ClusTCR: a Python interface for rapid clustering of large sets of CDR3 sequences with unknown antigen specificity

Supplementary materials

Sebastiaan Valkiers, Max Van Houcke,
Kris Laukens & Pieter Meysman

Contents

1 Methods

1.1 Data

1.1.1 Epitope-labeled data

TCR sequences with known epitope specificity were used to benchmark the clustering quality of our method. For this the latest version of the VDJdb (2021-02-02 release) was downloaded at https://github.com/antigenomics/vdjdb-db [1, 2]. From this, three separate data sets were created, containing only CDR3 α, CDR3 β or paired CDR3 $\alpha - \beta$, each matched with their cognate epitope as indicated in the VDJdb. This leveraged 24,267 unique α, 34,522 β and 22,186 paired $\alpha - \beta$ sequences. Each TCR-epitope pair in VDJdb is annotated with a quality score, which ranges from 0 to 3 (with 3 indicating the highest quality database entries, and 0 indicating the lowest quality database entries). For each category (α, β, $\alpha - \beta$) two subsamples were taken according to the VDJdb quality score. These subsamples only contained sequences that satisfied quality scores of ≥ 1 and ≥ 2. The subsamples consisted of 3392 and 1006 unique CDR3 sequences respectively, for the β chain. For the α chain, the subsets consisted of 920 and 360 sequences respectively.

1.1.2 Unlabeled data

The availability of epitope-labeled TCR sequencing data is limited to several ten thousand TCR-epitope pairs. Since ClusTCR was developed as a method as a method for clustering millions of TCR sequences, this number of sequences is insufficient as a benchmark to evaluate the computational performance of our algorithm (and other state-of-the-art algorithms). Therefore, unlabeled TCR repertoire data was used to evaluate the speed and memory usage of ClusTCR and other TCR/CDR3 clustering methods. Because ClusTCR is an unsupervised clustering approach (i.e. it does not require additional information about epitope-specificity), the use of unlabeled TCR data is perfectly suitable for this task.

The data set by Emerson et al. (2017) [3] contains a large amount of unique CDR3 sequences and (2 cohorts containing 666 and 120 samples respectively) is therefore well suited for performance benchmarking. The data was downloaded from the immuneACCESS database (https://clients.adaptivebiotech.com/pub/emerson-2017-natgen). From cohort 1, we randomly selected samples and extracted a fixed number of unique sequences to construct 'metarepertoires'. These metarepertoires may therefore contain CDR3 sequences that originate from different repertoire samples. For the performance benchmarking, different metarepertoires were sampled (see table S1). Note that this data was merely used the assess the speed of different clustering algorithms. The extent to which each of these methods is able to cluster together sequences with common epitope-specificity was evaluated using labeled TCR data (see 1.1.1).

1.2 ClusTCR workflow

1.2.1 Overview

ClusTCR is a two-step clustering algorithm that combines the speed of the Facebook artificial intelligence similarity search (Faiss) Python library, combined with the accuracy and flexibility of the Markov clustering algorithm (MCL). During the first clustering step we use Faiss' efficient K-means implementation to rapidly subdivide data sets of CDR3 sequences into superclusters. In the next step, a similarity graph is constructed from the sequences within each supercluster and MCL is applied to identify groups of epitope-specific CDR3 sequences within each graph.

1.2.2 Sequence vectorization

Faiss is optimized for rapid processing of vectors. Therefore, a new embedding must be created for each CDR3 amino acid sequence, in the form of an n-dimensional vector describing its functional properties. We do this by describing the sequence using the physicochemical characteristics of the amino acids within that sequence. The length of the vector n depends on two variables: (1) the length of the largest sequence in the data set and (2) the number of physicochemical properties used to describe the sequence. The total length of the vector is the product of these two values. Sequences are padded to ensure equal length of all vectors. For each physicochemical property (predefined, see later), we produce a vectorized representation of the sequence, where the values of that vector are described by that physicochemical property of the amino acid at every position. To produce the final vector of length n, all individual vectors (one for each physicochemical property) are concatenated. Different combinations of physicochemical properties were tested and the resulting clustering quality and speed were evaluated (for an exhaustive list of the combinations, see fig. S1). All vectors are stored in a single 2D tensor with shape (n, m), where n is the size of a single vectorized sequence and m is the total number of sequences in the input data.

1.2.3 Computing superclusters

Roughly dividing the sequence data into groups of approximate neighbours drastically speeds up the total clustering process by minimizing the number of direct comparisons between sequences. ClusTCR computes a number of superclusters s. This number s corresponds to m/θ, the total number of sequences in the data set (m) divided by the average number of sequences in a supercluster (θ). Although the limited availability of epitope-labeled CDR3 data prevents us from exactly determining an optimal supercluster size, 5000 provided consistent results while maintaining high performance (fig. S2A).
To retrieve the superclusters, an approximate nearest neighbour search is performed using the Faiss library [4]. First, an index is constructed ('trained'), which will be used to efficiently search large sets of CDR3 sequences. During this step $m/5000$ centroids are computed. Next, we use Faiss' efficient K-Means implementation to assign each vector to its most similar centroid (as defined by the Euclidean or L2 distance between the vector and the centroids). For this step, Faiss additionally provides GPU support for CUDA-compatible GPUs.

1.2.4 Sequence hashing to speed up graph construction

Networks are effective structures for representing relationships between objects or processes, such as biomolecular interactions. Likewise, networks can be used to represent the sequence similarity landscape of TCR repertoires. These similarity networks are undirected graphs $G(V, E)$, consisting of a set of nodes or vertices (V) represented by the CDR3 sequences, which are connected by a set of edges (E) representing (dis)similarity between nodes. In the case of amino acid sequences, dissimilarity can be described by an edit distance. Here, edit distance was defined as the Hamming distance (HD) between two strings. Given two strings u and v, the HD between them ($d(u, v)$) is defined as the number of positions where u and v differ. Additionally, HD implicitly assumes that both strings have an identical length. Thus HD only allows substitutions, and no deletions or insertions. The graph describes the adjacency matrix A of pairwise HDs between the CDR3 amino acid sequences:

$$A = \begin{pmatrix} 0 & \cdots & HD_{1,m} \\ \vdots & \ddots & \vdots \\ HD_{m,1} & \cdots & HD_{m,m} \end{pmatrix}. \tag{1}$$

HD = 1 was chosen as a criterion for drawing an edge between two sequences. Hence A will be a binary matrix that can be compressed into a sparse matrix format for efficient memory allocation.

To determine all pairs of CDR3 sequences with HD = 1, a simple hash function was used to convert each sequence into hashes that contain either the odd or even positions of the original sequence. By doing this, only sequences that are assigned to the same hash need to be compared against each other. This drastically reduces the total amount of pairwise comparisons. This hashing method correctly identifies all sequence pairs with a HD of 1, while being much faster than any brute-force comparison method.

1.2.5 Alternative distance metrics

To provide additional flexibility, ClusTCR offers the Levenshtein distance (LD) as an alternative distance metric to determine similar pairs of TCR sequences. LD differs from HD in that it does not assume equal length of strings. Therefore, LD additionally allows deletions and insertions. LD between sequences is calculated using the optimized *Levenshtein* python package. Optimized sequence hashing was not implemented for LD, resulting in an overall slower clustering procedure using this metric.

1.2.6 Graph clustering

For each individual supercluster, a graph is constructed from its CDR3 sequences. The goal is to resolve clusters of sequences with high similarity ($\sim$ epitope-specificity). MCL is particularly appropriate for this task [5]. MCL is a graph clustering algorithm that identifies dense network substructures in a graph or network by simulating stochastic flow. The algorithm performs a random walk on the graph, which is calculated using Markov chains. MCL performs two operations on the stochastic matrix: *expansion* and *inflation*. Expansion is taking the power of the stochastic matrix using the normal matrix product. This determines how much flow is allowed between different regions of the graph. This effect is further amplified by the inflation parameter, taking entrywise powers of the 'expanded' matrix, followed by a rescaling step so that the matrix elements correspond to probability values (i.e. making the matrix stochastic again). Inflation further strengthens current between (already) strong neighbours, while at the same time weakening current between weak or distant neighbours. This process of alternating between expansion and inflation operations is repeated until the graph is partitioned into individual substructures (while paths between substructures no longer exist). We evaluated different thresholds of the expansion and inflation parameters, as illustrated in fig. S2B.

ClusTCR applies the Python 3 implementation of MCL (see github). To further increase performance, ClusTCR also applies multiprocessing to parallelize MCL for different superclusters at the same time.

1.3 Clustering evaluation

1.3.1 Evaluation metrics

Evaluating the quality of clustering results is not a trivial task, even when ground truth labels (in this case, the epitope to which the TCR sequence is specific) are available. Important factors to consider when evaluating TCR clustering results are the coverage of the clustering method, the extent to which epitope-specific sequences are clustered together and how consistent this process is.

The following metrics for the evaluation clustering quality were set out to evaluate these criteria: retention, purity and consistency. Although each metric has its limitations, collectively they offer a solid framework for evaluating an unsupervised clustering approach like ClusTCR.

1.3.2 Retention

Not all input sequences end up in the final clustering results. ClusTCR only considers those sequences that share at least one neighbour with a HD of 1. We define retention as the number of TCR/CDR3 sequences s that participate to any cluster c divided by the size of the data set:

$$\text{retention} = \frac{\sum |s \in c|}{\sum |s|} \tag{2}$$

1.3.3 Purity

Purity is a simple and interpretable metric to describe the extent to which clusters contain sequences specific to a single epitope. Purity is calculated as the fraction of sequences within one cluster targeting the same epitope. For each cluster c, we count the number of sequences s specific for the most common epitope γ, sum the values and divide them by the total number of sequences in any cluster. Formally, purity is computed as

$$\text{purity} = \frac{\sum |\gamma(s \in c) = \gamma_{max(c)}|}{\sum |s \in c|}. \tag{3}$$

Since clustering methods that return very small clusters (i.e. cluster size = 2-3) automatically achieve high purity, we also calculated the fraction of clusters with purity $> 90\%$ as an alternative metric for clustering quality. Thereby, more weight is assigned to larger clusters.

1.3.4 Consistency

Consistency describes the fraction of epitope-specific CDR3 sequences that are assigned to a single cluster. In the case of having multiple epitopes assigned to one cluster, the largest epitope (i.e. with the most sequences specific to it) is given preference. Also, if two or more clusters contain sequences specific for that epitope, the cluster containing the most epitope-specific CDR3 sequences is assigned as the true cluster. Formally, we computed consistency as:

$$\text{consistency} = \frac{\sum |\gamma(s \in c)\gamma_{true(c)}|}{|s|} \tag{4}$$

1.4 Benchmarking clusTCR

To benchmark the performance of ClusTCR, it was compared against existing methods for clustering TCR sequences. For each method, the performance (algorithmic runtime and memory usage) and clustering accuracy were evaluated using the same data. Quality benchmarking was performed using different subsets of the VDJdb, while the performance of the different algorithms was evaluated on artificially large metarepertoires of different sizes (see table S1). Note that all other methods additionally require V gene information, hence this was also provided.

- **iSMART:** The code for the iSMART algorithm was downloaded from github (see source). Although iSMART implements parallel CPU processing, we encountered fatal kernel errors when using this feature. Therefore, iSMART was used at 1 CPU. Additionally, a kernel error appeared when trying to cluster $>$ 200K sequences.

- **GLIPH2:** We downloaded the .centos executable version of GLIPH2 (see source) and installed it in the ClusTCR repository. Each GLIPH2 input TCR included a CDR3 sequence, V gene, subject id and frequency. We wrote a python wrapper that provided GLIPH2 with the appropriate data and executed the algorithm.

- **TCRDist:** Pairwise distances between TCR sequences were calculated using the *tcrdist3* python package (see source). Since *tcrdist3* does not offer native clustering procedures, we assessed the performance of three clustering approaches. First, the clustering procedure described in the original TCRDist article by Dash et al. (2017) [6] was implemented as decribed in the article. In brief, this method involves a 'greedy', fixed-distance-threshold clustering algorithm. The algorithm starts by computing all pairs of sequences that satisfy the predefined distance threshold (edge list). Next, it finds the sequence with the highest degree (i.e. the most neighbors), assigns this as the cluster centre, and removes it and its neighbors from the edge list. This process is repeated until all sequences are clustered. Next, we evaluated two of the most popular clustering algorithms available: K-means and density-based spatial clustering for applications with noise (DBSCAN). For the K-means method, the optimal number for k was determined using the elbow method. For DBSCAN, the trade-off between purity, consistency and retention was evaluated by varying the ϵ paramater. An appropriate cut-off of 100 was used as this reflected best the retention of other TCR clustering approaches. Collectively, these results indicate DBSCAN as the most favorable candidate for clustering the TCRDist-calculated distance matrix. For this reason, DBSCAN was used during benchmarking over the original approach and K-means clustering.

Benchmarking was performed on a 64-bit machine with 15.3 GB RAM, and an Intel® Core™ i7-10875H CPU @ 2.30GHz, running Ubuntu 20.04.2.LTS.

Additionally, to include a baseline performance, ClusTCR was compared against a simple, 'greedy' graph clustering approach. This greedy approach involved constructing a network in which an edge is drawn between each pair that has a HD of 1. Next, each connected component in the graph is assigned to a single cluster. To calculate the the pairwise HDs between CDR3 sequences, a parallelized multiprocessing approach was used.

1.5 Downstream cluster analysis

1.5.1 Cluster features

To allow downstream machine learning applications of the clustering results, clusters were represented numerically by calculating a feature matrix describing multiple properties of the CDR3 amino acid sequences in a cluster.

- **CDR3 length:** Length of CDR3 sequences (number of amino acids) in the cluster. Since ClusTCR handles with an exact similarity criterion of HD = 1, all sequences within one cluster have an equal length.

- **Cluster size:** The number of sequences in a cluster.

- **Cluster entropy:** Cluster entropy describes the positional variation within a cluster. It is calculated as the average information content within a cluster by determining the Shannon entropy at each position (excluding positions 0 and -1) and taking the average across all positions. Additionally, small-sample correction is applied to account for differences in cluster size [7]. Since all sequences within a single cluster have an equal size, multiple sequence

alignment is not necessary. Formally, we compute the average information content R of a cluster:

$$R = \frac{1}{l} \sum_{i=1}^{l} \left(\log_2 20 - (H_i + e_n) \right), \tag{5}$$

where l equals the length of the sequences in the cluster and n represents the number of sequences in the cluster. H_i is the Shannon entropy at a given position i:

$$H_i = \sum_{i=1}^{n} f(x_i) \cdot \log_2 f(x_i). \tag{6}$$

e_n is a correction factor that accounts for differences in cluster size. It is calculated as folows:

$$e_n = \frac{1}{\ln 2} \cdot \frac{s-1}{2n}, \tag{7}$$

where s is the size of the alphabet. Because we are handling amino acid sequences, $s = 20$.

- **Physicochemical features:** ClusTCR calculates the average and variance of following physicochemical features: basicity, hydrophobicity, helicity and mutation stability. Physicochemical features provide a functional encoding of the sequences within a cluster.
- **Generation probability:** Generation probability (P_{gen}) is the probability by which a TCR or CDR3 sequence is generated through the V(D)J recombination process. This probability is estimated through stochastic modelling of V(D)J recombination. P_{gen} values were calculated using the Optimized Likelihood estimate of immunoGlobulin Amino-acid sequences (OLGA) algorithm [8].

1.5.2 Predicting cluster quality

To analyse ClusTCR's clustering output, a binary classification system was constructed that predicted whether a cluster is of high quality or not. Cluster quality was here defined as having at least 0.90 purity. Labels and features were generated by performing ClusTCR two-step clustering on all unique CDR3 sequences from the human TRB sequences in the VDJdb. Purity of each individual cluster was calculated and these purity values were discretized according to the defined purity cut-off. Clusters with purity > 0.90 were assigned 1, every cluster with lower purity was assigned 0. Next, cluster features were calculated according to the procedure described in the previous section. We used Python's scikit learn library to train a random forest classifier consisting of 100 decision trees. This model was fitted to the cluster features and labels. The classifier was evaluated through 10-fold stratified cross-validation. We used the receiver operating characteristic (ROC) and the area under the ROC (AUC) to express the performance of this classification model (fig. S9A). Additionally, the importance of the features towards cluster quality classification was evaluated (fig. S9B).

1.6 Software and documentation

ClusTCR is a licensed software that is available for non-commercial use. The source code is freely available on github (https://github.com/svalkiers/clusTCR). ClusTCR can be installed as a *conda* package. Documentation describing installation instructions and use of ClusTCR can be found at https://svalkiers.github.io/clusTCR/.

2 Tables

Table S1: Runtime (in seconds) of different TCR/CDR3 clustering algorithms at different data set sizes (n = 3). Samples of varying sizes (ranging from 5K up to 1M sequences) were collected by randomly sampling repertoires from a large cohort consisting of 666 TCR repertoire samples [3]. The speed of GLIPH2, iSMART, ClusTCR and TCRDist was evaluated thrice at every sample size. iSMART and TCRDist were unable to process samples containing >200K unique sequences. Note that this data was merely used the assess the speed of different clustering algorithms. The extent to which each of these methods is able to cluster together sequences with common epitope-specificity was evaluated using labeled TCR data (see 1.1.1).
* The runtime of TCRDist indicates pairwise distance calculation only using tcrdist3, no clustering. Only one replicate was collected for 200K input sequences with TCRDist.

n sequences ($\times 10^5$)	**GLIPH2**	**iSMART**	**ClusTCR**	**TCRDist***
0.05	5.28 ± 1.91	2.14 ± 0.18	0.81 ± 0.16	19.84 ± 0.73
0.10	9.88 ± 0.77	5.68 ± 0.4	1.16 ± 0.19	73.11 ± 1.91
0.25	42.78 ± 2.21	24.79 ± 1.06	2.51 ± 0.44	439.91 ± 17.58
0.50	113.47 ± 7.64	92.7 ± 3.74	4.91 ± 0.25	1690.65 ± 64.39
1.00	307.86 ± 19.44	347.91 ± 23.21	13.32 ± 1.17	6450.21 ± 64.19
2.00	937.74 ± 97.32	1298.96 ± 326.05	32.96 ± 7.28	25342.9
5.00	4664.02 ± 598.57	/	117.99 ± 27.74	/
7.50	6833.25 ± 310.21	/	188.4 ± 35.41	/
10.00	9812.83 ± 660.06	/	239.53 ± 30.31	/

3 Figures

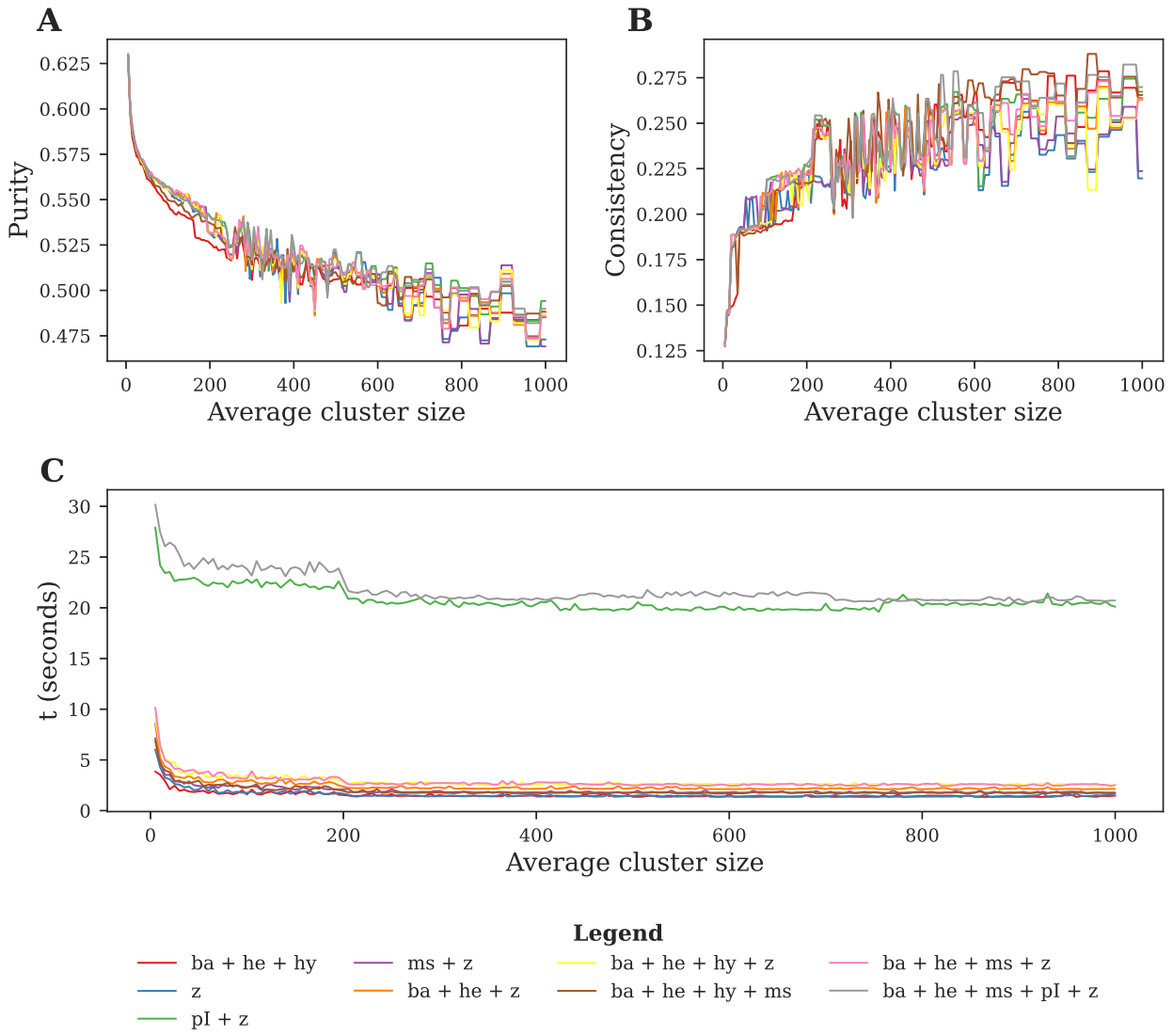

Figure S1: Sequence vectorization for efficient K-means clustering. Different combinations of features were used to numerically encode CDR3 sequences to allow vector clustering with Faiss' K-means implementation. **A.** Influence of different features on cluster purity. There is no single combination of physicochemical properties that outperforms others in terms of cluster purity. **B.** Influence of different features on cluster consistency. **C.** Elapsed time for computing different numerical vectors. Combinations including isoelectric point (pI) take significantly longer to compute. Abbreviations: ba, basicity; he, helicity; hy, hydrophobicity; ms, mutation stability; pI, isoelectric point; z, amino acid z-scores.

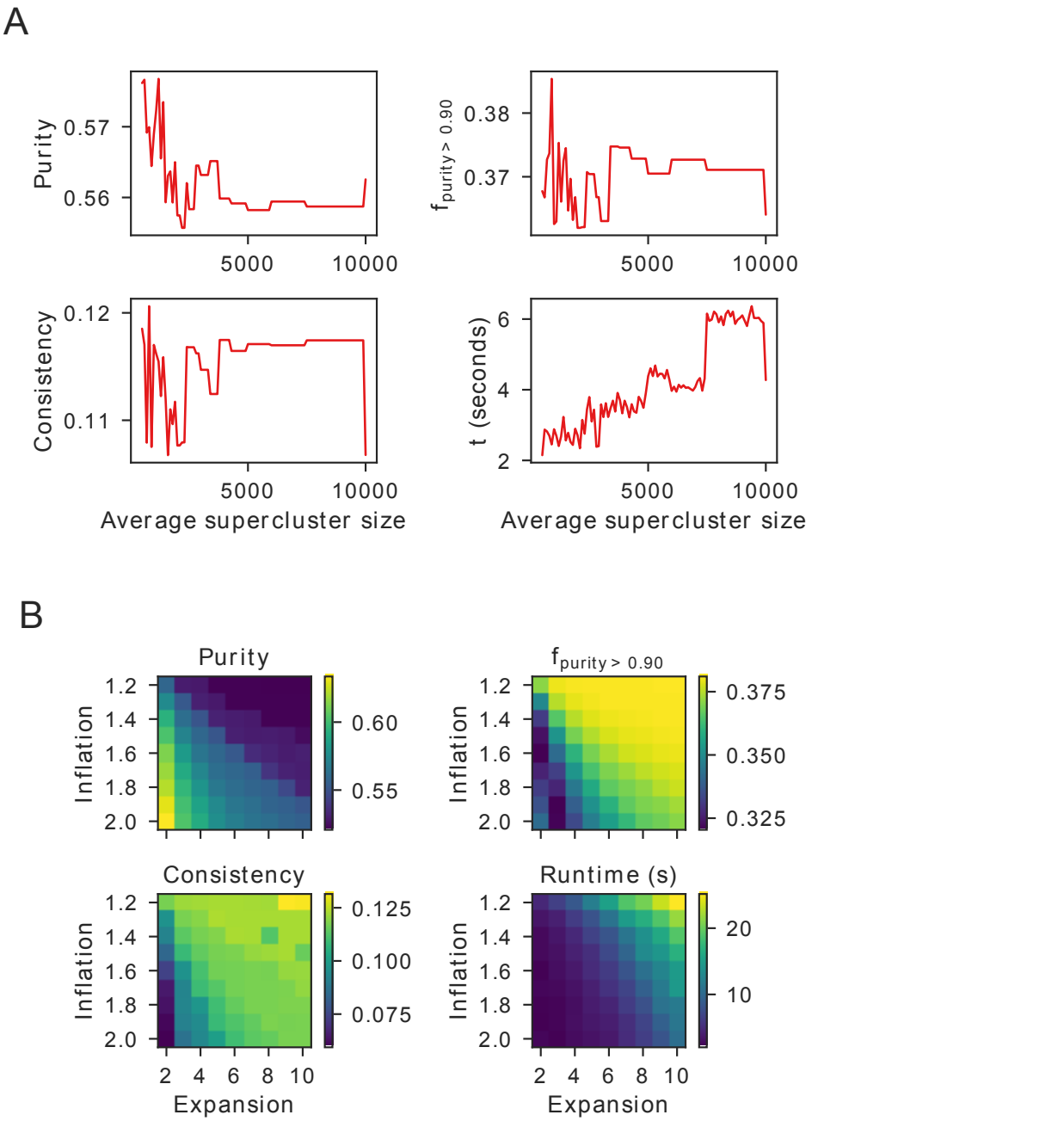

Figure S2: Evaluation of ClusTCR's hyperparameters. **A.** Influence of supercluster size on clustering performance, as defined by a combination of different clustering metrics: purity, fraction of clusters with purity > 90% and consistency. Two-step clustering (Faiss + MCL) was performed at different thresholds for the size of the superclusters computed using Faiss. Smaller supercluster sizes generally result in a higher clustering *purity*. No such pattern is observed for the fraction of clusters with *purity* > 90%. Smaller supercluster sizes also come with larger inconsistencies for the *consistency* metric. However, larger superclusters also require longer computation time. **B.** Evaluation of MCL hyperparameters. Different MCL hyperparameters were used and the clustering quality was evaluated using different evaluation metrics: purity, fraction of clusters with purity > 90% and consistency. A trade-off is observed between purity and consistency. ClusTCR applies a default inflation of 1.2 and expansion of 2. This may not provide the optimal clustering results, but increasing these parameters results in decrease in speed of the algorithm.

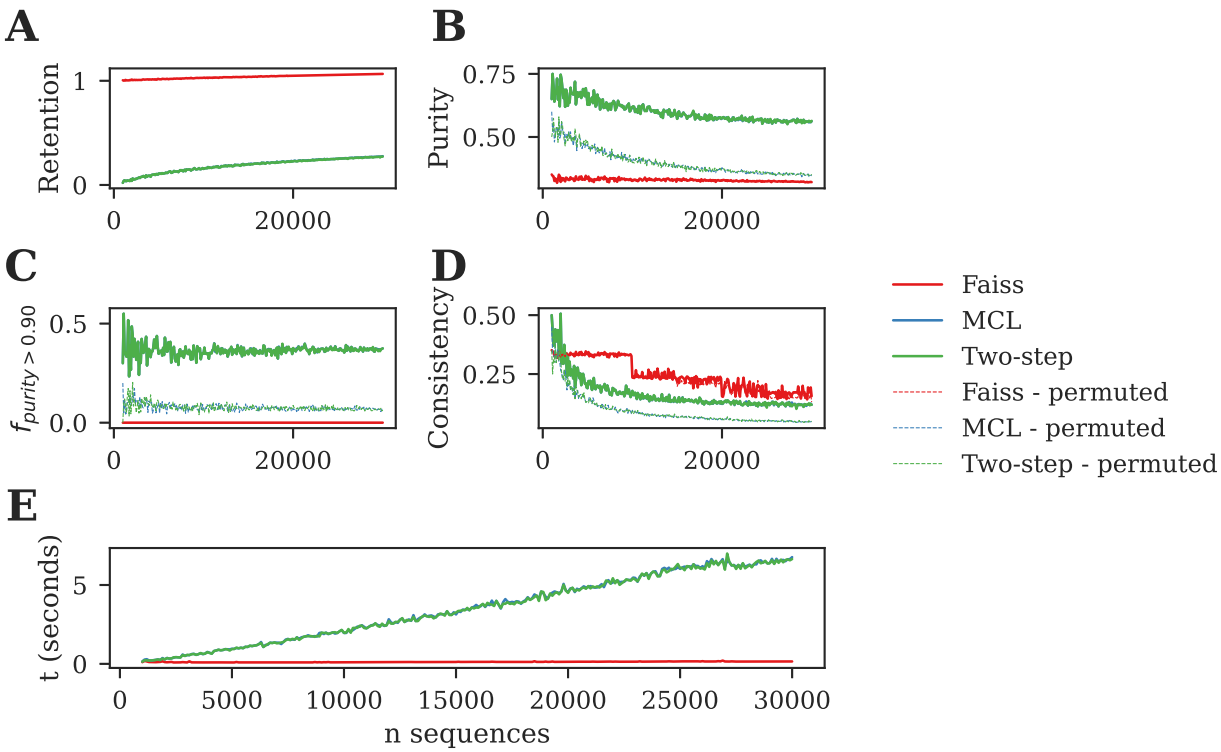

Figure S3: Evaluation of the individual steps in the ClusTCR clustering process illustrates the power of combining broad pre-clustering with detailed secondary clustering. No accuracy is lost when pre-clustering the data using Faiss (as indicated by the overlay of the 'two-step' method over MCL). **A.** Faiss implements K-means clustering and therefore considers all sequences (retention = 1). Retention of MCL and ClusTCR two-step depends on the number of sequences that have an edge in the similarity graph. **B.** High purity is achieved using MCL and the two-step method. **C.** The fraction of clusters with a purity higher than 0.90 remains stable around 37% for MCL and the two-step method. Due to the large predefined cluster size of the Faiss method (5000), none of the clusters reaches a purity > 0.90. **D.** Clustering consistency. The high consistency values of the Faiss clustering method are attributed to the large predefined cluster sizes. **E.** Algorithmic runtime. For datasets of sizes ranging from 1,000 up to 30,000 sequences, MCL and the two-step require equal runtime.

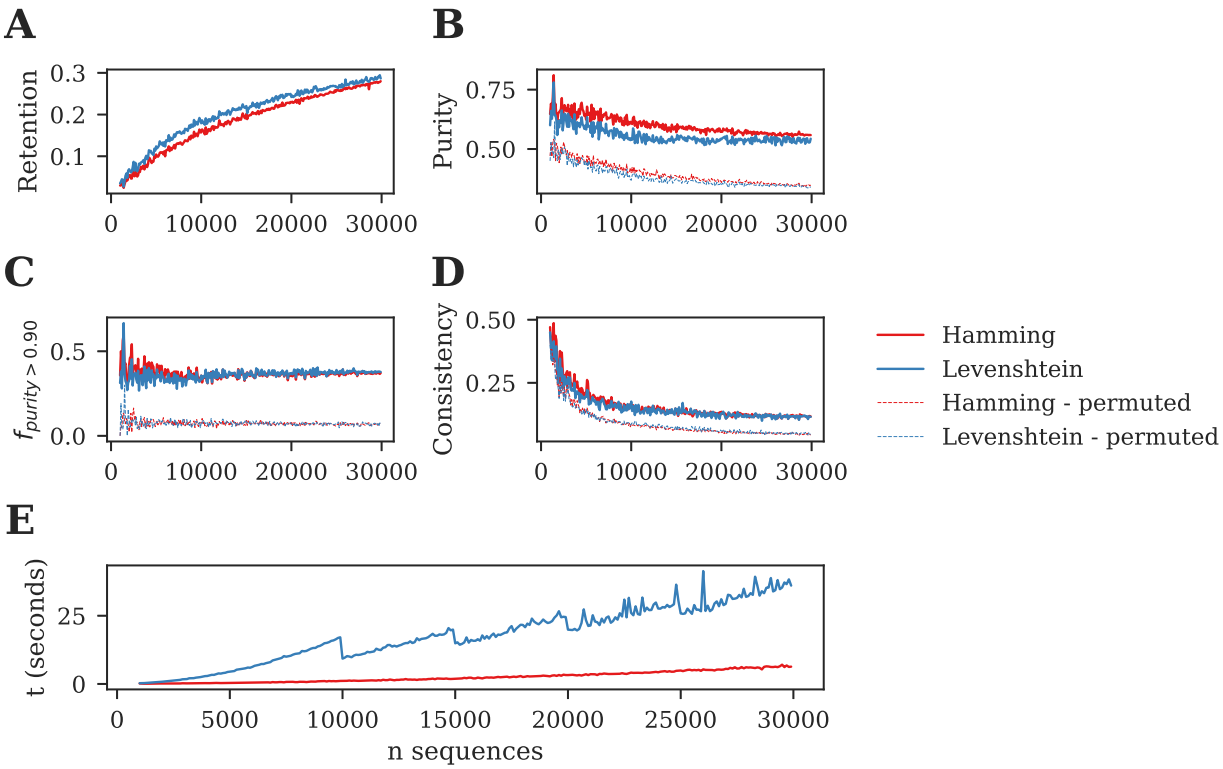

Figure S4: Evaluation of the influence of different (edit-)distance metrics (Levenshtein and Hamming) on the clustering of CDR3β amino acid sequences by ClusTCR. Here, an edit-distance equal to 1 was used, both for Hamming and Levensthein distance. Cluster labels were randomly permuted to provide a baseline, which should reflect a random clustering procedure. **A.** Retention. Cluster retention is marginally higher when using Levensthein distance. **B.** Purity. Slightly higher cluster purity is achieved when using Hamming, compared to Levenshtein distance. **C.** Fraction of clusters with $> 90\%$ purity. Both distance metrics achieve a comparable fraction of clusters with a purity value exceeding 90%. **D.** Consistency. Clustering consistency is equal for both distance metrics.

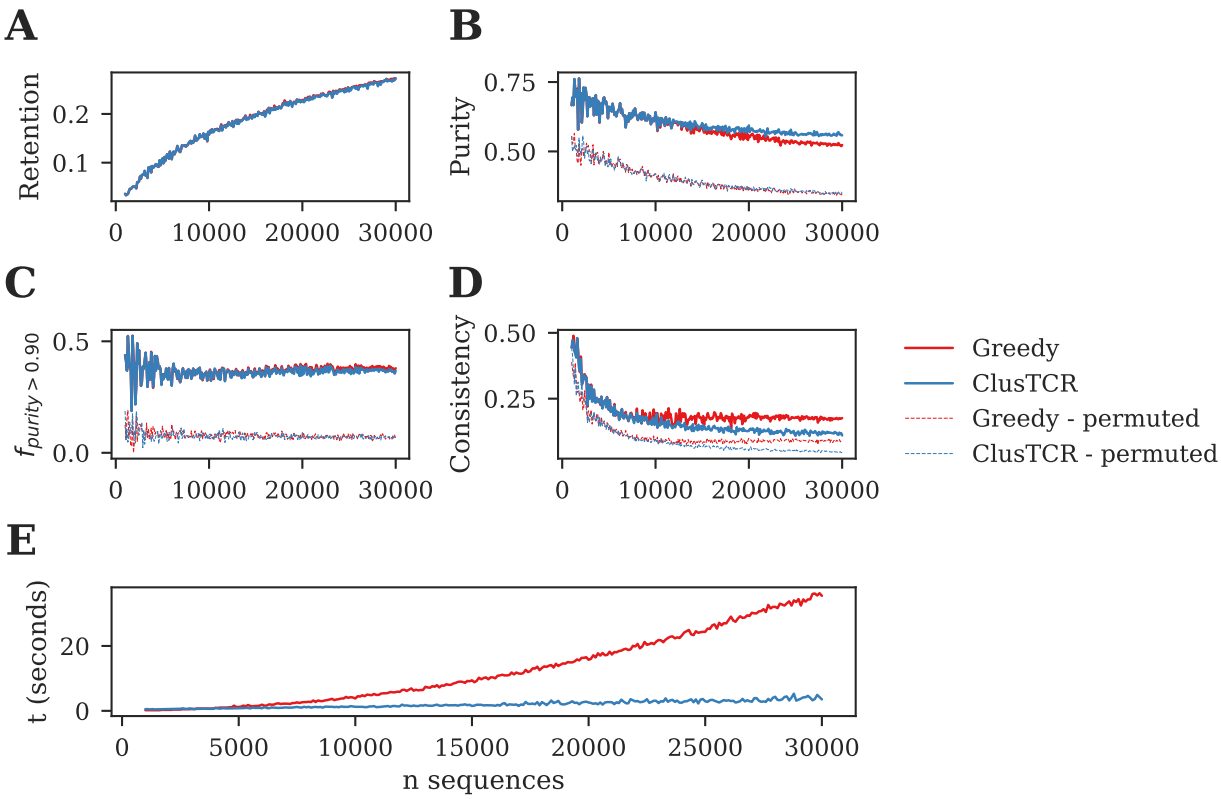

Figure S5: Performance comparison of a simple, Hamming distance-based clustering approach versus ClusTCR. In brief, the simple clustering method consisted of the following steps: pairwise Hamming distances were calculated and sequence pairs with a maximum Hamming distance of 1 were added to a graph. Next, a greedy clustering approach was used to assign all interconnected nodes to the same cluster. Both methods were evaluated on a range of samples from varying size (ranging from 1,000 up to 30,000 sequences). Each sample consisted of a number of CDR3β sequences with known antigen specificity. Clustering performance of both was measured using different metrics. Cluster assignments were randomly permuted to provide a baseline comparison (dotted lines). **A.** Retention was comparable between both methods. **B.** ClusTCR achieved slightly higher overall purity. **C.** The fraction of clusters with purity $> 90\%$ was equal for both methods. **D.** The greedy clustering approach achieved slightly higher consistency over ClusTCR. **E.** ClusTCR is significantly faster than the simple clustering approach, and scales up better to larger sample sizes.

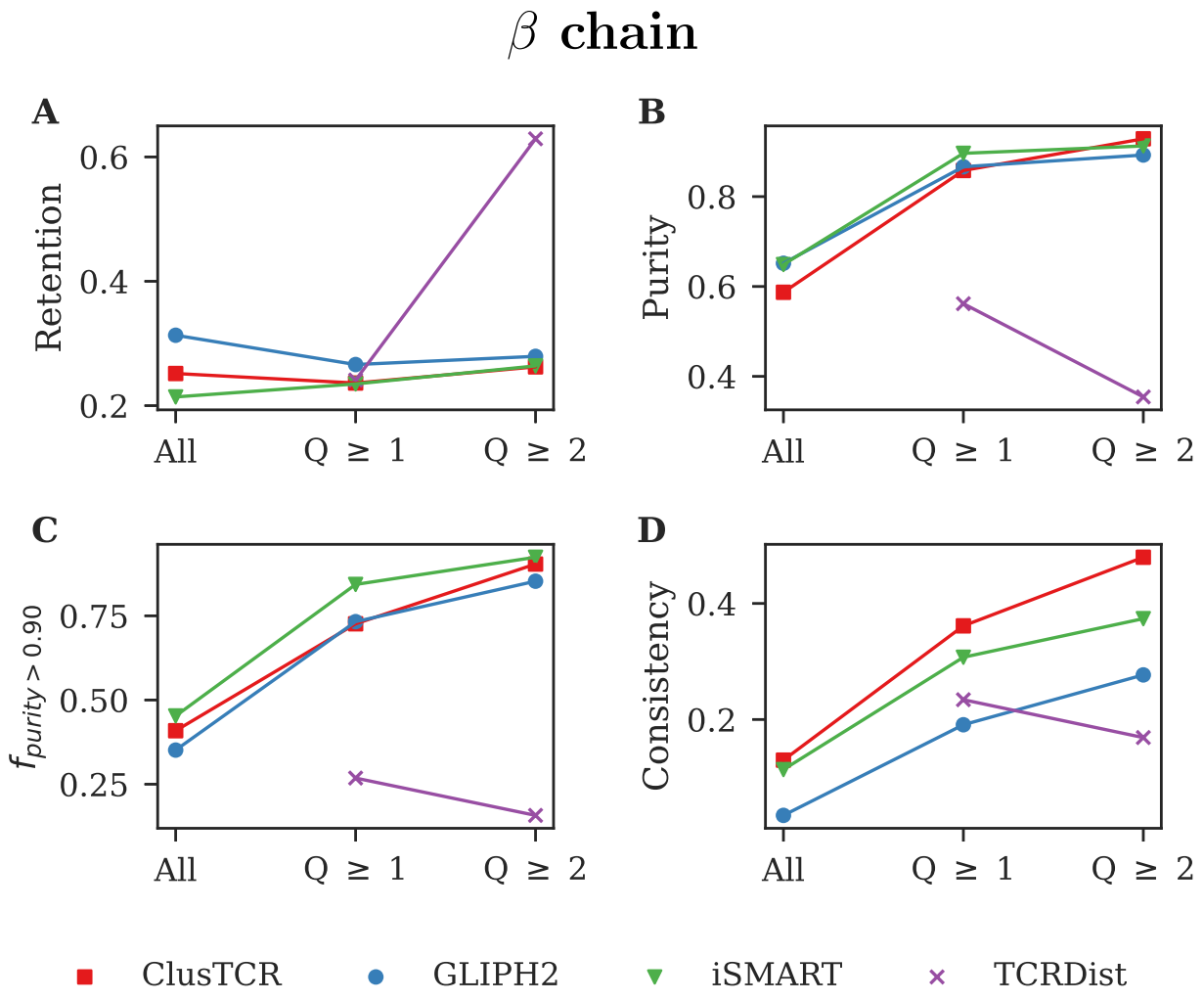

Figure S6: Evaluation of the ability of different clustering methods to group TCRβ/CDR3β sequences with common epitope specificity. Methods used for benchmarking included: GLIPH2, iSMART and TCRDist. Clustering quality was evaluated by means of four metrics: retention, purity, the fraction of clusters with purity > 0.90 and consistency. Data used for benchmarking included three subsets of VDJdb with varying annotation quality. The DBSCAN clustering algorithm was used to cluster CDR3 sequences from a TCRDist distance matrix. For data sets exceeding 10,000 sequences, it was not possible to calculate pairwise distances using the regular TCRDist algorithm (hence TCRDist was only evaluated for the Q $\geq$ 1, Q $\geq$ 2 subsets). **A.** GLIPH2 achieved the highest retention on the complete VDJdb (0.31), while ClusTCR cover about 25% of the total number of input sequences. iSMART shows the lowest retention value for the complete VDJdb (0.21). TCRDist achieves substantially higher retention for the small, high quality subset. **B.** On the complete VDJdb ClusTCR achieves slightly lower average cluster purity compared to iSMART and GLIPH2. ClusTCR, GLIPH2 and iSMART achieved similar purity on both subsets (Q $\geq$ 1 and Q $\geq$ 2), while TCRDist scored consistently lower. **C.** Percentage of clusters with purity >90% was roughly equal for ClusTCR, GLIPH2 and iSMART, with iSMART achieving slightly higher scores for this metric on the complete data set and Q $\geq$ 1 subset. TCRDist-based clustering resulted in a much smaller fraction of clusters with high purity. **D.** Consistency was highest for ClusTCR across all 3 data sets.

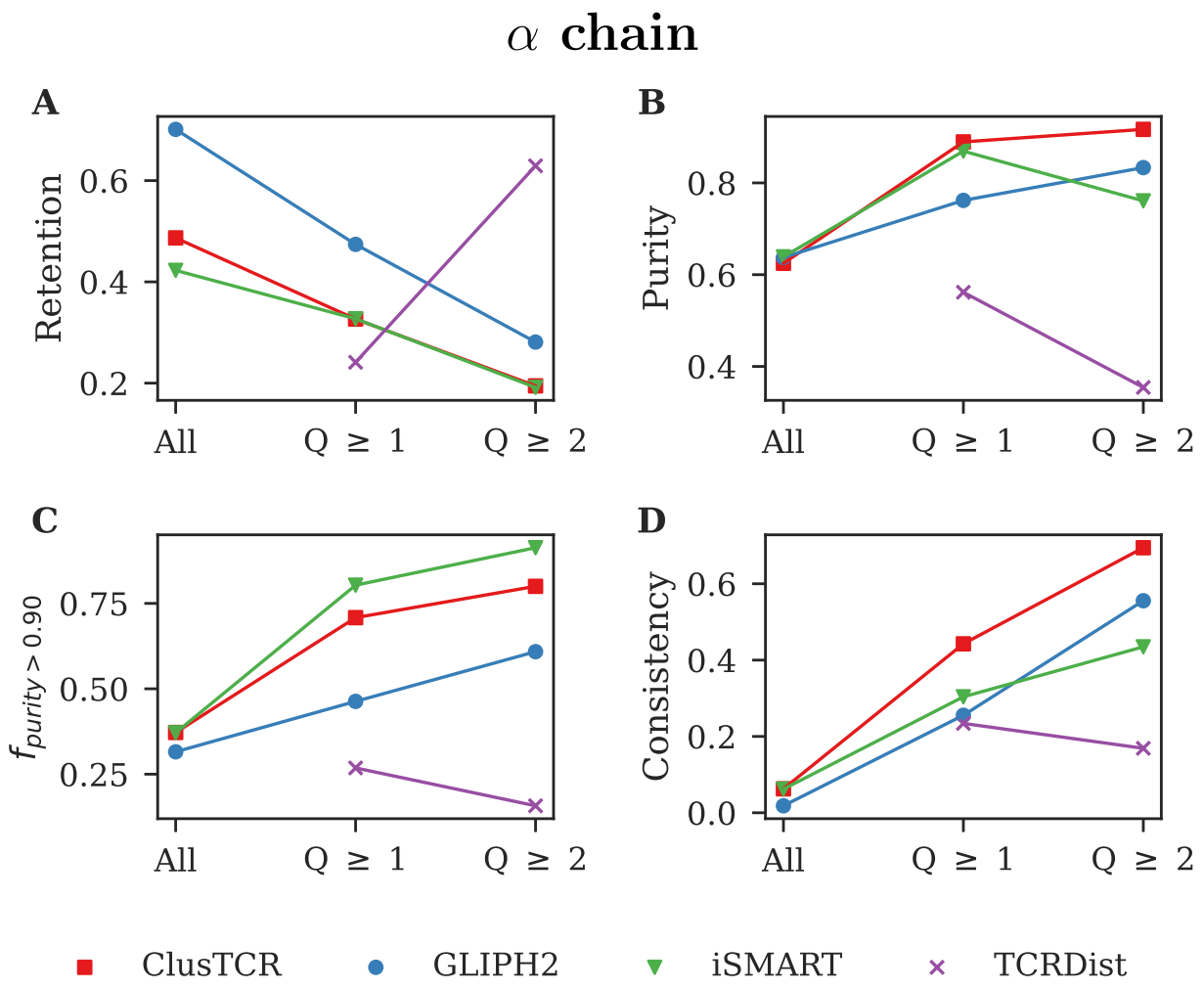

Figure S7: Evaluation of the ability of different clustering methods to group TCRα/CDR3α sequences with common epitope specificity. Methods used for benchmarking included: GLIPH2, iSMART and TCRDist. Clustering quality was evaluated by means of four metrics: retention, purity, the fraction of clusters with purity > 0.90 and consistency. Data used for benchmarking included three subsets of VDJdb with varying annotation quality. The DBSCAN clustering algorithm was used to cluster CDR3 sequences from a TCRDist distance matrix. For data sets exceeding 10,000 sequences, it was not possible to calculate pairwise distances using the regular TCRDist algorithm (hence TCRDist was only evaluated for the Q $\geq$ 1, Q $\geq$ 2 subsets). **A.** GLIPH2 achieved the highest retention on the complete VDJdb set (0.70) and the Q $\geq$ 1 subset (0.47), while ClusTCR covered about half and iSMART 42% of the total number of input sequences. **B.** On the complete VDJdb all methods achieve near identical cluster purity. ClusTCR scored slightly better on both smaller subsets. TCRDist scored consistently lower than all other methods. **C.** Percentage of clusters with purity >90% was roughly equal for ClusTCR, GLIPH2 and iSMART, with iSMART achieving slightly higher scores for this metric across the smaller subsets. TCRDist-based clustering resulted in a much smaller fraction of clusters with high purity. **D.** Consistency was highest for ClusTCR and iSMART for the large set of sequences. ClusTCR achieves the highest score for this metric across both subsets.

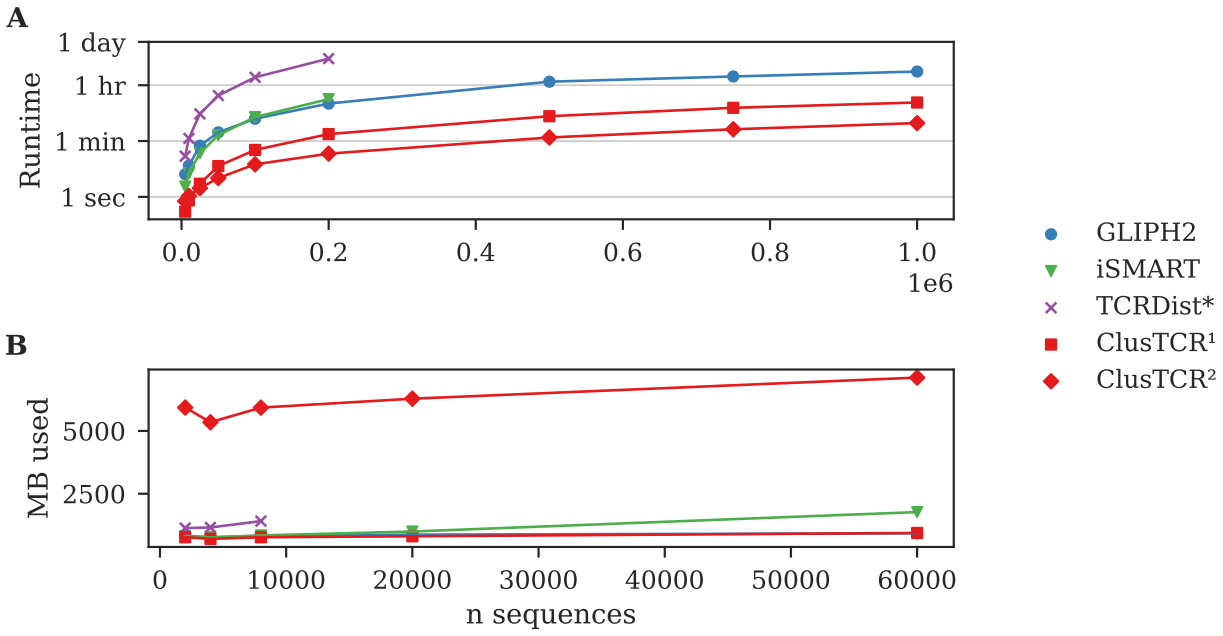

Figure S8: **A.** Average runtime of different TCR/CDR3 clustering methods at various sample sizes. Input data was generated by randomly sampling repertoires from a large cohort of 666 individuals. At 10^6 sequences, ClusTCR provides a $> 50\times$ speed improvement (with an Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz (using 16 CPUs) over other methods. Exact runtime values are indicated in table S1. **B.** Memory profiling of different TCR clustering methods. iSMART and regular ClusTCR (no multiprocessing) use slightly less memory compared to GLIPH2 and TCRDist. With multiprocessing enabled, ClusTCR utilizes significantly more memory. ¹ ClusTCR without multiprocessing (1 CPU); ² ClusTCR with multiprocessing (16 CPUs); * The runtime or memory usage of TCRDist only includes the computation of pairwise distances, not clustering. TCRDist distances were calculated using a sparse, chunked method.

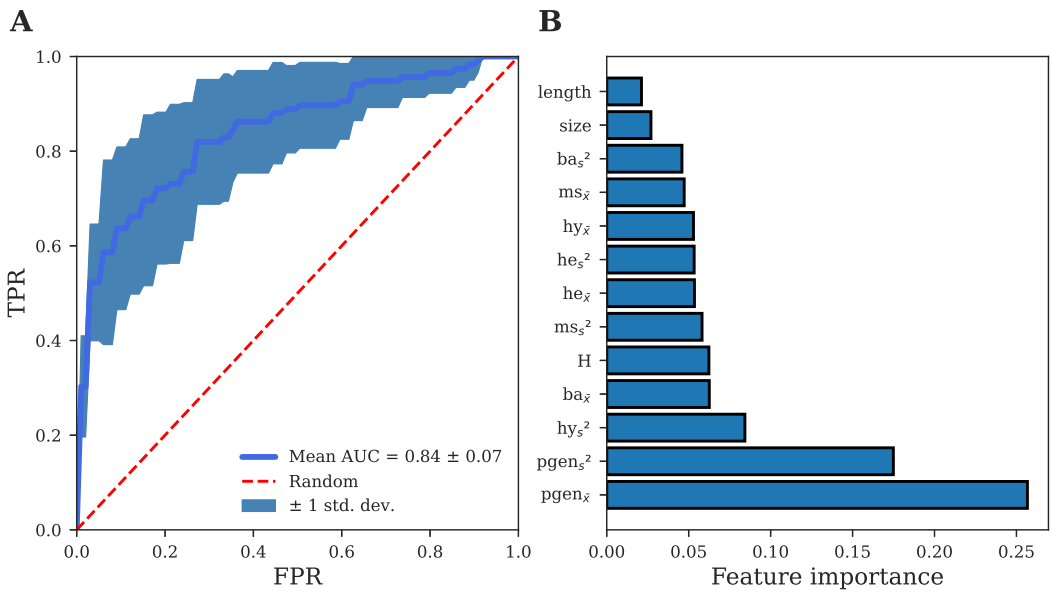

Figure S9: **A.** Cluster quality prediction model ROC curve. A classification model was built that identifies a good cluster from a bad cluster. Good clusters were defined as those clusters with purity > 0.90. For every cluster, a series of features was calculated, which included length of the amino acid sequences, the size of the cluster, average physicochemical properties, cluster entropy, and average generation probability. A random forest classifier was trained to identify clusters with high purity (> 0.90), based on the calculated cluster features. The model was evaluated through 10-fold stratified cross-validation. The model had an average area under the curve (AUC) of 0.84, with a standard deviation of 0.07. This model is provided as the default model for cluster quality prediction in ClusTCR. **B.** Importance of different cluster features in predicting cluster quality. Abbreviations: ba, basicity; he, helicity; hy, hydrophobicity; ms, mutation stability; pgen, generation probability.

References

[1] Mikhail Shugay, Dmitriy V Bagaev, Ivan V Zvyagin, Renske M Vroomans, Jeremy Chase Crawford, Garry Dolton, Ekaterina A Komech, Anastasiya L Sycheva, Anna E Koneva, Evgeniy S Egorov, et al. Vdjdb: a curated database of t-cell receptor sequences with known antigen specificity. *Nucleic acids research*, 46(D1):D419–D427, 2018.

[2] Dmitry V Bagaev, Renske MA Vroomans, Jerome Samir, Ulrik Stervbo, Cristina Rius, Garry Dolton, Alexander Greenshields-Watson, Meriem Attaf, Evgeny S Egorov, Ivan V Zvyagin, et al. Vdjdb in 2019: database extension, new analysis infrastructure and a t-cell receptor motif compendium. *Nucleic Acids Research*, 48(D1):D1057–D1062, 2020.

[3] Ryan O Emerson, William S DeWitt, Marissa Vignali, Jenna Gravley, Joyce K Hu, Edward J Osborne, Cindy Desmarais, Mark Klinger, Christopher S Carlson, John A Hansen, et al. Immunosequencing identifies signatures of cytomegalovirus exposure history and hla-mediated effects on the t cell repertoire. *Nature genetics*, 49(5):659–665, 2017.

[4] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[5] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2000.

[6] Pradyot Dash, Andrew J Fiore-Gartland, Tomer Hertz, George C Wang, Shalini Sharma, Aisha Souquette, Jeremy Chase Crawford, E Bridie Clemens, Thi HO Nguyen, Katherine Kedzierska, et al. Quantifiable predictive features define epitope-specific t cell receptor repertoires. *Nature*, 547(7661):89–93, 2017.

[7] Thomas D Schneider, Gary D Stormo, Larry Gold, and Andrzej Ehrenfeucht. Information content of binding sites on nucleotide sequences. *Journal of molecular biology*, 188(3):415–431, 1986.

[8] Zachary Sethna, Yuval Elhanati, Curtis G Callan Jr, Aleksandra M Walczak, and Thierry Mora. Olga: fast computation of generation probabilities of b-and t-cell receptor amino acid sequences and motifs. *Bioinformatics*, 35(17):2974–2981, 2019.