2
SVRDNHNQPQH	TRBV29-1	TRBJ1-5
ATSRDSQGSYGYT	TRBV15	TRBJ1-2
ASSIRTNYYGYT	TRBV19	TRBJ1-2
ASSLETYGYT	TRBV5-6	TRBJ1-2
ATSRVAGETQY	TRBV15	TRBJ2-5
ASRPQGNYGYT	TRBV28	TRBJ1-2
ASSIWGLDTEAF	TRBV19	TRBJ1-1
ASSSDSGGTDTQY	TRBV7-3	TRBJ2-3
SVRDNFNQPQH	TRBV29-1	TRBJ1-5
ASSLTGGNSGNTIY	TRBV7-2	TRBJ1-3
SVEVRGTDTQY	TRBV29-1	TRBJ2-3
ASSSGTGDEQY	TRBV5-1	TRBJ2-7
ASSPRWQETQY	TRBV27	TRBJ2-5
ASSEARTRAF	TRBV6-1	TRBJ1-1
ASSVLAGPTDTQY	TRBV9	TRBJ2-3

ASSEEAGGSGYT	TRBV6-1	TRBJ1-2
ASRTDSGANVLT	TRBV6-4	TRBJ2-6
ASSEAPSTSTDTQY	TRBV2	TRBJ2-3
ASSRNRESNQPQH	TRBV6-5	TRBJ1-5
ASNRDRGRYEQY	TRBV6-1	TRBJ2-7
ASSLGASGSRTDTQY	TRBV7-9	TRBJ2-3
ASSESGDPSSYEQY	TRBV10-1	TRBJ2-7
ASSLGDRPDTQY	TRBV11-2	TRBJ2-3
ASSLQAGANEQF	TRBV7-2	TRBJ2-1
ASRTGESGYT	TRBV6-5	TRBJ1-2
ASRGQGWDEKLF	TRBV6-5	TRBJ1-4
ASSWDRGTEAF	TRBV6-5	TRBJ1-1
ASSHRDRNYEQY	TRBV7-9	TRBJ2-7
ASSPSRNTEAF	TRBV4-3	TRBJ1-1
ASSIQGYSNQPQH	TRBV5-8	TRBJ1-5
ASSPGQEAGANVLT	TRBV5-1	TRBJ2-6
ASSLVIGGDTEAF	TRBV5-1	TRBJ1-1
ASSYGGLGSYEQY	TRBV6-5	TRBJ2-7
ASSPPGQGSDTQY	TRBV18	TRBJ2-3
SVEEDEGIYGYT	TRBV29-1	TRBJ1-2
ASRSDSGANVLT	TRBV6-4	TRBJ2-6
AISESQDRGHEQY	TRBV10-3	TRBJ2-7
ASSLVASGRETQY	TRBV5-6	TRBJ2-5
ASSSGQVYGYT	TRBV5-6	TRBJ1-2
ASSQGRHTDTQY	TRBV14	TRBJ2-3
ASSGLNEQF	TRBV6-1	TRBJ2-1
ASRDWDYTDTQY	TRBV2	TRBJ2-3
ASSSRGTGELF	TRBV28	TRBJ2-2
ASSPISNEQF	TRBV28	TRBJ2-1
ASSLGHRDPNTGELF	TRBV5-1	TRBJ2-2
ASSLGIDTQY	TRBV5-4	TRBJ2-3
ASSLEGQGFGYT	TRBV5-1	TRBJ1-2
ASSPHRNTEAF	TRBV4-3	TRBJ1-1
ASSESGHRNQPQH	TRBV10-2	TRBJ1-5
ASSFHGFNQPQH	TRBV5-6	TRBJ1-5
ASSEGARQPQH	TRBV10-2	TRBJ1-5
ASSSRTGEETQY	TRBV11-3	TRBJ2-5

ASSLEAENEQF	TRBV7-2	TRBJ2-1
ASSLVAAGRETQY	TRBV5-6	TRBJ2-5
ASSLAVLPTDTQY	TRBV7-9	TRBJ2-3
ASSLGRGYEKLF	TRBV5-6	TRBJ1-4
ASSWDRDNSPLH	TRBV25-1	TRBJ1-6
ASRDRDRVNTEAF	TRBV6-1	TRBJ1-1
ASSPTGGELF	TRBV18	TRBJ2-2
ASSVETGGTEAF	TRBV2	TRBJ1-1

Supplementary Table 4 | Three epitope-specific datasets in the use case 2.

Dataset	Number of TCRαβ receptors	Epitope	Epitope species	Data source: epitope-specific TCRs	Data source: naive receptors	Organism
AVFDRKSDAK	3460	AVFDRKSDAK	EBV	VDJdb[54]: 10x Genomics[82] and EBV study[83]	immuneACCESS: randomly paired TCRαβ CDR3 from peripheral blood of 4 healthy donors from Heikkilä et al.[73]	Human
GILGFVFTL	4090	GILGFVFTL	Influenza A	VDJdb[54]: multiple studies[16,17,82,84–88]		
KLGGALQAK	27386	KLGGALQAK	CMV	VDJdb[54]: 10x Genomics[82]		