Refinement of a statistical model for antibody humanization

Master Thesis

Pilar Calvo Príncipe

September 2019

Supervisor: Pierpaolo Bruscolini

Abstract

Antibody therapeutics are usually developed in animals, commonly in mouse, and then they are humanized by Complementary-Determining Regions (CDR) grafting. Several times, this key step fails because of the lack of stability and/or functionality of the new antibody and the immunogenicity. Different efforts have been made to improve the humanization; many of them rely on the quantification of the humanness of the variable region. Previous researches have reported a statistical approach based on a Multivariate Gaussian (MG) model which successfully distinguishes between human and murine sequences.

However, the strength and weaknesses of this model have not be properly studied yet, and a full understanding of where its efficacy comes from, and how the model could be refined to improve it, is still lacking.

Here, some tests and attempts of refinement of the MG model are performed to understand if the resulting interaction map is related to the protein's structure, to see if the predictions can be improved by introducing some score corrections and to find out which are the most relevant columns and how the number of sequences in the learning dataset affects the classification capabilities.

The results that we obtain are somewhat surprising: We show that no strong correlation between the contact map and the emerging interactions between pairs of residues from the model was found, the classification is still good when a much smaller learning dataset is included and gap corrections do not affect the predictions power. We also present different indicators to identify key positions and infer the Kullback-Leibler divergence as the best one.

Keywords

Antibody humanization| Statistical sequence analysis | Multivariate Gaussian model | Residue – residue covariance | Maximum-entropy modeling | Variable domain | Machine learning | ROC curve

Table of Contents

List of Figures

List of Tables

Introduction

Monoclonal antibodies (mAbs) are one of the most important classes of therapeutics with a fast-growing market due to the potential application in a large number of diseases. However, there still are some limitations in the process of antibody humanization, which includes the lack of efficient humanization methods and the high incidence of unwanted immune responses. Over the past decades, antibody-based drugs have repeatedly proven their efficacy and increasing importance in a wide range of conditions, including viral and bacterial infections, auto-immunity and inflammation, as well as the induction of anti-tumor responses. As antibody technologies have evolved, the number of patent applications relating to antibodies has increased dramatically over the past 20 years [5].

It is estimated that world-wide sales of this type of drugs will be nearly $125 billion by 2020 and there are currently hundreds of antibody-based products in clinical development [1]. However, such development is a long and difficult process, prone to fail at different stages. This seriously limits the widespread and repeated application to treat many diseases. Due to this concern, there is an increasing demand for more predictive preclinical models to minimize failures in following clinical phase.

One of the key steps in the preclinical stage is antibody humanization. New antibodies are generally developed in animal models (most commonly in mouse), but they are limited by both the high incidence of unwanted immune responses and the lack of adequate effectors function. Thus, it is fundamental to introduce mutations on the murine antibody to produce a more human-like sequence in order to decrease the immunogenicity [10].

Antibodies are globular plasma proteins and produced by cells of the immune system known as B-lymphocytes. They consist of four polypeptide chains: two heavy chains and two light chains joined by disulfide bonds to form a "Y" shaped molecule (see Fig. 1). There are two types of light chain; kappa (κ) chain and the lambda (λ) chain. The part of the antibody known as antigen-biding fragment (Fab) is the one that recognizes the antigen; while the rest of the structure, called fragment crystallizable region (Fc region) interacts with other proteins, such as phagocytes, to activate the immune system.

The Fab region is is composed of one constant and one variable domain from each heavy and light chain of the antibody. The variable regions of antibodies consist of 4 framework regions (FRs), which are very conserved and 3 CDRs (also known as hypervariable regions) (see Fig. 2). In total, there are six loops CDR: H1, H2, H3 of the heavy chain variable domain (VH) and L1, L2, L3 of the light chain variable domain (VL).

Thanks to the somatic recombination or V(D)J recombination of the immunoglobulins, a huge number of antibodies with unique variables regions can be generated. This variable region is encoded in three pools of gene segments (or subgenes) and exons: one encodes κ light chains, one λ light chains and one heavy chains. These subgenes are called variable (V), diversity (D) and joining (J) segments. By randomly combining gene segments that code for VL and VH regions, hundreds of different light chains and thousands of different heavy chains can be made and then pair to form antibodies with millions of different antigen-binding sites [16].

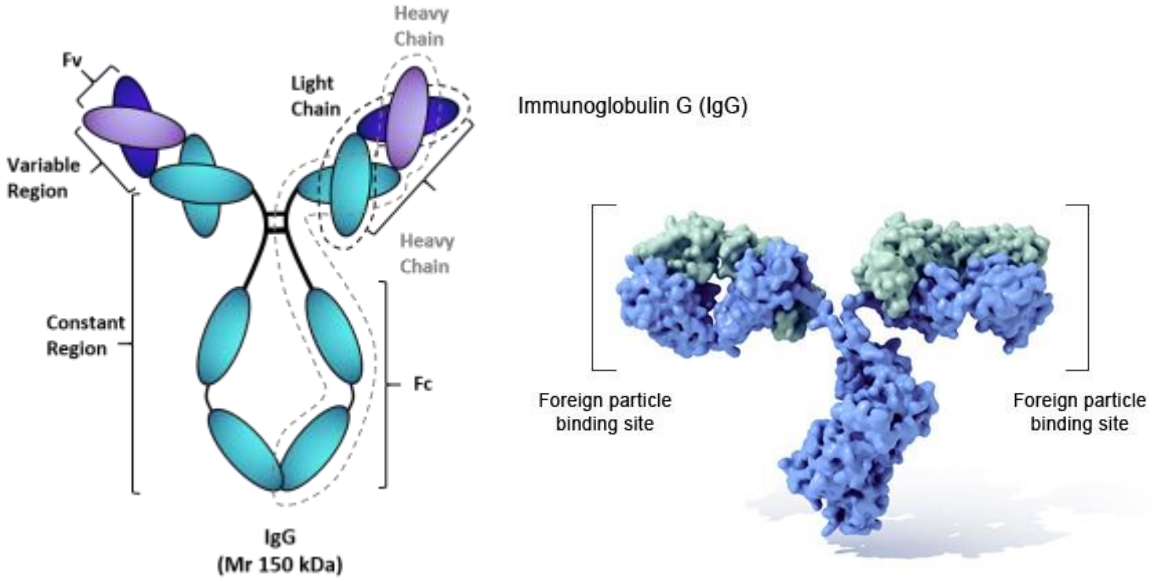

Figure 1. Antibody structure

Left figure retrieved from Topin, I. (2018, November 13). Monoclonal antibodies – all you need to know about antibody generation (https://www.tebu-bio.com/blog/2018/11/13/monoclonal-antibodies-all-you-need-to-know-about-antibody-generation/). Right figure retrieved from Darling, D. Immunoglobulin. (https://www.daviddarling.info/encyclopedia/I/immunoglobulin.html)

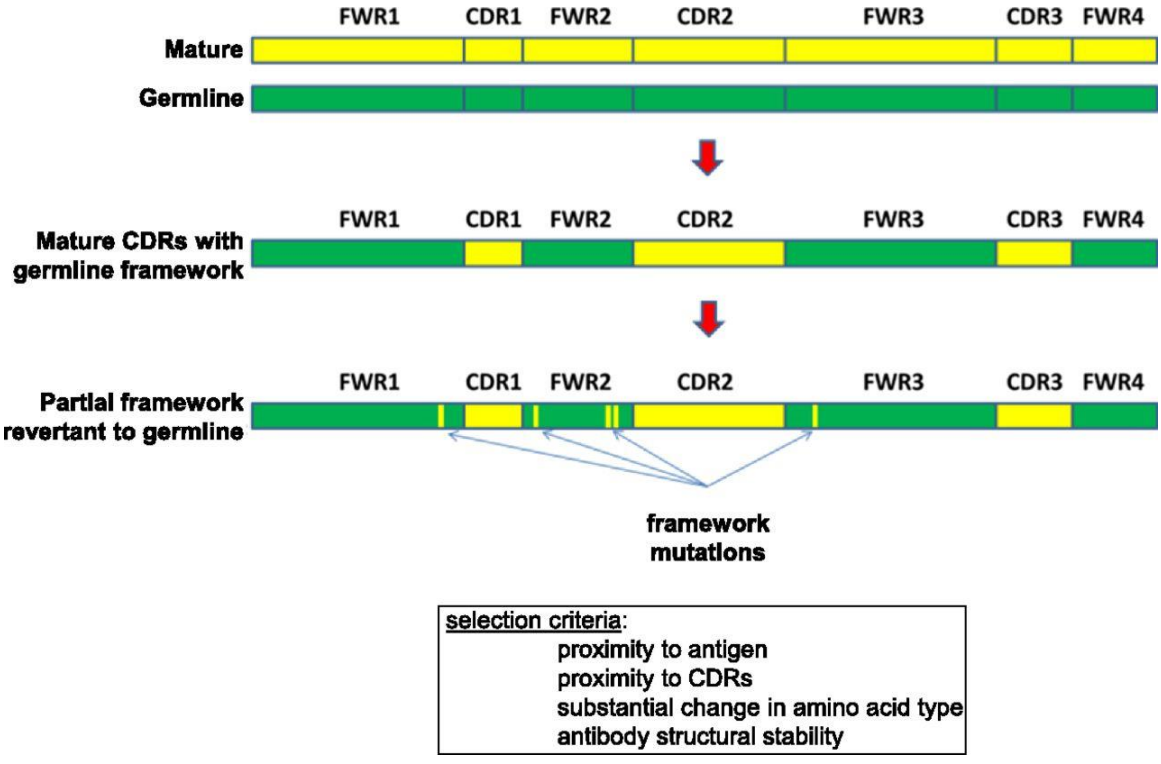

Figure 2. CDR grafting procedure

Figure retrieved from Georgiev, I. S., Rudicell, R. S., Saunders, K. O., Shi, W., Kirys, T., McKee, K., … Kwong, P. D. (2014, February 1). Antibodies VRC01 and 10E8 Neutralize HIV-1 with High Breadth and Potency Even with Ig-Framework Regions Substantially Reverted to Germline (https://www.jimmunol.org/content/192/3/1100)

CDR1 and CDR2 are found in the V segment and CDR3 includes some of V, all of D and J regions. The CDRs are in direct contact with the antigen, whereas the FR regions support the binding of the CDR to the antigen and help to stabilize the overall structure of the variable domains. To improve its stability, the FR regions have less variability compared to the CDR [17].

Nowadays, one of the initial and well-known methods for humanization involves Complementary-Determining Regions (CDR) grafting (see Fig. 2), where a functional antigen-binding site from a non-human “donor” antibody is grafted onto a human “acceptor” antibody, meaning CDRs are combined with human Framework Regions (FR) sequences [5]. In this approach, the hope is that the combination of human FRs with the original murine CDRs will result in an antibody (Ab) that still maintains its stability and activity, but it is tolerated by the human immune system. Most of the times, it is not entirely successful and researches must try further mutations, until an antibody with the desired properties is identified. In conclusion, available methods are time demanding and their predictions are hard to assess [6].

These humanization techniques also can lead to a loss in antibody affinity and/or specificity because of inaccurate definition of the CDR sequences, incorrect choice of the human framework scaffold used for loop grafting or wrong identification of residues from different species.

Different attempts have previously been made to determine a humanness score of the variable region sequences of antibodies such as H-score, germinality index, G-score and T20 score [11-14]; which can be a helpful tool during the antibody drug development process. For a rational design of a humanized sequence, some reliable quantity indicating how much a given sequence is human-like is mandatory.

In general, such scores are based on "one-site" properties, as for instance, the similarity of the given sequence to sequences from a human dataset, where the "similarity" is defined in terms of the number of mutations (i.e., the "Hamming distance") between sequences. However, the necessity of "backmutations" in the humanization pipeline, from a more human-like sequence backwards towards the original sequence, to cope with stability/aggregation/immunogenicity problems, suggests that correlations between pairs of residues at different positions should be taken into account, and correlated mutations at pair of sites, instead of independent ones dictated just by the similarity, should be performed in order to improve the humanization process.

This prompts for the study of the probability distribution of the human sequences, a task that recently has been addressed in the field of structural biology, when trying to infer a protein contact-map from the multiple sequence alignment of a family of similar proteins. Indeed, several global statistical inference approaches have emerged in the last years to predict residue contacts from sequence data: direct-coupling analysis, sparse inverse covariance estimation (PSICOV) and algorithms based on pseudo-likelihood maximization [7,8,9,15].

In every case, exact inference approaches are time demanding; an alternative way was proposed based on a Multivariate Gaussian Modeling, in which and practically out of reach; the discrete amino acid variables are replaced by continuous Gaussian variables [3,4]. The statistical model is a multivariate Gaussian distribution whose parameters are the mean and the covariance $N(\mu, \Sigma)$.

The input is a multiple sequence alignment (MSA) consisting of M homologous sequences of length L and is converted into a M x (QxL) matrix, being Q the 20 possible different amino acids. The residues are defined by a binary alphabet {0, 1}, 1 if the amino acid is present and 0 if not (thus gaps are represented as all 0) (Figure 3). In that way, each position of the MSA is defined by small real-valued vectors.

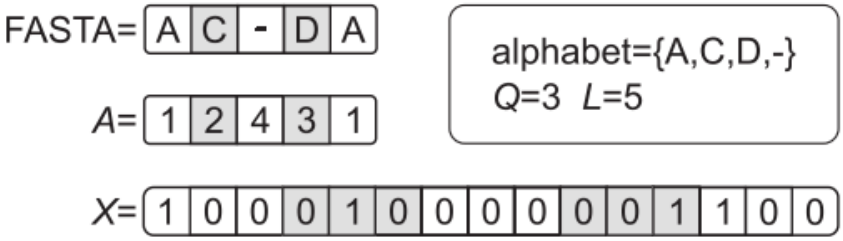

Figure 3. Binary representation of a sequence

Example of the encoding of a sequence is fasta format to it binary representation. For simplicity, only an alphabet with Q=3 is considered. In this research, Q=20 and L=298. Figure retrieved from Baldassi, C., Zamparo, M., Feinauer, C., Procaccini, A., Zecchina, R., Weigt, M., & Pagnani, A. (2014). Fast and Accurate Multivariate Gaussian Modeling of Protein Families: Predicting Residue Contacts and Protein-Interaction Partners. PLoS ONE, 9(3).

Being M the total number of sequences, each sequence denoted as x, the empirical average and covariance are:

$$\bar{x_\iota} = \frac{1}{M} \sum_{m=1}^{M} x_i^m \, w_m$$

$$C_{ij} = \frac{1}{M} \sum_{m=1}^{M} (x_i^m - \bar{x_\iota}) \, (x_j^m - \bar{x_\jmath}) \, w_m$$

where $w_m=1/n_m$ represents the weight of each sequence.

$$n_m = \sum_{l=1}^{M} \theta(\vartheta_{lm} - L\Omega)$$

n_m is the number of sequences whose similarity ϑ_{lm} with sequence m is above the threshold $L\Omega$. The threshold parameter Ω is determined by the value that maximizes the Frobenius norm of C_{ij}.

However, C_{ij} is typically not invertible due to the low number of sequences. To estimate proper parameters, a Bayesian inference method is used, which needs the introduction of a prior distribution over μ and Σ. It is assumed that each of the M sequences in the databases is drawn from a normal distribution and a Normal Inverse Wishart prior distribution for the parameters μ,Σ, so a final posterior parameter are derived as:

$$\langle \mu \rangle_{post} = \lambda \eta + (1 - \lambda)\bar{x}$$

$$\langle \Sigma \rangle_{post} = \lambda U + (1 - \lambda)\bar{C} + \lambda(1 - \lambda)(\bar{x} - \eta)(\bar{x} - \eta)^T$$

where η and U are the mean and covariance estimates of a priori uniformly distributed sample and the parameter λ determines the contribution of the prior.

Based on this model, a new method using a multivariate Student distribution was developed to define a "humanness score" (called MG score) that is able to distinguish between human and murine variable regions [18].

$$p(y|X) = t_N \left(\frac{M}{1 - \lambda} + 2, \langle \mu \rangle_{post}, \left(1 + \frac{1 - \lambda}{M} \right) \langle \Sigma \rangle_{post} \right)$$

being y any given sequence and X the database of sequences. The MG score is defined as the logarithm of this probability density.

The results obtained were very promising, since the method outperformed the others in classification, and provided reasonable results in the humanization task. However, at present it is not clear if the "interaction" (i.e. the precision matrix Σ^{-1}) found have a structural meaning, which are the most important columns or the most important interactions or how the number of sequences in the learning dataset affects the results. The main objective of the project is to analyze in more detail the above issues.

Hypothesis and objectives

Correlation between the contact map and model covariance

One of the most important issues in antibody humanization is to produce a peptide with a solid and stable structure. The prediction of the future humanized antibody's structure and the knowledge of which residues are in contact are very valuable to be successful in the process.

On the other hand, assuming that the human antibody sequences are extracted from a Multivariate Gaussian Distribution implies that the inverse covariance matrix Σ^{-1} plays the role of a kind of interaction between pairs of residues of different type at different positions. In the MG approach, such interactions are inferred from the correlations observed between pairs of residues, and the latter might be due to very different causes: physical (i.e. structural) proximity, interaction with the antigen (epitope), phylogenic rules (involving the way the antibodies sequences are generated at the gene level), etc.

So, it is not clear how much structural information is contained in the inferred Σ matrix and if the latter can be understood in terms of the protein contact map. Thus, the first specific objective is to study the correlations between the covariance or interaction matrix, with the average antibody contact maps, i.e. the matrix of contacts between residues in known antibody structures, to see if and how much the inferred interaction matrix reproduces the physical interactions.

Identification of key positions

The second aim consists of discovering which positions from the MSA are more relevant. It is reasonable to think that the columns in the alignment do not equally contribute to the performance of the model in the classification or humanization tasks. Actually, since the latter is difficult to assess quantitatively (the quality of a humanized sequence is ultimately determined just by the experiments), we focus on the former task.

Therefore, the goal is to identify how many and which residues in the sequences are the most relevant to discriminate between the human and murine classes. To do so, we consider a series of quantities that could explain the relevance of the different columns in the classification task. The first of these indicators that we consider is symbol entropy.

It is expected that the columns in highest entropy would play the most important role in the prediction of the MG model. Asti *et al*. [3] eliminated columns progressively from the highest entropy (variability) to the lowest entropy and observed that such predictive power stayed constant until only the 60 more variable columns were used. The same method will be followed.

Other quantities that we will consider are the difference between murine and human distributions and the average intensity of the interaction on a certain position.

Study of the importance of the dataset

The next purpose is to study the classification performance of the model as a function of the size of the learning database. Nowadays, the number of antibody's sequences available is still limited. Moreover, one of the issues seen in the model is a possible overfitting of the learning dataset [18], since the number of sequences is very low, so it is appealing to study how the predictive power of the MG model changes with the size.

On the one hand, the aim is to find the number of sequences of the variable domain needed to successfully separate the two categories. It is expected that the more sequences are used, the better the classification performance will be as more information is gathered.

On the other hand, light and heavy chains repertoires are usually only separately available because they are translated into different mRNA molecules. Consequently, it is difficult to match both domains and a huge number of sequences were removed in the current VHVL learning dataset to avoid false partnerships [18]. The next objective is to study how well the model behaves with larger datasets consisting of only VH or VL regions.

MG-score correction

The MSAs that have been used in the model, and mostly any MSA, contain numerous gaps, which could affect the efficiency of the statistical model. On average, the proportion of gaps in all datasets used in the current project is 0.25.

The aim is to determine the error produced by them, following the same methodology as Asti *et al.* [3] and how much they influence the MG score and the final performance classification. It is expected that somehow the score will be affected.

Methods

Available materials and datasets

The initial point is to consider a multiple sequence alignment (MSA), where each row represents a different homologous sequence of the variable domain (combined VH and VL regions) of antibodies. Databases are the same as in Ref [18] and consist of 1,309 VH-VL human sequences and 373 VH-VL murine (*Mus musculus*) sequences. Only the human learning database is used for the performance classification. The test database includes 1,388 VH-VL human sequences and 1,379 VH-VL murine sequences. In all databases, according to the AHo numbering [24] the VH region corresponds to residues 1-149 and the VL region to residues 150-298. In addition, to increase the statistics, we consider also two other human datasets of only VH domain (7,720 sequences) and only VL domain (3,723 sequences) again collected in [18]. These two datasets contain the sequences used for the VHVL dataset plus the sequences that did not match between the two and were removed for the combined dataset.

	VH	**VL**
Seq 1	----------AASG-FTFRS-----YWMTWVRQASGKGLE WVANIKQD---GSDKYYVDSVKGRFTISRDNAKNSLYLQMNSL RAEDTAVYYCARSGIVLVPA----------APGLYYMDVWGQ--------	----MTQSPDSLAVSLGERATINCKSS—QSVLYSSNNKN YLAWYQHKPGQPPNLLIYW--------ASTRESGVPDRFSG SGSG--TDFTLTISSLQAEDVAVYYCQQYYS---------- ------TPYTFGQGTKLEIK-
Seq 2	----------EVKKPGASVKVSCKASG-YTFTN-----YYIYWVRQAPG QGLEWMGIINPS---GGSTSYAEEFQGRVTLTRDTSTSTVYME LSSLRSEDTALYYCARDFQAYY----------RYGYLYAWGQG TLSSVSS	------P-PSVSGSPGQSVTISCTGTS-SDVGG---- YNRVSWYQQPPGTAPKLMIYD--------VSYRPSGVPD RFSGSKSG—NTASLTISGLQTEDEADYYCSSYTS S----------SLYVFGTGTKVTVL-
Seq 3	EEQVVES-GGGFVQPGGSLRLSCAASG-FTFSP----- YWMHWVRQAPGKGLVWVSRINS---- DGSTYYADSVKGRFTISRDNARNTLYLQMNSLRAEDTAVY YCARDRYY----------GPEMWGQGTMVTVSS	DVVMTQSPLSLPVTLGQPASISCRSS--QSLVYSD- GNTYLNWFQQRPGQSPRRLIYK-------- VSNRDSGVPDRFSGSGSG-- TDFTLKISRVEAEDVGVYYCMQGTH---------- WPLTFGGGTKVEIK-
Seq 4	-----ES-GPTLVKPTQTLTLTCNLSG-FSLSTS--- GVSVGWIRQPPGKALEWLALIYW---- DDDKRYSASLKSRLAITKDTSKNQVVLRMSNMDPADTGTYF CAHSWGL----------GFDFWGQGTLVTVSS	----TQSPSSLCASVGDRVTITCRAS--QSIS------ SYLNWYQQKPGKAPKLLIYA-------- ASSLQSGVPSRFSGSGSG—TDFTLTISSLQPEDFATYY CQQS----------Y----------

Table 1. Example of sequences of the dataset

Example of four sequences of the multiple sequence alignment of the VH-VL human test dataset. The total length of the alignment is 298: VH 1-149 and VL 150-298.

For convention, the following characters will be used: Q refers to the number of different amino acids (Q=20), M to number of sequences (peptides), L to the length of the sequences (L=298 for the VH-VL combined, L=149 when one variable domain) and N to the number of binary elements in each sequence (N=LxQ). Each column (L) corresponds to the length of the sequence according to the residue numbering scheme AHo.

Julia codes with functions to generate the MG model and score sequences according to it were written by Clavero-Álvarez *et al* [18]. All the other codes used for the analysis presented in this report were written by myself, using Julia version 1.0.3 [2], when speed was an issue, and R version 3.5.2 [22], when the availability of specific libraries for bioinformatics analysis, still not present in Julia, were crucial. The codes are provided in the appendix.

Correlation between the contact map and model covariance

The first objective is to study residue correlation using the parameter covariance matrix Σ generated by the multivariate Gaussian distribution N(μ, Σ) [4,18].

Using the available code from Ref [18] and using the same values for the parameters λ (determining the contribution of the prior) and Ω (reweighting of the sequences in the learning datasets, to reduce biases: see [3,18]), we obtain the most likely estimation of the parameters μ, Σ in the Multivariate Gaussian Distribution N(μ, Σ).

In particular, for the combined VHVL case, the matrix Σ is estimated from the observed covariance matrix computed from the MSA of the learning database, having a 5960x5960 dimension (LQ x LQ = N x N).

Its inverse (Σ^{-1}), i.e. the precision or interaction matrix, represents the effective interactions that generate the observed distribution: indeed, its element ρ, τ with ρ =(i-1)*Q+ α and τ =(j-1)*Q+ β, represents the interaction between a residue of type α at position i and a residue of type β at position j, being α, β =1,...Q and i,j=1...L. In order to study how these interactions compare with the residues contact map, we need to reduce to a LxL matrix, that in some way accounts for the different species that can be found at any site. To do so, the Frobenius Norm (FN) of the inverse of the covariance matrix (Σ^{-1}) is computed as described in [8],

$$S = \left\| \Sigma_{ij}^{-1} \right\|_2 = \sqrt{\sum_{k,l=1}^{Q} \Sigma_{ij}^{-1}(k,l)^2}$$

so that a single score is obtained for each Q x Q block and the final matrix S has a dimension L x L. To simplify the comparison with the contact map, the diagonal and the lower triangular were set to zero so that the matrix contain only one value per pair of different residues.

These steps were implemented using Julia version 1.0.3 [2] (*code is provided in the Appendix*) for each MSA (human and murine).

In order to see if S contains some structural information, we need to compare it with a matrix that describes structural interactions. However, it is expected that every sequence in the database has its own (unknown, in most cases) structure, with small differences from one another, so it is pointless to choose a given structure and calculate precisely what energy would have a particular sequence adopting it, since this does not account for the structural adjustments that would affect the energy. On the contrary, it is better to use a coarse grained description of the interactions, as provided by a contact map. To this end, an average residue-residue contact or distance matrix $\overline{D}$ is calculated using 56 PDB structures that contain the VH and VL regions (Table 2), yielding n=58 VH-VL sequences in total, from the Protein Data Bank [19]. In all cases, the VH and VL sequences are assigned as different chain identifier, so they were combined to construct the VHVL contact map.

The expression for $\overline{D}$ reads:

$$\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$$

where D is the 298x298 contact map matrix obtained from the PDB file of each sequence as described in the following.

The PDB file of each antibody contains the coordinates of all its atoms, so it is easy to calculate a contact map, i.e., a matrix C whose elements C_{ij} are 1 if two atoms from i and j are closer than a given threshold d_0, and 0 otherwise. However, residues indices are challenging to deal with, since they refer to the residue position along the sequence of each chain in the PDB file, and of course, they do not contain gaps.

Hence we generated two separate VH and VL fasta files with 58 sequences and they were aligned in ANARCI [21] to create a correspondence between the old (own PDB ID numbering) and the new indexes (AHo numbering scheme). Then, they both domain were combined so that there is a unique table of residues indices for each PDB. Thereby, having a correspondence between the PDB indexes ("old") and the indexes in accordance with AHo numbering ("new"): $i_{new} = f(i_{old})$, the LxL matrix D can be initialized to the zero matrix and then filled in by setting $D_{i\_new,j\_new} = C_{i\_old,j\_old}$.

Since for each protein the contact matrix D is composed of 0 and 1, the final average contact matrix $\overline{\mathrm{D}}$ satisfies $0 \leq \overline{D}_{ij} \leq 1$.

The contact matrix $\overline{\mathrm{D}}$ depends on the threshold d_0 defining the contacts and was calculated for different contact thresholds: from 1.4 to 50 Å by 0.1 of difference (these are the cutoff distance values below which atoms, in this case alpha carbon atoms, are considered in contact). Such thresholds were selected because below 1.4 no less than 20 contacts are observed and the maximum was calculated using the software PyMol [20] to determine an approximated maximum distance between residues: at d_0=50 Å, basically all pairs of residues are in contact, so nothing changes upon further increasing the threshold.

To properly calculate the correlation between the S and $\overline{\mathrm{D}}$ matrix, another subtle technicality should be addressed: by construction, S will be a full matrix, with few (or none) null elements. On the other hand, depending on the distance threshold d_0, $\overline{\mathrm{D}}$ can be a more or less sparse matrix. In order to avoid biases on correlations just due to the increase of the non-zero elements, we proceeded as follows: for each $\overline{\mathrm{D}}$, the number of non-zero elements c (when $\overline{D}_{ij} > 0$) was counted. Then, the c largest values of the Frobenius matrix S (see Fig. 4) were maintained and the rest of the elements of the matrix converted to 0. These contact map matrices were obtained in R program [22] using the package Bio3D [23] (*code is provided in the Appendix*).

The correlation coefficients between the human and murine S and all average contact matrices $\overline{\mathrm{D}}$, was calculated in R using the Pearson and the Spearman methods (*code is provided in the Appendix*). Only the upper triangular of all matrices are considered in order to have 1 unique value per pair of residues. The results are reported in Table 4.

PDB ID structure			
1A14	1I3G	1OAR	4CKD
1A6U	1FO0	1N4X	2YSS
1A7N	1FVC	1NMC	3DUR
1AP2	1I8I	1QNZ	4LLV
1BVF	1IC4	1UA6	4LRN
1BVK	1J1O	1VFA	4M8Q
1BVL	1J05	1WZ1	4M62
1C08	1JHL	2A0L	4OB5
1DL7	1JV5	2DQE	4QXT
1DLF	1KB5	2DQF	5AYU
1DQL	1KIP	2EKS	43C9
1DSF	1MFA	2GSG	3DUS
1EZV	1MQK	2OTU	1DVF
1FGV	1MVU	2UZI	1QFW

Table 2. List of PDB structures analyzed

The 56 PDB IDs from the Protein Data Bank (https://www.rcsb.org/) used to generate an average contact matrix. The search was done by the filter Fv, which corresponds to the variable fragment of the antibody. All of them contain 1 VH sequence and 1 VL sequence, except 1DVF and 1QFW entries, which include 2 sequences of each variable domain, yielding in total 58 VH-VL sequences.

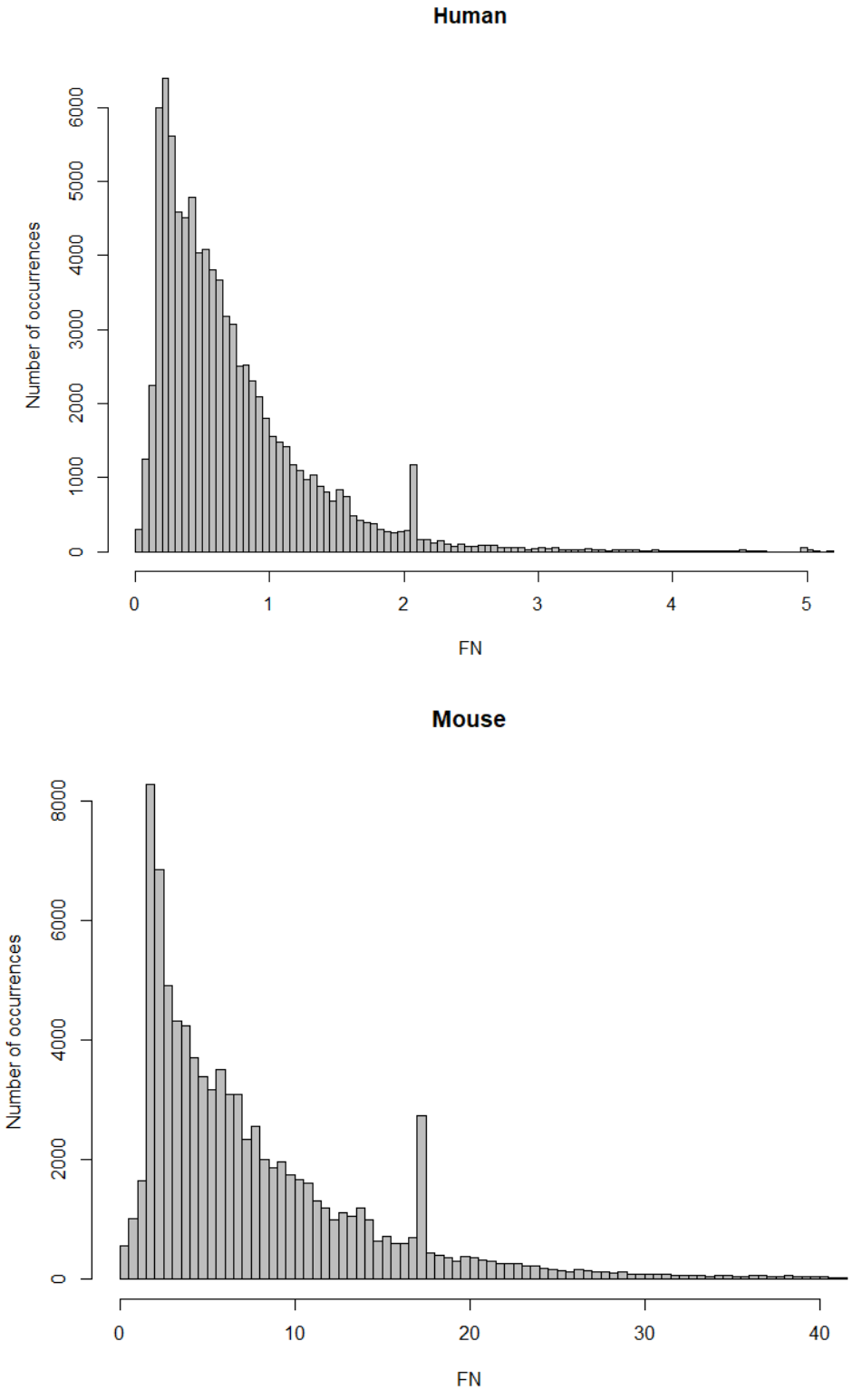

Figure 4.Distribution of the values of the FN of Σ^{-1} in the human and murine sequences.

Histogram with the distribution of the values of the inverse of the Frobenius Norm (FN) of the covariance matrix. It is calculated based on the parameter of the Multivariate Gaussian (MG) model covariance matrix Σ using the learning human and mouse database respectively, then performing the Frobenius Norm (FN) of its inverse (Σ^{-1}). Only the upper triangular matrix is left to avoid duplicated values. Maximum FN values are 53.16 in human sequences and 487.03 in murine sequences, but histogram has been cut because the large majority of the values are much lower. Peaks at 2 in human and at 17 in mouse corresponds to FN values between two gap positions.

Identification of key positions

To analyze the importance of the columns of the MSA in the classification of human and murine sequences, we formulate the following questions: what are the columns whose removal most affect the classification effectiveness? Can such columns be identified independently, just based on some statistical properties? To translate these questions into a quantitative criterion, we use the Area Under the Curve of the Receiver Operating Characteristic curve as a measure of the goodness of a classification. Following Ref [18], we define as "positive" instances the human sequences in the test database, and "negative" instances the mouse sequences. So, a "true positive" prediction will correspond to a human sequence correctly predicted as such, while a "false negative" will represent a human sequence predicted as murine by the method, and so on. From these quantities, a ROC curve in the (TPR, FPR) plane can be drawn (see Fig. 5). The bigger the area under the curve, and close to 1, the better the performance of the classifier is. The column properties that we want to relate with the AUC are the site entropy, the Kullback-Leibler divergence between murine and human empirical distributions and "energy bias" at each position, as defined below. For each indicator, we sorted the columns from the least to the most important and deleted them following this order, progressively reducing the MSA length. Each time a new column was removed from the learning and test datasets; the posterior parameters of the model μ and Σ were generated again, without changing the values of the parameters λ and Ω.

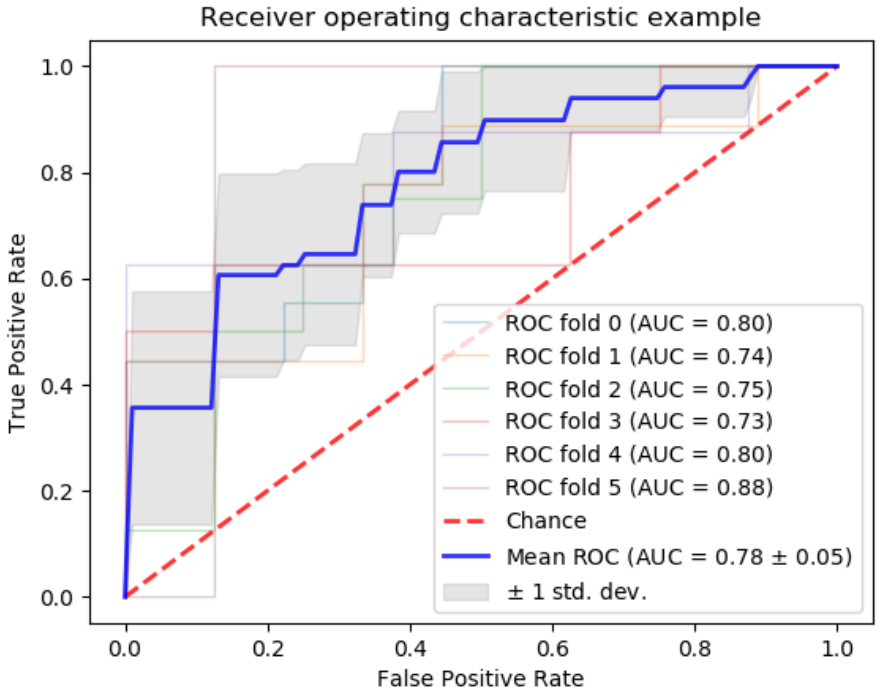

Figure 5. Example of a ROC curve

Figure retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html. It is a two-dimensional graph in which the false positive rate (FPR, proportion of murine sequences incorrectly considered as human) is plotted on the X axis and the true positive rate (TPR, proportion of human sequences correctly classified as human) is plotted on the Y axis. Each point represents a TPR/FPR corresponding to a particular threshold. The higher the Area Under the Curve (AUC), better the model is at predicting; AUC=1 represents the perfect test. The dashed red line in the diagonal represents the ROC curve of a random predictor (AUC=0.5).

Once the posterior parameters of the model are calculated, the MG-score of each sequence of the test datasets are determined, yielding two vectors with the scores values of the human dataset (M=1,388) and the murine dataset (M=1,379). This score is defined as the logarithm of the probability density [18]. According to whether such scores are below or above the threshold score found in Ref.[18], the sequences are classified as murine or human, and we can assess whether such predictions are True Positive, False Positives, True Negatives or False Negatives.

To analyze the performance classification, a receiver operating characteristic (ROC) curve is created upon varying the threshold score and the Area Under the Curve (AUC) is used to compare the different ROC curves. All calculations were made in Julia version 1.0.3 (*code is provided in the Appendix*). Thus, in the end, we have a curve in the plane (number of deleted columns, AUC), indicating how much eliminating columns according to the proposed observable reduces the classification power of the method.

Naive ("teleologic") approach

Our first approach was simply to investigate which column of the 298 caused, upon removal, the smallest decrease in the AUC (or possibly, also an increase), remove it, then finding which of the 297 left causes the next smallest decrease upon removal, and so on. This procedure would yield a smooth curve that, possibly after a small increase, would decay without noise or bumps, by construction. Unfortunately, this approach resulted to be very expensive computationally, since it involves calculating hundreds of ROCS before removing any column, and each ROC involves many calculations with different threshold scores, between human and murine.

So we used a different approach: we calculate at the beginning which column causes, upon removal, the smallest decrease in AUC, which one causes the second smallest, and so on, and then we remove them in this order, without calculating the AUC again.

Site entropy

The entropy of each column was defined by the information entropy (S) function:

$$S_i^\alpha = -\sum_i p_i^\alpha \log p_i^\alpha$$

where p_i^α is the frequency of each amino acid character i that appears in columns α of the learning MSA.

Kullback-Leibler divergence

The relative entropy or Kullback–Leibler divergence (D_{KL}) of each column was calculated as:

$$D_{KL}^\alpha(p|q) = \sum_i^N p_i^\alpha \log \frac{p_i^\alpha}{q_i^\alpha}$$

where p_i^α is the frequency of each amino acid character i in columns α of the human MSA and q_i^α is the frequency of each residue i in columns α of the murine MSA.

Energy bias

Notice that the D_{KL} divergence is a quantity that depends on both the human and the murine learning datasets, while the entropy depends only on the human one. Another quantity just dependent on the human dataset is the energy bias, defined as follows. Consider that the interaction can be written as:

$$-(x-\mu)(\Sigma^{-1})^T(x-\mu) = -x\,(\Sigma^{-1})^T x + h^T\,x + constant$$

where h_j

$$h_j = 2\left(\sum_i \mu^i\,(\Sigma^{-1})_{ij}\right)$$

h_j acts as an external field, being μ and Σ^{-1} the mean and the inverse of the covariance matrix of the MG model. The interaction term cannot be written as an external field influencing the symbol x_j appearing at position j because of the quadratic term in x. However, we can give an estimate of how relevant is the interaction for biasing a position, upon defining the average:

$$\overline{v_j} = \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N} x^{i(m)}(\Sigma^{-1})_{ij}$$

where v_j is the weight of interactions between residues and x are the human sequences from the dataset. Introducing $\varphi_j = \overline{v_j} - h_j$,we have that:

$$\varphi_i = \sum_{\alpha=1}^{Q} \Phi^{(i-1)Q+\alpha}$$

i=1...L, is an indicator of the amount of bias a residue feels at position i.

Following the same steps explained before, columns were removed from the smallest to the largest φ_j.

Study of the importance of the dataset

The learning database used to study the classification consists of 1,309 human sequences. It was reduced progressively by 20 sequences until only 9 sequences left (66 databases of different sizes in total), using always the same test databases. The choice of sequences selected each time to be deleted was done randomly by the program.

The parameters of the model μ and Σ were recalculated, then the MG score of the human and murine test sequences was estimated and the performance classification was measured by AUC. The whole process was done with Julia program too (*code is provided in the Appendix*).

Those steps were performed 20 times for each size of the database (overall, the average AUC and the standard deviation were estimated) for the complete sequences (VH-VL), just VH (residues 1-149 of all datasets selected) and just VL (residues 150-298) (Figure 11). In the case of just VH or VL regions, the optimal λ and Ω are different, thus such values were changed to the appropriate value, which were already calculated by Clavero-Álvarez *et al* [18].

The same methodology was repeated for the other two human learning datasets of VH and VL (Figures 12 and 13). Because of the larger size of those databases, they were reduced each time by 100 and 50 sequences respectively. The parameters λ and Ω were optimized before (*codes are provided in the Appendix),* obtaining λ=0.4 and Ω=0.4286 for the VH dataset and λ=0.1 and Ω=0.225 for the VL dataset.

Ω is defined as the value that maximizes the Frobenius norm of C_{ij} [18]. From a range of values, being 0 < Ω < 1, the weights of the sequences of the dataset are calculated as $w_m^\Omega = {}^1/_{n_m^\Omega}$ where

$$n_m^\Omega = \sum_{l=1}^{M} \theta(\vartheta_{lm} - L\Omega)$$

The empirical covariance for each Ω:

$$C_{ij}^\Omega = \frac{1}{M} \sum_{m=1}^{M} (x_i^m - \bar{x_i})\,(x_j^m - \bar{x_j})\, w_m^\Omega$$

The score is defined as $1 - \left|C_{ij}^\Omega\right|$, being $\left|C_{ij}^\Omega\right|$ the Frobenius norm of the empirical covariance. The optimal value of Ω is the one that maximizes the score.

For λ, the optimal value was calculated by analyzing different ROC curves. Again, from a range values, being 0 < λ < 1, the posterior parameters of the model μ^λ and Σ^λ are generated with the optimal Ω previously calculated. Then, the MG-scores of each sequence of the test datasets are determined, yielding two vectors with the scores values of the human dataset and the murine dataset. To analyze the performance classification, a ROC curve is created upon varying the threshold score and the AUC is used to compare the ROC curves of the different λ values. We chose the optimal value as the one that with the highest AUC.

MG-score correction

In Ref.[4], a correction to the MG-score was proposed in order to reduce the influence of the gaps in the alignment, whose presence may bias the interaction matrix. This correction consists in subtracting, from the plain score, the one obtained upon learning from a different Multiple Sequence Alignment, where the gaps are maintained at their place and all other residues are randomized. Following here the same recipe, all amino acids positions were changed randomly, but the gap positions were unaltered *(code provided in appendix*) according to their original frequency in the MSA (Table 3).

Amino acid	Frequency	Amino acid	Frequency
Alanine (A)	0.063	Methionine (M)	0.013
Cysteine (C)	0.019	Asparagine (N)	0.028
Aspartic acid (D)	0.043	Proline (P)	0.044
Glutamic acid (E)	0.031	Glutamine (Q)	0.058
Phenilalanine (F)	0.031	Arginine (R)	0.045
Glycine (G)	0.105	Serine (S)	0.136
Histidine (H)	0.007	Threonine (T)	0.082
Isoleucine (I)	0.038	Valine (V)	0.069
Lysine (K)	0.037	Tryptophan (W)	0.024
Leucine (L)	0.071	Tyrosine (Y)	0.057

Table 3. Frecuencies of each amino acid in the VHVL human sequences.

Frequency of each amino acid in the VHVL human sequences, gaps were not included. In order to see the error produced by gaps, the residue positions were changed randomly with weights according to this proportion table.

Using this modified dataset, the posterior parameters μ_{rand} and Σ_{rand} from the gap-corrected dataset are calculated yielding a new MG-score (that is a log probability score, see Ref [18].

These steps were repeated 30 times to obtain an average MG random score. Lastly, the performance classification was studied with the MG-score corrected:

$$MG\ score\ corrected = MG\ score - \overline{MG\ random\ score}$$

Following previous sections, the performance classification of the MG-score and the MG-score corrected was also compared by considering the AUC.

All steps were performed for each of the three learning datasets (VH-VL, VH and VL) and for the VHVL sequences, they also were applied to the two separate domains. Parameters λ and Ω were not modified; the values previously optimized were used according to each dataset.

Results and Discussion

Correlation between the contact map and model covariance

Table 4 reports the Pearson (r) and Spearman (s) correlations coefficients between the average contact map matrix for different distance thresholds and the Frobenius norm (FN) covariance matrix of the statistical model and its inverse.

In the case of the inverse of the covariance matrix Σ^{-1} the best correlation coefficients are obtained when the minimum cutoff distance used to build the contact map (Å=1.4), in which the number of non-null elements in the contact matrix $\overline{D}$ is 260. Yet, the correlation coefficients are low (for human sequences r=0.298 and s=0.297 and for mouse r=0.38 and s=0.324), which indicates the correlation is weak. Taking into account that the total number of residues is 298, the contacts mainly correspond only to first neighbors. These results suggest that Σ^{-1} cannot predict the residue-residue correlations and it is only somehow coincident because of the first neighbor contacts.

Even if our interest was on the correlation between the interaction Σ^{-1} and the contact map, we also test the association between the latter and Σ.

In the case of the covariance matrix Σ, the highest correlation coefficients correspond to the maximum cutoff distance (Å=50). The number of non-null elements in $\overline{D}$, that is, the number of contacts, is quite big, exceeding 28500 contacts. In this situation, the coefficients indicate a stronger correlation (for human sequences r=0.671 and s=0.782 and for mouse r=0.551 and s=0.757), although we suspect that it might be more related to the increase in the number of no-null matrix elements in the comparison, than to the actual similarity of the distributions.

Distance threshold(Å)	**1.4**	**10**	**20**	**30**	**40**	**50**
Number of contacts	**260**	**12433**	**23135**	**27792**	**28514**	**28516**
	Pearson correlation coefficients (r)					
FN Σ^{-1} human	0.298	0.17	0.071	0.039	0.026	0.024
FN Σ^{-1} murine	0.38	0.212	0.146	0.137	0.14	0.141
FN Σ human	0.03	0.385	0.574	0.667	0.674	0.671
FN Σ murine	0.03	0.358	0.489	0.543	0.553	0.551
	Spearman correlation coefficients (s)					
FN Σ^{-1} human	0.297	0.043	0.09	0.09	0.084	0.085
FN Σ^{-1} murine	0.324	0.12	0.223	0.256	0.262	0.266
FN Σ human	0.026	0.41	0.634	0.77	0.781	0.782
FN Σ murine	0.023	0.343	0.566	0.729	0.754	0.757

Table 4. Correlation coefficients between average contact matrix and the covariance matrix

Table with the Pearson and Spearman correlation coefficients obtained as a measure of the correlation between the Frobenius norm (FN) covariance matrix of the statistical model and the average contact map for different thresholds created with PDB structures. All correlations are statistically significant, with p-values less than 0.5. Only the upper triangular matrices were used to calculate the correlation. Cutoff distances values to consider alpha carbon atoms in contact were calculated from 1.4 to 50 Å by 0.1, but only 6 thresholds are shown. This cutoff distance value below which atoms are considered in contact. The number of contacts corresponds to the number of elements different from 0 in each average contact map matrix D.

Our results are in line with those in Asti *et at.* [3], where the authors were not able to predict any structural information with the MG model again in the case of antibodies. Baldassi *et al*. [4] used also the Frobenius Norm in the MG model to predict residue-residue contacts and they obtained very good results, but the average number of sequences used in their alignment (more than 30,000) could affect the difference in results compared to the current <M>= 841. Another significant difference is that they study the contacts in a variety of protein families. Because of the nature of the variable region of antibodies, the high variability in the CDR, the residue-residue contacts could be more difficult to predict based on sequence information only.

Other explanation of not being able to predict contacts is the relatively high presence of gaps in the MSA (average proportion of gaps being 0.25), already been discussed how they can affect in the contact map prediction [29]. Authors explain in their work that when a gap correction term is applied in the model or when they do not include the inferred couplings involving gaps in the final scoring of the coupling matrix Σ^{-1}, the accuracy of contact prediction significantly increases.

Relation between site properties and classification power

We move now to the study of the importance of the columns in the classification task, using the area under the ROC curve as a measure of the goodness of the prediction.

As explained in Methods, our strategy is to remove the columns according to an order dictated by the value of some observable and see how this affects the AUC.

Naive ("teleologic") approach

As explained in methods, in this approach the columns are ranked according to increasing values of the quantity $\delta = AUC_{before\_column\_removal} - AUC_{after\_column\_removal}$ (that may also be negative). Then, they were removed according to such ranking. Fig 9 reports the results.

There are several comments that we might do: first, we observe that the fact that we do not recalculate the rank after each removal (time consuming, as explained in the methods) does not affect the results too much. Indeed, we see a rather smooth curve, with very few and small irregularity. If we had adopted the "correct" removal method, by definition the curve would have at most one maximum.

Second, the prediction power remains more or less constant compared to the AUC value obtained when all columns are used in the MG model (AUC=0.966) until it reached a maximum value of 0.982 when only 66 columns are left. Only a few columns, which most of them are either conserved residues or located in the VHVL surface, as will be explained later, are enough to obtain a good classification.

Third, a large number of the columns that even slightly improve the performance when removed correspond to the CDRs regions, most in VH: 17 of the top 20 less relevant columns are located in the CDRs. This indicates that CDRs regions negatively affect the correct classification of the two classes.

Fourth, the number of important columns is approximately only 5, when AUC is already larger than 0.95. They correspond to the positions 45 (S_{human}=0.04, S_{mouse}=0.84), 51 (S_{human}=0.43, S_{mouse}=1.18), 167 (S_{human}=1.2, S_{mouse}=1.73), 250 (S_{human}=1.12, S_{mouse}=1.53) and 256 (S_{human}=1.55, S_{mouse}=1.54).

It seems quite impressive that only a few columns are required to distinguish between the two classes. However, a closer inspection reveals that the residues of such positions are highly different in each group. The clearest example is column 250: 82% of the human sequences in the databases consist of the amino acids E or F, while in the murine sequences those amino acids are rarely present and the most frequents are A and L (64%).

To understand the structural role of the important positions, we resort to the correspondence between AHo numbering and Kabat numbering [27], that gives some structural information. In this way we notice that column 51 in the MSA corresponds to the VH residue G44 according to Kabat numbering, which is located in the surface of VH that interacts with VL [26]. It is one of the most critical residues in the dimer interface [24]. In the antibody humanization process, the dimer interface residues must admit that any VL domain can combine with any VH domain to establish a functional and stable structure.

Column 45 refers to the VH residue R38 and it is also located near the VH-VL interface. Both are positioned in the framework region between CDR H1 and CDR H2. Critical residues in the interface of the two variable domains are frequently highly conserved and it is a common grafting strategy to maintain such residues because they affect the orientation of the region [28].

Column 256 (107 in VL region in AHo numbering) refers to the VL residue 89, it is located in antigen interface [24] and is an important residue in the grafting process because it interacts with a crucial residue in CDR L1 [28]. It is a structurally conserved amino acid according to [30].

Column 167 (position 18 in VL according to AHo numbering) corresponds to the VL residue R18. It is situated in the first FR region. Column 250 (101 in VL in AHo numbering) refers to the VL residue 83 is located in FR3b. In the case of these two columns, no structural or functional relevance was found.

After finding in this section a list of columns, whose ordered removal increasingly affects the classification capabilities of the method, we try to relate the classification relevance of these columns with other independent observables to understand if there is a way to predict when and why a column is relevant for the classification task.

In particular, we consider the site entropy, the relative entropy (Kullback-Leibler divergence) and the "energy bias" defined in Methods.

Site entropy

Following [4], we start by considering the values of the information entropy ($S_i = -\sum_i^Q p_i \log p_i$), being i the specie, of each column that are reported in figure 7 with the top ranking positions in Table 5.

We observe that the majority of the most entropic columns are located in the CDRs regions especially in CDR3 (in VH located in positions 107-138 and in VL positions 257-287) following by CDR2 (58-68 and 207-217) (Figure 7) as expected because they are the hypervariable regions, where the antigens bind to. Of these regions, the one that carry the most entropy is the third CDR of the VH domain, which matches with previous studies saying it is the most diverse of the six regions [25].

Information entropy Human					
Ranking	**Residue Position**	**S**	**Ranking**	**Residue Position**	**S**
1	*112*	*2.85*	*11*	*135*	*2.38*
2	*111*	*2.78*	*12*	*109*	*2.37*
3	*110*	*2.74*	*13*	*60*	*2.30*
4	*113*	*2.70*	*14*	*67*	*2.29*
5	57	2.58	*15*	*61*	*2.29*
6	*284*	*2.47*	*16*	*132*	*2.24*
7	*133*	*2.47*	*17*	*207*	*2.17*
8	*114*	*2.46*	*18*	*285*	*2.14*
9	*134*	*2.44*	*19*	*59*	*2.10*
10	*286*	*2.40*	*20*	*40*	*2.10*
Information entropy Mouse					
Ranking	**Residue Position**	**S**	**Ranking**	**Residue Position**	**S**
1	*109*	*2.65*	*11*	*181*	*2.21*
2	57	2.53	*12*	*259*	*2.14*
3	*110*	*2.51*	*13*	*61*	*2.14*
4	*111*	*2.48*	*14*	*40*	*2.11*
5	*207*	*2.42*	*15*	*258*	*2.11*
6	*284*	*2.37*	16	220	2.10
7	*67*	*2.29*	*17*	*59*	*2.03*
8	*112*	*2.25*	*18*	*134*	*2.02*
9	*135*	*2.24*	19	191	2.00
10	69	2.21	*20*	*286*	*1.96*

Table 5. Top 20 positions with the greatest information entropy in the human and murine database. Entropy defined as ($S = -\sum_i p_i \log p_i$). The maximum possible entropy is Smax=3.05, as there are 21 possible characters (the 20 amino acids and gap). In italics, position located in CDRs. The large majority of the top positions correspond to CDR locations.

Kullback-Leibler divergence

The observation that, following the naive approach, the most relevant columns are those with a definite difference in the residues frequency between human and murine database suggests to look at the Kullback-Leibler (D_{KL}) divergence as another relevant column property (Table 6), even if, at difference from the entropy, D_{KL} depends not only from the human distribution, but also from the murine one.

The values of the Kullback-Leibler divergence $D_{KL} = \sum_i p_i \log \frac{p_i}{q_i}$, being p_i the human distribution and q_i the murine distribution of the residues, are indicated in figure 6 and the top ranking positions in Table 6. Contrary to the site entropy results, columns with the highest D_{KL} values are mostly situated in FR regions. This suggests that such columns might play an important role in the model to differentiate both classes of mAbs. The figure also shows a few residues that stand out, which differs with the site entropy results too, where entropy distribution is more uniform.

Kullback–Leibler divergence (D_{KL})					
Ranking	**Residue Position**	**D_{KL}**	**Ranking**	**Residue Position**	**D_{KL}**
1	200	3.98	11	296	1.37
2	250	2.83	12	294	1.30
3	220	2.56	*13*	*287*	*1.16*
4	12	1.71	*14*	*261*	*1.16*
5	244	1.61	*15*	*259*	*1.14*
6	238	1.55	16	245	1.14
7	47	1.55	17	173	1
8	78	1.45	18	98	0.84
9	295	1.44	19	71	0.83
10	297	1.42	20	20	0.74

Table 6. Top 20 positions with the largest Kullback-Leibler divergence.

D_{KL} defined as $\sum_i p_i \log \frac{p_i}{q_i}$. The human distribution p_i was used as a reference. In italics, position located in CDRs. The large majority of the top positions correspond to FR locations.

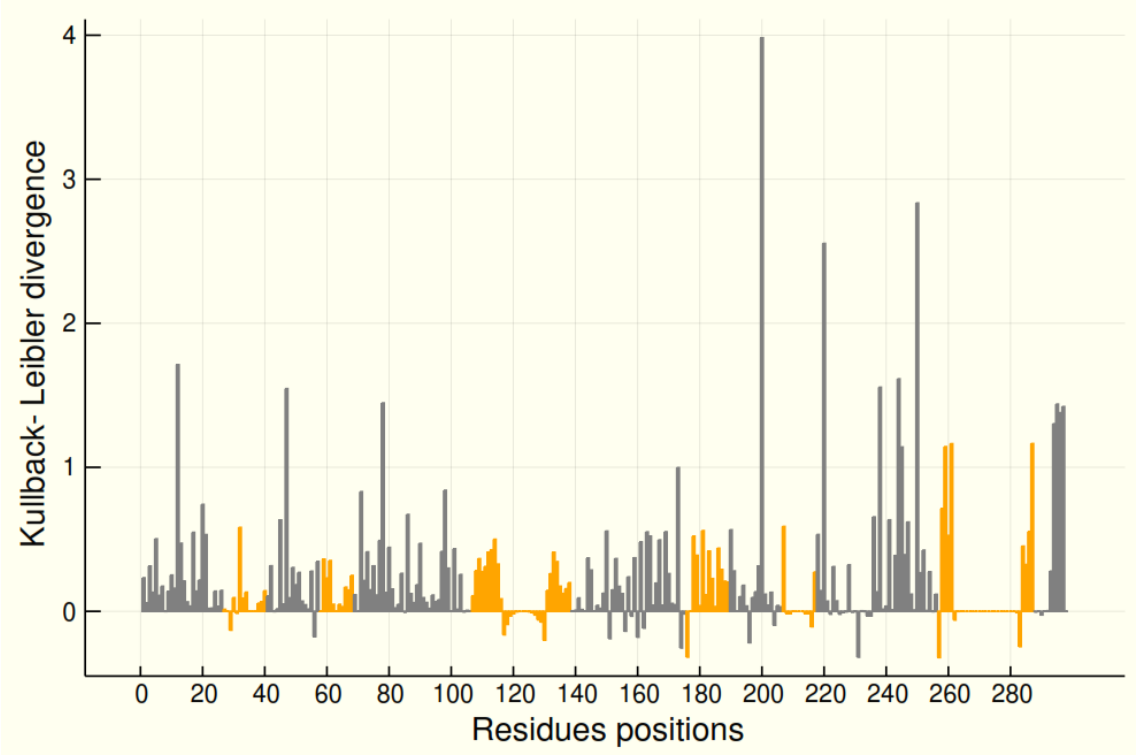

Figure 6. Kullback-Leibler divergence of each column

D_{KL} defined as $\sum_i p_i \log \frac{p_i}{q_i}$. The human distribution p_i was used as a reference. The columns with the highest divergence are 200, 250, 220, 12, 244 and 238. CDR regions indicated in orange. The total length of the MSA is 298.

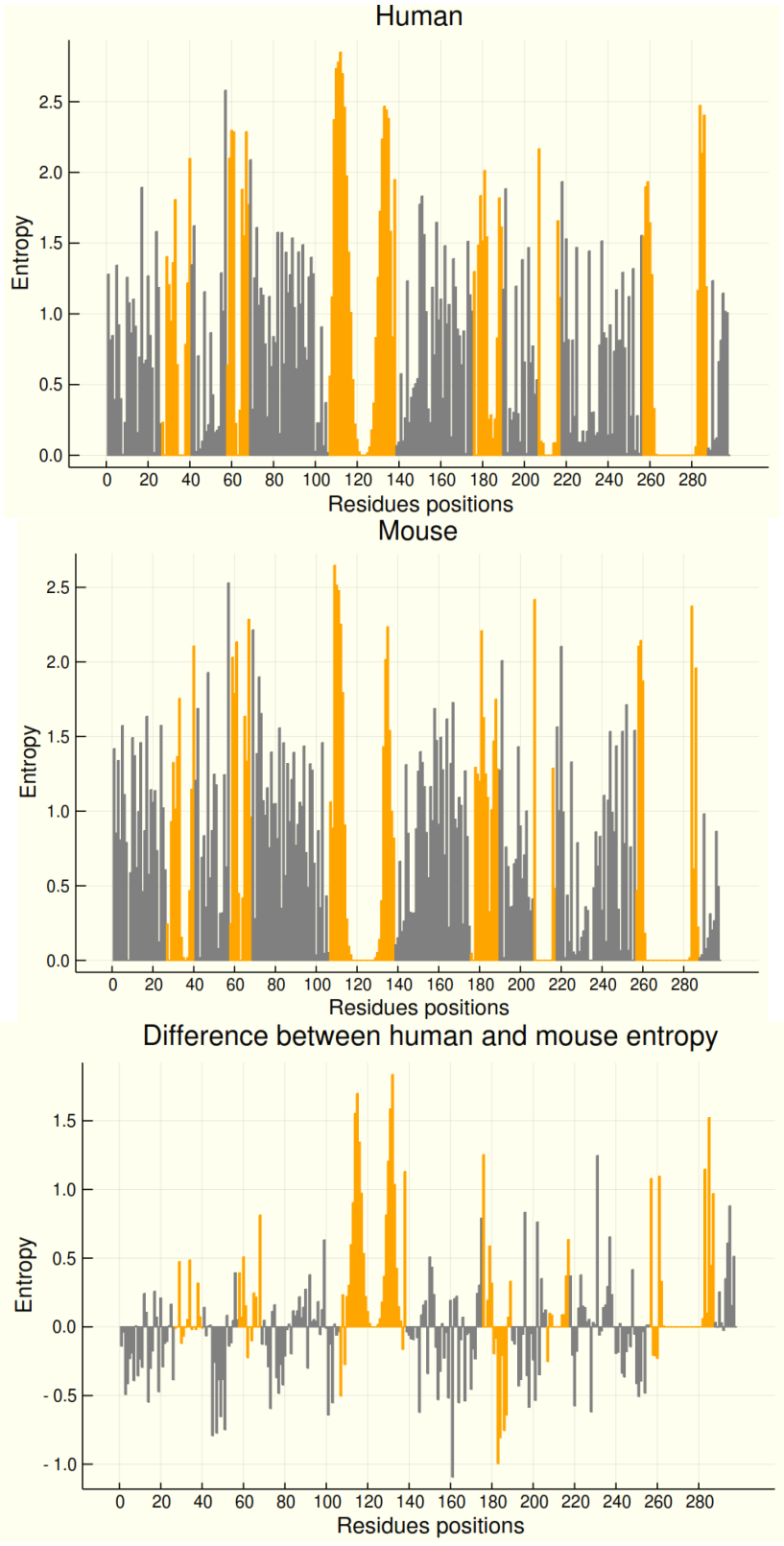

Figure 7. Distribution of the values of the information entropy of each column

Entropy ($S= -\Sigma\ p \log p$) of each column of the human and murine MSA and the difference ($S_{human} - S_{mouse}$). Because of the alphabet of amino acids (Q=20) and gaps appearing as the 21st possible character, the maximum possible entropy is S_{max}=3.05. CDR regions are indicated in orange. The total length is 298. Positions with zero entropy are columns composed of only gaps in the MSA.

Energy bias

As explained in Methods, the quantity φ_j gives an information of the effective "field" acting on a certain residue (i.e., column of the MSA), even if its intensity will indeed depend on the detailed amino-acid species that is placed at that position. A φ_j that is small in norm means that the position is not very biased towards any species; a big positive φ_j implies that the interaction term on average dominates over specific site-dependent preferences towards a residue, acting as an external field, while a big negative φ_j implies that the latter term (the "external field") dominates over the interaction one.

In table 7 and Figure 8, the distribution of $|\varphi_j|$ are shown. According to this measure of the importance of each column, the more relevant residues are located in FR2 and FR3 of both VH and VL domains.

Top $\|\varphi_j\|$ columns					
Ranking	**Residue Position**	$\|\varphi_j\|$	**Ranking**	**Residue Position**	$\|\varphi_j\|$
1	201	6170.59	11	226	5694.80
2	255	6141.94	12	240	5669.80
3	249	6115.76	13	233	5659.35
4	100	6021.36	14	242	5640.61
5	43	5968.98	15	222	5587.93
6	192	5879.67	16	83	5567.50
7	253	5861.92	17	104	5554.56
8	45	5839.75	18	229	5537.58
9	106	5735.89	19	77	5522.72
10	198	5713.67	20	26	5489.84

Table 7. Top 20 position with the highest $|\varphi_j|$, defined as the difference between h_j (external field) and v_j (interactions between residues). φ_j acts as an indicator of the amount of bias a residue feels at column j. All top 20 residues are located in FR regions, most in FR2 and FR3 of both variable domains.

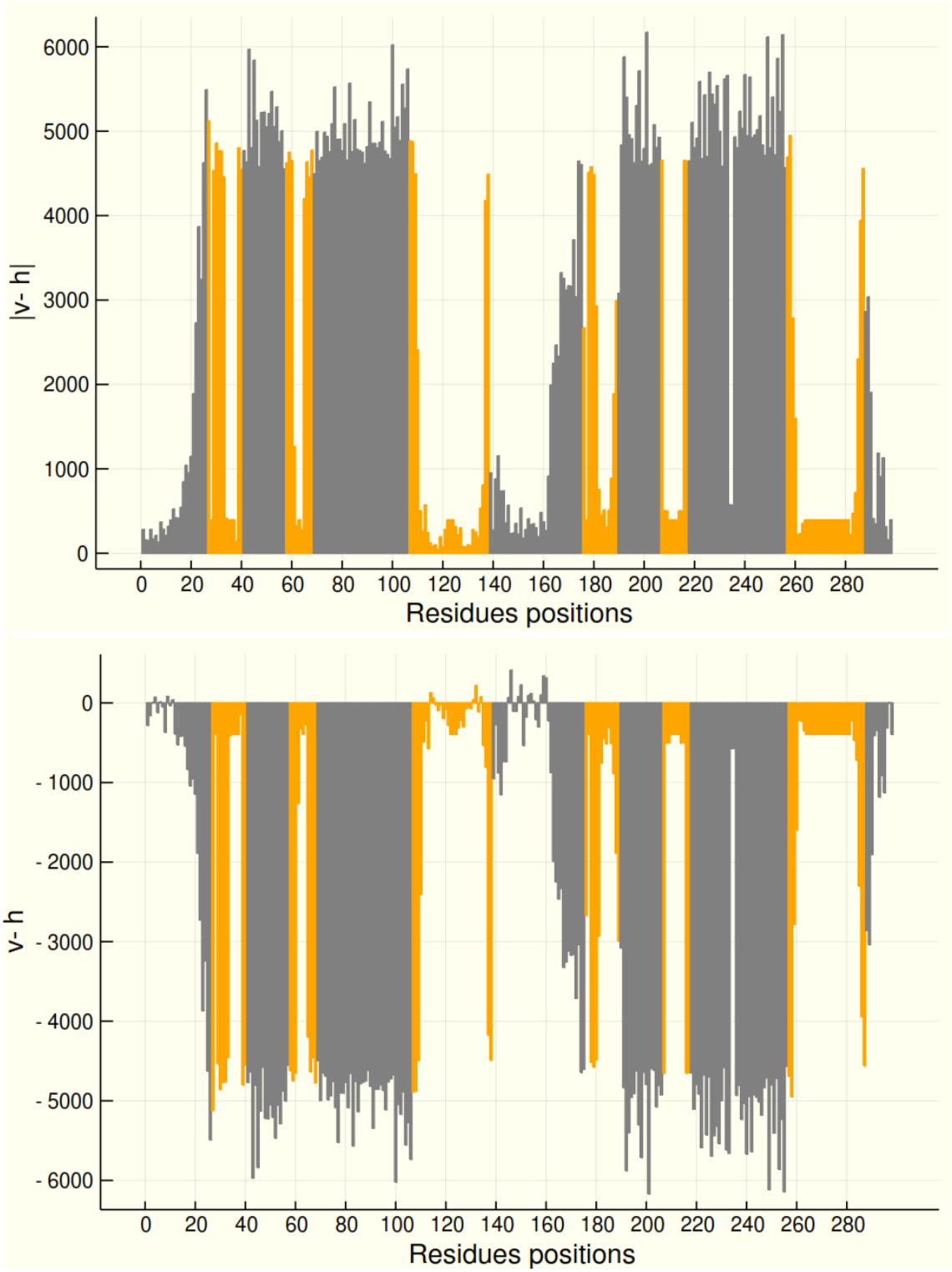

Figure 8. Distribution of the values of $|\varphi_j|$ and φ_j of each column

φ_j is defined as the difference between h_j (external field) and v_j (interactions between residues). φ_j acts as an indicator of the amount of bias a residue feels at column j. CDR residues are coloured in orange. Positions with lowest values, located particularly in CDR correspond to columns in the MSA with only gaps. The bottom figure shows how the external field h prevails over the interaction term v except some residues of FR1 and CDR3 of VH and FR1 of VL.

Comparison of all indicators

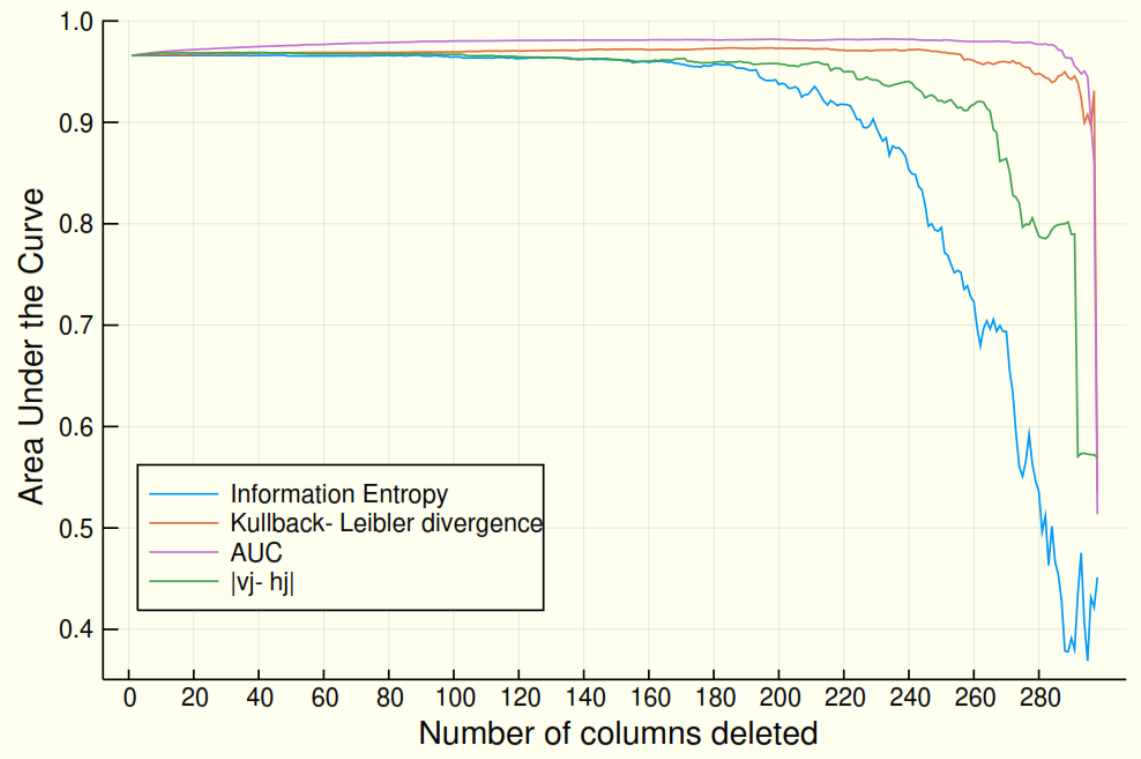

Figure 9. AUC as a function of the MSA length for the different approaches

The total length is 298. AUC was calculated each time a new column was removed according to different indicators. In blue, Area Under the Curve when the columns are removed from the lowest to the highest site information entropy. In orange, the order of the columns depends on the Kullback-Leibler divergence. In purple, columns were removed according to the obtained AUC when they were deleted. In green, the order was followed according to φ_j,which mean the difference between h_j (external field) and v_j (the norm of interactions between positions). The best indicator is the Kullback-Leibler divergence, only 2 columns are enough for AUC larger than 0.9.

Fig. 9 reports the results of the drop in AUC upon removing the columns, according to increasing values of entropy, KL divergence, or energy bias $|\varphi_j|$. We see that removing the columns according to the entropy, KL divergence or energy bias reduces the AUC faster, and none of them is optimal, in reproducing the "naive" approach.

Following the information entropy order, AUC values are considerably constant (AUC > 0.9) until only approximately 90 columns are left, then the performance classification starts to fall significantly. When L ≈ 30, the model cannot longer distinguish between human and murine sequences. However, following the D_{KL} results order, only a few columns are necessary to get a good AUC value. Specifically, with 2 columns AUC= 0.93 and AUC=0.95 when L ≈ 10. These results suggest that the most important columns are the ones with the highest Kullback-Leibler distance and not the ones with the greatest entropy.

Notice that two columns with the highest D_{KL} that are able to distinguish between the two species are 200 and 250. The column 250 was already explained before, as it is also one of the most important columns detected in the naive approach. The position 200, which corresponds to the residue 51 in the VL domain according to AHo numbering and residue 43 in the Kabat numbering scheme, is located in FR2. It has been found that A43 is highly conserved and is directly in contact with the VH domain [25].

It is clear that KL divergence is much more effective than Entropy in identifying which residues are relevant to account for in a classification task. However, it must be noticed that it has an important weakness: it depends on the murine and human distribution, and it is not an intrinsic characteristic of the latter. This means that it could not be calculated in a classification task where we simply should distinguish between human and not human, without knowing the alternative species (and its residues distribution). On the other hand the energy bias φ_j could be a reasonable alternative that just depends on the human distribution, as the entropy, but with a better performance: by following this approach, the MSA length has to be much larger to obtain a high AUC (AUC is close to 0.8 if L≈20 and AUC>0.9 if L≈30).

From the results of the naive approach, we calculate the difference in the Area Under the Curve when a column was removed with respect to the previous AUC value: $dAUC_i = AUC_{i-1} - AUC_i$. The Pearson correlation coefficient between the values of dAUC of each column and Kullback-Leibler divergence of each column is 0.34, p value=1.45×10^{-9}.

Figure 10 suggests why the correlation is not very high, even between the KL ordering and the naive approach: even if the 5 most relevant residues in the latter have a substantial KL divergence, there are several columns with high values of KL that do not correspond to crucial residues.

This can be further seen in Table 8, that reports the top positions (most relevant columns, the last to be removed in the corresponding graphs in fig 9), for the KL and Naive approach. There are some top positions in common and indeed 5 of the first 7 high ranking KL residues are found in the first high ranking positions according to the Naive approach. However, there is no clear relation overall between the two ranks, and the global Spearman correlation coefficient is approximately zero, since low ranking positions are poorly related in the two lists.

Naive ("teleologic") approach				Kullback-Leibler divergence			
Ranking	**Residue position**	**Ranking**	**Residue position**	**Ranking**	**Residue position**	**Ranking**	**Residue position**
1	250	21	73	1	200	21	258
2	256	22	190	2	250	22	86
3	167	23	186	3	220	23	236
4	45	24	228	4	12	24	241
5	51	25	157	5	244	25	45
6	191	26	71	6	238	26	247
7	49	27	204	7	47	27	207
8	290	28	145	8	78	28	32
9	184	29	173	9	295	29	190
10	220	30	194	10	297	30	181
11	77	31	103	11	296	31	150
12	238	32	187	12	294	32	286
13	284	33	78	13	287	33	169
14	200	34	21	14	261	34	163
15	207	35	98	15	259	35	17
16	47	36	166	16	245	36	218
17	156	37	86	17	173	37	21
18	258	38	245	18	98	38	260
19	101	39	164	19	71	39	178
20	170	40	13	20	20	40	164

Table 8. Comparison between the 40 most important columns according to the method used

In color, columns that appear in top positions in both methods. The top 7 columns of KL divergence are present in the top 16 columns of the Naïve ("teleologic") approach.

Compared to previous researches, the outcome of the reduction of the MSA based on the entropy of the columns is similar to Asti *et al* [3]. The performance of the MG model remained the same until only the 60 more variable columns are left when they used a hypermutated cluster antibodies test, but they obtain approximately 10 important columns with a germline cluster.

However, they compare the performance of the MG score with the neutralization power of the Abs, so it is related to the most entropic columns (CDR). In the present project, the performance is compared to the ability to differentiate human and murine classes, in which the importance of columns might depend not in the variability or entropy, which is higher in CDR regions, but on the columns located in FR regions.

To sum up, the best method to select the positions that carry the greatest information is the KL divergence, since a few high-valued columns are enough to produce good classifications. However, it depends on two distributions and cannot be applied for generic classification human/non-human.

Moreover, even using the KL divergence, it is difficult to predict exactly, before performing the test, which columns are important. This is frustrating, because it implies that we cannot simplify the method using shorter MSA that disregard less relevant columns. However, this could also be taken as an indication that interactions between residues (and co-mutations) are indeed relevant also in classification, and no single-column quantity, as the ones that we have analyzed here, is enough to account for the changes in AUC.

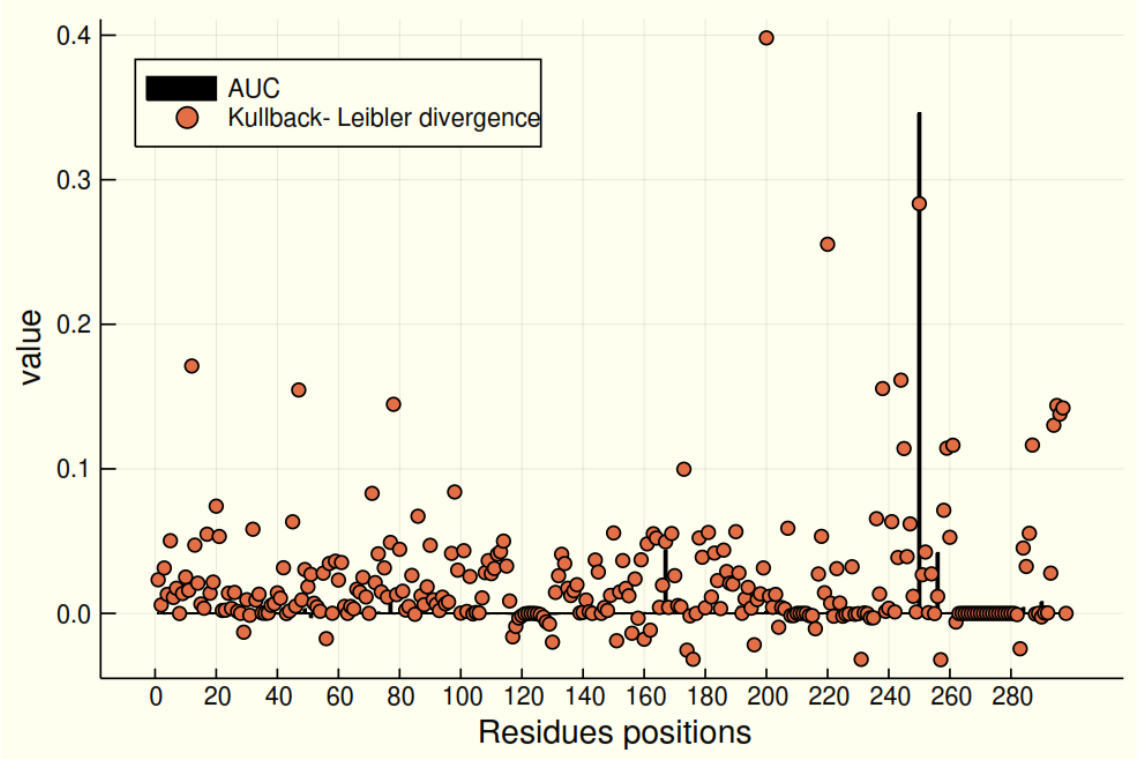

Figure 10. Correlation between Naive approach and D_{KL} approach

The length of the MSA is 298. In orange, Kullback-Leibler divergence values of each column. In black bars, the difference in the Area Under the Curve when a column was removed with respect to the previous AUC value: $dAUC_i = AUC_{i-1} - AUC_i$. The Pearson correlation coefficient is 0.34, p value=1.45 x 10^{-9}.

Study of the importance of the dataset

When progressively reducing the size of the learning database by 20 sequences each time, (initially M=1309), the performance classification remain constant until M ≈ 300 (Figure 11). If only the residues corresponding to the VL domain are used, the results overall are better than the VH domain.

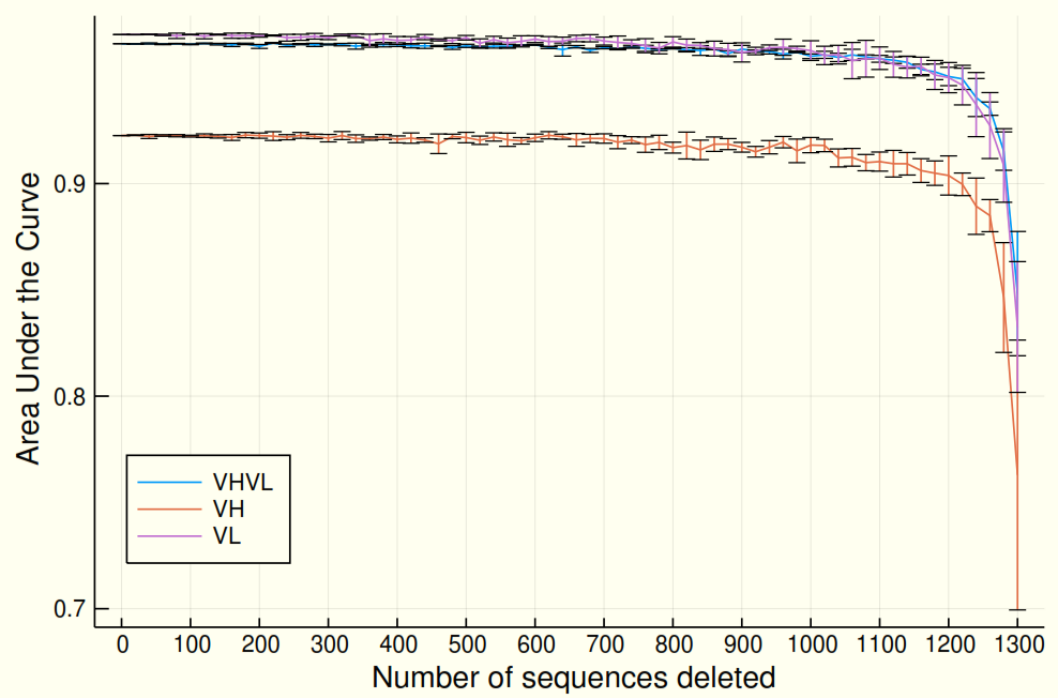

Figure 11. AUC as a function of the size of the VHVL learning database

Average Area Under the Curve (AUC) along the reduction of the VH-VL learning dataset with a total of 1,309 sequences. Results are shown for the whole MSA (VH-VL) and for only VH and only VL. In the three cases, sequences are the same but only the columns to the corresponding domain are used. Standard deviations are represented by confidence bars. Each time, the number of sequences was reduced by 20.

The results of the VH learning dataset (M=7,720) and the VL learning dataset (M=3,723) are also very similar (Figures 12 and 13): only when the size of the dataset is approximately 200, AUC starts to fall. It can be confirmed that the VL domain performs better in the classification even with a smaller learning dataset (Table 9).

	Average Area Under the Curve (AUC)				
Number of sequences	**VH-VL**	**VH of VH-VL dataset**	**VL of VH-VL dataset**	**VH**	**VL**
7700	-	-	-	0.925	-
3700	-	-	-	0.923	0.971
1300	0.966	0.923	0.97	0.923	0.97
200	0.958	0.91	0.959	0.91	0.963
20	0.848	0.762	0.832	0.844	0.907

Table 9. AUC of the different learning datasets

The number of sequences was randomly reduced and then an average AUC was calculated. Here only 5 different-size datasets are shown.

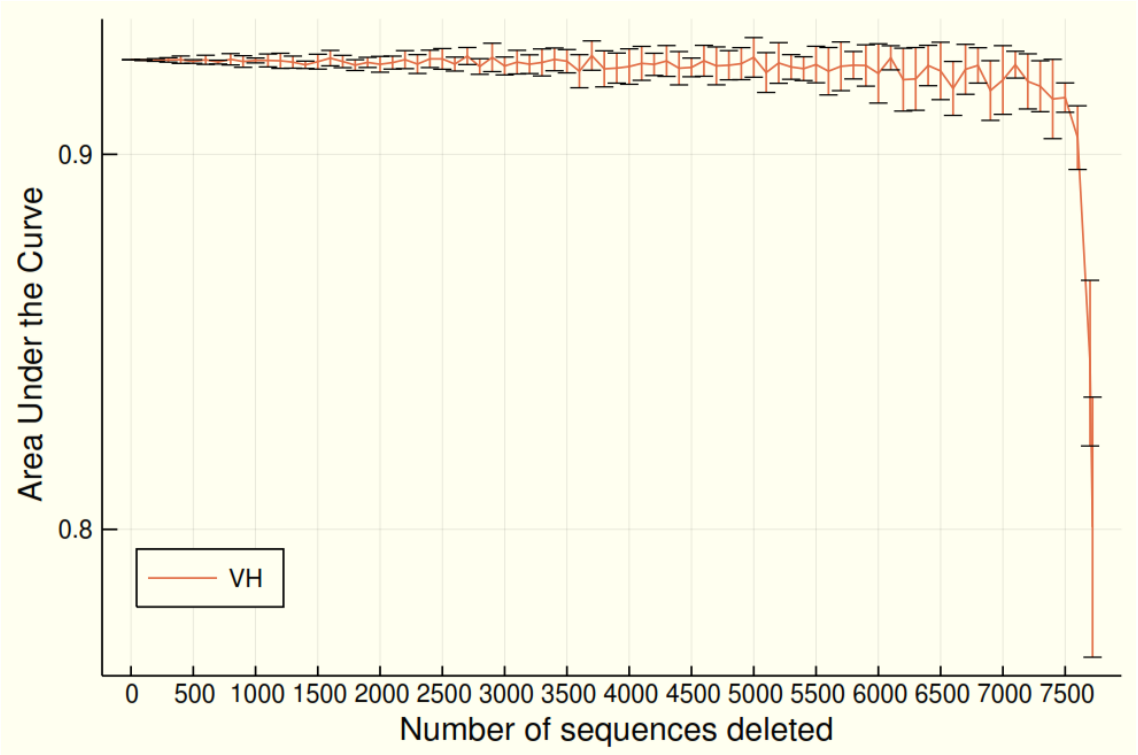

Figure 12. AUC as a function of the size of the VH learning database
Average Area Under the Curve (AUC) along the reduction of the VH learning dataset with a total of 7,720 sequences. Standard deviation is represented by bars. Each time, the number of sequences was reduced by 100.

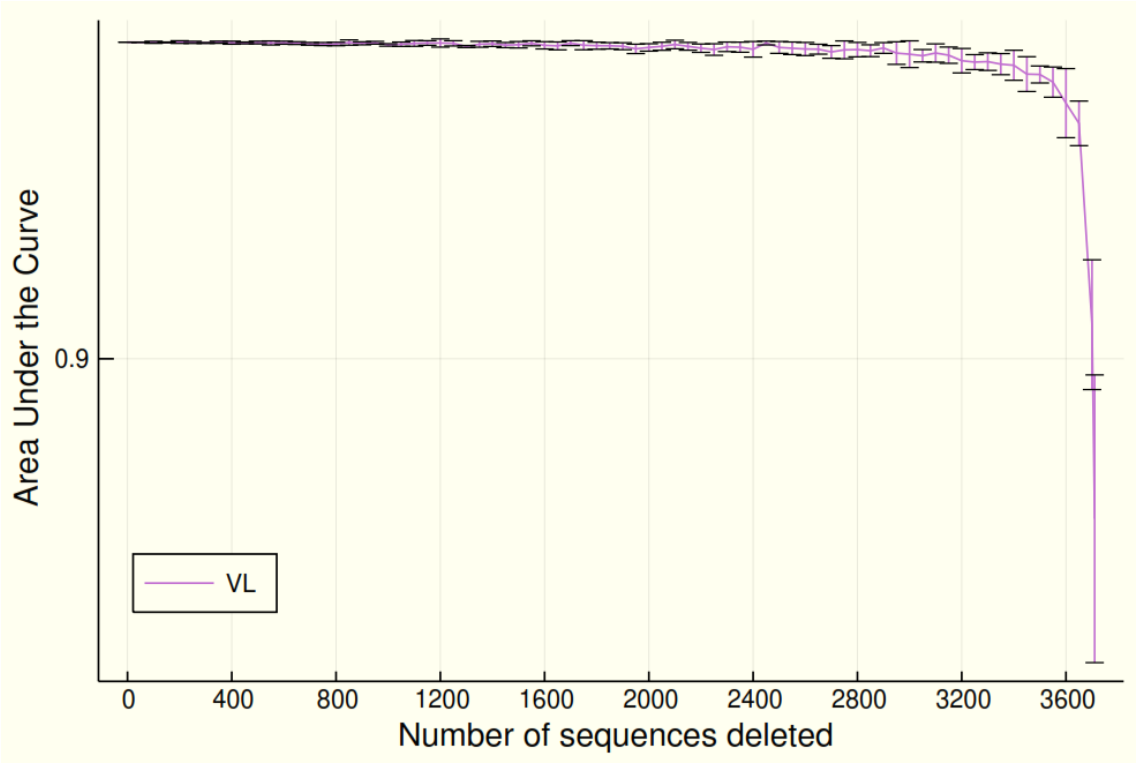

Figure 13. AUC as a function of the size of the VL learning database
Average Area Under the Curve (AUC) along the reduction of the VH learning dataset with a total of 3,723 sequences. Standard deviation is represented by bars. Each time, the number of sequences was reduced by 50.

In general, it seems that not many sequences in the learning dataset are necessary to perform a good classification while we expected a steeper decline in the prediction power.

On the one hand, it could be related to the entropy results of the previous section. They have shown that there are a few columns in the MSA that are sufficient to differentiate both classes of sequences. Although the dataset is progressively reduced, the important columns might keep the crucial information for a successful prediction. The results in the previous section could also explain why the VL sequences perform always better than the VH sequences even when the size of the VH dataset is much larger (with a size difference of 7500 sequences, VL still predict slightly better). Indeed, the distribution of the top columns in table 8 reveals that there are more important residues located in the VL domain (positions 140-298). According to the Kullback-Leibler divergence, only 6 of the top 20 position are located in VH (Table 6) and the top 3 most important residues depending on the AUC values correspond to VL (167, 256 and 250). In the case of S and φ_j there is no a significant difference in the distribution of the most relevance columns, although these two methods perform worse in the prediction when columns were removed in their corresponding order.

On the other hand, we could think, based on the observed results, that the weight of the prior distribution U weight is greater than the empirical parameters, as reducing sequences does not affect significantly the prediction power. However, λ values are quite low, indicating that the empirical mean $\bar{x}$ and covariance $\bar{C}$ are not annulled: $\langle\mu\rangle_{post} = \lambda\eta + (1-\lambda)\bar{x}$, $\langle\Sigma\rangle_{post} = \lambda U + (1-\lambda)\bar{C} + \lambda(1-\lambda)(\bar{x}-\eta)(\bar{x}-\eta)^T$.

MG-score correction

Applying the score correction proposed by Asti and coworkers to correct for gap effects (see Methods) we see that there is no considerable difference between the original MG-score and the adjusted score in the VHVL dataset. The performance classification slightly improves when applying the gap corrected MG-score for the VH sequences, but worsens a little for the VL region (Table 10). Overall, there is no significant difference with the gap correction.

	Area Under the Curve (AUC)				
	VH-VL	**VH of VH-VL dataset**	**VL of VH-VL dataset**	**VH**	**VL**
Original MG-score	0.9658	0.9226	0.97	0.9252	0.971
Gap corrected MG-score	0.9664	0.9487	0.9674	0.9476	0.9633

Table 10. AUC using the original and the gap corrected MG-score

The performance classification is compared for different datasets. An average random MG-score is calculated using the original sequences, but changing randomly the residues and maintaining the gaps positions. Then subtract from the original scores the random score.

In Ref [3] upon implementing the gap corrected score, the prediction of the binding affinities of the antibodies against a specific antigen improved. However, in this case, the aim is to improve the model in order to successfully differentiate human sequences from other species. As the results show, gaps in the MSA of the datasets do not affect the humanness score obtained with the MG model in terms of correct human and murine predictions.

Conclusions

In the creation of antibody therapeutics, the antibody usually developed in mouse is humanized by Complementary-Determining Regions (CDR) grafting. This technique, however, still requires additionally mutations until it has the stability and functionality required as well as the necessary immunogenicity and safety.

In the recent years, several attempts have been made to improve the humanization process, most of which requiring on some kind of humanness score. Humanness scores of the variable region have been proposed based on the similarity/difference between human and murine sequences. In the present work, we have focused on the humanness score provided by a statistical approach based on a Multivariate Gaussian Distribution, in which the discrete amino acid variables are replaced by continuous Gaussian variable.

In previous tests, the MG model has proved to be better than competitors in the classification of human and murine sequences, and has provided promising humanized candidates.
However, the model had not been analyzed in details, to understand what are its strength and limits.
In the present work, we have analyzed the model to understand the meaning of the inferred parameters, trying to understand if it could benefit of shorter multiple sequence alignments or from a change in the number of sequences in the datasets, and trying to refine it with score modifications, to compensate for the influence of gaps. The results we obtain are interesting and somewhat surprising, even if they don't clarify completely what are the principal features of the model.

By construction, the inverse covariance matrix Σ^{-1} of the Multivariate Gaussian Distribution represents an effective interaction between pairs of residues. To study if these interactions are related to physical interactions, due to the protein structure, we have thoroughly compared the interaction matrix with the average contact maps obtained from a set of 58 structures at difference distance cutoffs, as a crude representation of the structural interactions. However, the correlation between the interaction matrix and an average contact map generated by PDB sequences is weak. The relatively low number of sequences in the dataset, the distinctive variability of the CDRs and the high presence of gaps could be factors that affect the correlation. Even if the most likely explanation, already put forward in Ref. [3], is that for antibodies, the effective interactions do not reflect faithfully physical interactions, and are deeply influenced by other factors, related to the process of generation of the antibody sequence.

Then, we resorted to the study of the importance of the different columns in the MSA, corresponding to different residues positions. Our goal was to understand if there are columns that emerge are more relevant than others, and we had to limit ourselves to the classification task, since it is much more difficult to quantify the performance of the MG model for the humanization task, while for the classification task, the Area Under the Curve in the ROC diagram, when classifying sequences from the human and murine test databases, is a reliable indicator. So, to analyze the importance of the columns of the MSA in the classification of human and murine sequences, different column properties were studied: site entropy, the Kullback-Leibler divergence between murine and human empirical distributions and "energy bias" at each position trying to understand which of them is more important in predicting the relevance of the columns.

Following a naive approach, we calculated at the beginning which column causes, upon removal, the biggest change in AUC, finding that the most relevant columns (only with 5 columns AUC is already larger than 0.95) are mostly conserved residues or located in the VHVL surface and present a clear difference in residue frequencies between the two classes. These columns correspond to 45, 51, 167, 250 and 256.

On the contrary, we observed that the majority of the most site entropic columns are located in the CDRs regions, but when we deleted the columns following this order, the model soon cannot longer distinguish between human and murine sequences. Therefore, these hypervariable regions do not carry the greatest information for classification task.

By sorting the columns from the least to the highest Kullback-Leibler divergence and progressively reducing the MSA length according to this order, we notice that two columns with the highest D_{KL} are already able to clearly distinguish between the two species, which are 200 and 250, located in the FRs of VL domain.

Although Kullback-Leibler divergence is the most effective in identifying the relevant residues, it depends on the murine and human distribution, which means that it could not be calculated in a classification task between human and other non-human species. That is why we also proposed an energy bias indicator (defined as the difference between an external field term and an interaction term), that depend only on human distribution, as the site entropy, but performed a better classification than the latter.

We also compare the order of relevance of the columns according to the two best indicators, AUC and Kullback-Leibler divergence, but a weak correlation was found. So, in the end we were not able to find columns that can be safely eliminated from the MSA for a generic classification purpose, even if criteria based on the energy bias (for generic classification) or KL divergence (for classification in just to classes) could be devised with a good expectation of classification performance. We believe that interactions between residues might be important in the classification and taking into consideration a single column is not enough to find important positions.

Furthermore, one of the main limitations in the field is the low number of antibody's sequences available so we studied the predictive power of the statistical model as a function of the size of the learning dataset. Surprisingly, the results show that approximately 200 sequences are enough to perform a good classification although a steeper decline was expected. We suspect that this is due to the fact that since in the end a few columns are sufficient to distinguish the two species, this reduces the amount of information needed so that a successful prediction is possible even when the dataset is highly reduced. When independently larger VH and VL datasets are used, we verify that VL domain performs better in the classification, which could also be related to the most important columns, as the majority of them are located in VL.

Finally, we applied a gap correction in the MG score, but, contrary to other MG score applications in previous researches, no difference in the correct prediction was found. Hence, even with a high number of gaps in the Multiple Sequence Alignment, they affect the final score but not the performance of the model.

These results are not conclusive, and prompt for more research, for instance to study the performance of the model in the classification task when different type of antibodies (and not just murine and human) are involved, and to try to understand better the relative role of interactions and site propensities as they emerge from the model.

References

[1] Ecker, D.M., Jones, S.D. & Levine, H.L. (2015). The therapeutic monoclonal antibody market. mAbs, 7, 9-14. DOI: 10.4161/19420862.2015.989042.

[2]Bezanson, J., Edelman, A., Karkpinski, S. & Shah, V.B. (2017). Julia: A Fresh Approach to Numerical Computing. SIAM Review, 59, 65-98. DOI: 10.1137/141000671.

[3] Asti, L., Uguzzoni, G., Marcatili, P. & Pagnani, A. (2016). Maximum-Entropy Models of Sequenced Immune Repertories Predict Antigen-Antibody Affinity. PLoS Computational Biology, 12(4), e1004870. DOI: 10.1371/journal.pcbi.1004870.

[4] Baldassi, C., Zamparo, M., Feinauer, C., Procaccini, A., Zecchina, R., Weigh, M. & Pagnani, A. (2014). Fast and accurate Multivariate Gaussian Modeling of proteins families: predicting residue contacts and protein interactions partners. PLoS ONE, 9(3), e92721. DOI: 10.1371/journal.pone.0092721.

[5] Petering, J., Mcmanamny, P., & Honeyman, J. (2011). Antibody therapeutics – the evolving patent landscape. New Biotechnology, 28(5), 538-544. doi:10.1016/j.nbt.2011.03.023

[6] Fischman, S., & Ofran, Y. (2018). Computational design of antibodies. Current Opinion in Structural Biology, 51, 156-162. doi:10.1016/j.sbi.2018.04.007

[7] Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., Zecchina, R., Onuchic, J.N., Hwa, T. & Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proceedings of the National Academy of Sciences, 108(49). doi:10.1073/pnas.1111471108

[8] Ekeberg, M., Lövkvist, C., Lan, Y., Weigt, M., & Aurell, E. (2013). Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models. Physical Review E, 87(1), 012707. doi:10.1103/physreve.87.012707

[9] Weigt, M., White, R. A., Szurmant, H., Hoch, J. A., & Hwa, T. (2008). Identification of direct residue contacts in protein-protein interaction by message passing. Proceedings of the National Academy of Sciences, 106(1), 67-72. doi:10.1073/pnas.0805923106

[10] Hwang, W. Y., & Foote, J. (2005). Immunogenicity of engineered antibodies. Methods, 36(1), 3-10. doi:10.1016/j.ymeth.2005.01.001

[11] Abhinandan, K., & Martin, A. C. (2007). Analyzing the “Degree of Humanness” of Antibody Sequences. Journal of Molecular Biology, 369(3), 852-862. doi:10.1016/j.jmb.2007.02.100

[12] Pelat, T., Bedouelle, H., Rees, A. R., Crennell, S. J., Lefranc, M., & Thullier, P. (2008). Germline Humanization of a Non-human Primate Antibody that Neutralizes the Anthrax Toxin, by in Vitro and in Silico Engineering. Journal of Molecular Biology, 384(5), 1400-1407. doi:10.1016/j.jmb.2008.10.033

[13] Thullier, P., Huish, O., Pelat, T., & Martin, A. C. (2010). The Humanness of Macaque Antibody Sequences. Journal of Molecular Biology, 396(5), 1439-1450. doi:10.1016/j.jmb.2009.12.041

[14] Gao, S. H., Huang, K., Tu, H., & Adler, A. S. (2013). Monoclonal antibody humanness score and its applications. BMC Biotechnology, 13(1), 55. doi:10.1186/1472-6750-13-55.

[15] Jones, D. T., Buchan, D. W., Cozzetto, D., & Pontil, M. (2012). PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. Bioinformatics, 28(2), 184-190. doi:10.1093/bioinformatics/btr638
[16] Alberts B, Johnson A, Lewis J, *et al*. (2002) Molecular Biology of the Cell. 4th edition. New York: Garland Science. The Generation of Antibody Diversity. Available from: https://www.ncbi.nlm.nih.gov/books/NBK26860/
[17] Ill, C. R., Gonzales, J. N., Houtz, E. K., Ludwig, J. R., Melcher, E. D., Hale, J. E., … Radhakrishnan, R. (1997). Design and construction of a hybrid immunoglobulin domain with properties of both heavy and light chain variable regions. Protein Engineering Design and Selection, 10(8), 949–957. doi: 10.1093/protein/10.8.949
[18] Clavero-Álvarez, A., Di Mambro, T., Pérez-Gaviro, S., Magnani, M. & Bruscolini, P. (2018) Humanization of Antibodies using a Statistical Inference Approach. Scientific reports, 8, 14820. DOI: 10.1038/s41598-018-32986-y.
[19] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne. (2000). The Protein Data Bank. Nucleic Acids Research, 28: 235-242. http://www.rcsb.org/
[20] The PyMOL Molecular Graphics System, Version 2.3.2 Schrödinger, LLC.
[21] Dunbar, J. & Deane, C.M. (2016). ANARCI: antigen receptor numbering and receptor classification. Bioinformatics, 32, 298-300. DOI: 10.1093/bioinformatics/btv552.
[22] R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.
[23] Grant, B. J., Rodrigues, A. P., Elsawy, K. M., Mccammon, J. A., & Caves, L. S. (2006). Bio3d: An R package for the comparative analysis of protein structures. Bioinformatics, 22(21), 2695-2696. doi:10.1093/bioinformatics/btl461
[24] Honegger, A., & Plückthun, A. (2001). Yet Another Numbering Scheme for Immunoglobulin Variable Domains: An Automatic Modeling and Analysis Tool. Journal of Molecular Biology, 309(3), 657-670. doi:10.1006/jmbi.2001.4662
[25] Herold, E. M., John, C., Weber, B., Kremser, S., Eras, J., Berner, C., Deubler,S., Zacharias, M. & Buchner, J. (2017). Determinants of the assembly and function of antibody variable domains. Scientific Reports, 7(1). doi: 10.1038/s41598-017-12519-9
[26] Wang, N., Smith, W. F., Miller, B. R., Aivazian, D., Lugovskoy, A. A., Reff, M. E., Croner, L. J. & Demarest, S. J. (2009). Conserved amino acid networks involved in antibody variable domain interactions. Proteins: Structure, Function, and Bioinformatics, 76(1), 99-114. doi:10.1002/prot.22319
[27] Wu, T. T., & Kabat, E. A. (1970). An Analysis Of The Sequences Of The Variable Regions Of Bence Jones Proteins And Myeloma Light Chains And Their Implications For Antibody Complementarity. Journal of Experimental Medicine, 132(2), 211–250. doi: 10.1084/jem.132.2.211.
[28] Honegger, A. (2008). Engineering Antibodies for Stability and Efficient Folding. Therapeutic Antibodies Handbook of Experimental Pharmacology, 47–68. doi: 10.1007/978-3-540-73259-4_3
[29] Feinauer, C., Skwark, M. J., Pagnani, A., & Aurell, E. (2014). Improving Contact Prediction along Three Dimensions. PLoS Computational Biology, 10(10). doi: 10.1371/journal.pcbi.1003847
[30] Dondelinger, M., Filée, P., Sauvage, E., Quinting, B., Muyldermans, S., Galleni, M., & Vandevenne, M. S. (2018). Understanding the Significance and Implications of Antibody Numbering and Antigen-Binding Surface/Residue Definition. Frontiers in Immunology, 9. doi: 10.3389/fimmu.2018.02278

Appendix

Julia code to obtain the covariance matrix Σ of LxL dimension from a MSA file

```
#Code to generate Σ^-1

###Packages
using LinearAlgebra
using DataFrames
using CSV

###Arguments
filename = ARGS[1] #Name of the learning database MSA file
λ   = parse(Float64, ARGS[2]) #Value of  λ
Ω   = parse(Float64, ARGS[3]) #Value of Ω

###Generate the covariance matrix Σ
seqmatrix = inputtoascii(filename) #Read MSA files and return matrix of characters
μ, Σ = posterior(seqmatrix, λ = λ, customΩ = Ω) #Mean and covariance a posteriori
invsig=inv(Σ) #Inverse of the covariance matrix Σ

###Compute the Frobenius Norm
   L=298
   Q=20
   S=zeros(Float64,L,L)
   for i=1:L
      for j=i+1:L
         for α=1:Q
            for β=1:Q
               k=(i-1)*Q + α
               l=(j-1)*Q + β
               S[i,j]=S[i,j]+(invsig[k,l]^2)
            end
         end
S[i,j]=sqrt(S[i,j])/Q
S[j,i]=S[i,j]
      end
   end

#Obtain upper triangular matrix
for i=1:L
   for j=1:L
      if i > j
         S[i,j]=0
      end
   end
end

###Save as CSV file
df = DataFrame(S)
CSV.write("filename.csv",df,writeheader=true)
```

R code to generate an average contact matrix

```
#Code to generate average contact map for different thresholds of VH-VL sequence with AHo
#numbering from PDB files
library(bio3d) #package to analyze protein structure and sequence data

###Select the variables for the contact map
assign("nfi","1IC4") #PDB ID (the 4-character identifier), example: 1IC4
assign("thres",seq(1.4, 50, by=.1)) #Threshold interval in Å
pdb <- read.pdb(nfi) #read PDB file from RCSB online database
VH.inds <- atom.select(pdb,chain="H",type="ATOM") #Select only atoms from VH domain
VL.inds <- atom.select(pdb,chain="L",type="ATOM") #Select only atoms from VL domain
#Warning: each PDB file has different names for the chains, change the letter assigned in each
#case
VHVL.inds <- combine.select(VH.inds,VL.inds,operator="+") #Combine both variable domains

###Generate contact map for each threshold
#Define variables as list
cm.mg <- list()
contacts <-list()
D <- list()
for (i in thres) {
  cm.mg[[i]] <- cmap(pdb,inds=VHVL.inds,dcut=i,scut=0) #function to construct contact map
  #dcut: cutoff distance value, scut: cutoff neighbour value
  dime <- dim(cm.mg[[i]])[c(1)] #dimension of contact matrix
  contacts[[i]] <- which(cm.mg[[i]] !=0, arr.ind = T) #obtain indices where there is a contact
  new <- read.table(paste(nfi, "txt", sep=".")) #read table of corresponding indices according to
#AHo numbering scheme. Warning: a text document with new indexes from ANARCII is needed for
#each PDB
  new2 <- new[c(1:dime),c(1)] #transform new indexes from list to integer
  if (dim(cm.mg[[i]]) != dim(new)) {
    print("different length between old indexes and new indexes")
    break  #Check all corresponding indexes are given, according to the length of contact matrix
  }
  old <-  1:dime #select old indexes
  #Change from old indexes to new indexes
  contacts[[i]][contacts[[i]] %in% old] <- new2[match(contacts[[i]], old, nomatch = 0)]
  D[[i]]=matrix(0,nrow=298,ncol=298) #Initialize L x L matrix
  D[[i]][contacts[[i]]]=1 #Put contacts according to new indexes
  #Write contact matrix in a file with name depending on PDB ID and threshold
  write.csv(D[[i]],paste0(nfi,"_D_",i, ".csv"), row.names = FALSE, quote=FALSE)
}

###Calculate average contact map
#Define variables as list
read_my_files <- list()
average_my_files <-list()
for (i in thres) {
  all_my_files <- list.files(pattern=paste0("_D_",i, ".csv")) #Select all files for each threshold
  read_my_files[[i]] <- lapply(all_my_files, read.csv) #Read selected files
  average_my_files[[i]] <- Reduce("+", read_my_files[[i]]) / length(read_my_files[[i]]) #Calculate
the average matrix
  #Write average contact matrix in a file with name depending on threshold
  write.csv(average_my_files[[i]],paste0("aveD_",i, ".csv"),row.names=FALSE,quote=FALSE)
}
```

R code to calculate the Pearson correlation coefficient of Σ and contact map

```
#Code to determine the Pearson correlation coefficient of covariance matrix and average contact
#map
###Define variables
assign("thres",seq(1.4, 50, by=.1))
aveD <- list()
contacts <-list()
threshold <- list()
r <- list()
sp <- list()

###Calculate Pearson correlation coefficient for each average contact map
for (i in thres) {
  S=read.csv("covariancematrixfilename.csv") #Read file with covariance matrix
  Svector <- as.vector(t(S))
  Ssorted <- sort(Svector,decreasing=TRUE) #Sort in descending order all values of S
  aveD[[i]] <- read.csv(paste("aveD_",i,".csv",sep="")) #Read file with average contact map
  contacts[[i]] <- length(which(aveD[[i]] > 0)) #Count number of contacts in average contact map
  threshold[[i]] <- Ssorted[contacts[[i]]] #Calculate threshold for such number of contacts
    for (k in 1:nrow(S)) {
      for (l in 1:ncol(S)) {
        if (S[k,l] < threshold[[i]]) {
          S[k,l]=0 #Leave only contacts above the threshold
          }
        }
    }
   S[lower.tri(S)] <- NA #delete values in lower triangular
  aveD[[i]][lower.tri(aveD[[i]])] <-NA
  r[[i]] <- cor(c(as.matrix(S)), c(as.matrix(aveD[[i]])),method="pearson",use="complete.obs")
#only select Non NA positions
  #Pearson correlation of the two matrices
  sp[[i]] <- cor(c(as.matrix(S)), c(as.matrix(aveD[[i]])),method="spearman",use="complete.obs")
  #Spearman correlation of the two matrices
}
}

#Save in CSV file
write.csv(r,paste0("newfilename.csv"), row.names = FALSE, quote=FALSE)
write.csv(sp,paste0("newfilename.csv"), row.names = FALSE, quote=FALSE)

#Histogram of the covariance matrix
S=read.csv("filename.csv") #Read matrix file
Svector <- as.vector(t(S)) #Change to vector mode
hist(Svector,main="Mouse",xlab="Value",ylab="Number of occurrences", breaks=1000,
xlim=c(0,40),col="gray")#Computes histogram
```

Julia code to calculate the entropy

```
#Code to calculate the entropy of each column

###Packages
using DelimitedFiles
using StatsBase
using Plots
using StatsPlots
```

```
###Information entropy given by S = − ∑_i p_i log p_i
seqmatrix = inputtoascii("filename.seqs") #read MSA file and return matrix of characters
(M,L)=size(seqmatrix)
Stot=[]
for i in 1:L
   d=proportionmap(seqmatrix[:,i]) #returns dictionary with proportions of characters
   S=[]
   for p in values(d)
      push!(S,p*log(p)) #Array with entropy of each amino acid character
   end
   push!(Stot,-sum(S)) #Array with total entropy of each column
end

Stotsorted= sortperm(Stot,rev=true) #returns indices that put the array into sorted order

writedlm("entropy_per_column_filename.txt", Stot) #save entropy of each column in txt file
writedlm("high_to_low_entropy_filename.txt",Stotsorted)
#save indices of columns from highest to lowest entropy in txt file

### Kullback–Leibler divergence
human = inputtoascii("human_filename.seqs") #read human MSA file and return matrix of
#characters
mouse = inputtoascii("mouse_filename.seqs") #murine MSA file
(M,L)=size(human)
(m,L)=size(mouse)
Dtot=[] #array to save total KL divergence
for i in 1:L #for each column of both MSA
#returns dictionary with proportions of characters of each column of human and murine
      d=proportionmap(human[:,i])
      e=proportionmap(mouse[:,i])
      D=[] #array to save KL diveregence of each character
      for k in keys(d)
         for l in keys(e)
            if k==l  #only when a character is present in both dicionaries
               p=d[k] #select proportion of that character in the human dict
               q=e[l] #select proportion of that character in the murine dict
               push!(D,p*log(p/q)) #Array with entropy of each amino acid
            end
         end
      end
   push!(Dtot,sum(D)) #Array with total entropy of each column
end

Dtotsorted= sortperm(Dtot,rev=true) #returns indices that put the array into sorted order

writedlm("Kullback_entropy_per column.txt", Dtot) #save entropy of each column in txt file
writedlm("Kullback_high_to_low_entropy.txt",Dtotsorted)
#save indices of columns from highest to lowest entropy in txt file

###Plot
x=[1:1:298;] #columns of the MSA
y=readdlm("entropy_per_column_filename.txt")[:,1] #read file with entropy per column
#make a bar plot
```

```
bar(x,y,xticks=0:20:298,legend=false,ylabel = "Entropy", xlabel = "Residues positions",
background_color = :ivory, color=:gray,linecolor=:gray,title="Human")
savefig("newfigurename.png") #save as png format
human=readdlm("entropy_human_filename.txt")[:,1] #read file with human entropy
mouse=readdlm("entropy_mouse_filename.txt")[:,1] #read file with mouse entropy
y=human-mouse #perform the difference of entropy per column
bar(x,y,xticks=0:20:298,legend=false,ylabel = "Entropy",
xlabel = "Residues positions", background_color = :ivory, color=:gray,
linecolor=:gray,title="Difference between human and mouse entropy")
savefig("newfigurename.png") #save as png format

#Code to calculate φ=vj-hj

###Packages
using LinearAlgebra
using DelimitedFiles

###Parameters of the model
seqmatrix = inputtoascii("exthuman_jointVHVL_AHo_final.seqs")
#Read MSA files and return matrix of characters
(M,L)=size(seqmatrix)
λ=0.1
Ω=0.4898
μ, Σ = posterior(seqmatrix, λ = λ, customΩ = Ω)
#Mean and covariance a posteriori
invsig=inv(Σ) #Inverse of the covariance matrix Σ
(N,N)=size(invsig)

###Calculate h (external field)
h=[]
for i in 1:N
   k=μ[i].*invsig[i,:]
   l=sum(k)
   push!(h,l)
end
hj=2*h
writedlm("Hvalues.txt",hj)

###Calculate v (interaction between positions)
x=asciitobinary(seqmatrix) #convert to binary matrix
vseq=[]
for j in 1:M
   for i in 1:N
      k=x[j,i].*invsig[i,:]
      l=sum(k)
      push!(vseq,l)
   end
end
vj=reshape(vseq, (N, div(length(vseq), N))) #divide by sequence
vf=(sum(vj,dims=2))/M #sum over all sequences to calculate average
writedlm("vvalues.txt",vf)
φ=abs.(vj-hj) #absolute value of the difference v-h
Q=20
φ=reshape(φ, (Q, div(length(φ), Q))) #reshape in blocks of 20 (each column one position)
φj=sum(φ,dims=1) #sum all elements of each column
```

```
writedlm("Importance_per_column.txt",vec(φj))
SortedCols = sortperm(vec(φj)) #Sorted from least to most relevance
writedlm("low_to_high_importance.txt",SortedCols)
```

Julia code to study the classification performance by eliminating columns

```
#Code to calculate Area Under the Curve after reducing number of columns in the MSA

#Packages
using LinearAlgebra
using SpecialFunctions
using DataFrames
using DelimitedFiles
using ROCAnalysis
using Plots

###Arguments
filename_human_learning = ARGS[1] #Name of the human learning database MSA file
filename_human_test = ARGS[2] #Name of the human test database MSA file
filename_mouse_test = ARGS[3] #Name of the human test database MSA file
λ = parse(Float64, ARGS[4]) #Value of λ
Ω = parse(Float64, ARGS[5]) #Value of Ω

###Progressively eliminate selected columns
c=readdlm("low_to_high_entropy.txt",Int64)[:,1] #read file with sorted columns
L=[1:1:298;] #length of the MSA
AUC_output=[] #array to save results
for i in L
    k=c[1:i] #select columns to delete
#Learning human dataset
   learnhuman = DataFrame(inputtoascii("filename_human_learning"))
   deletecols!(learnhuman, k)
   learnhuman=convert(Matrix,learnhuman)
#Test human dataset
   testhuman= DataFrame(inputtoascii("filename_human_test"))
   deletecols!(testhuman, k)
   testhuman=convert(Matrix,testhuman)
#Test mouse dataset
   testmouse= DataFrame(inputtoascii("filename_mouse_test"))
   deletecols!(testmouse, k)
   testmouse=convert(Matrix,testmouse)

###Generate new parameters
M = size(learnhuman, 1)
μ, Σ = posterior(learnhuman, λ = λ  , customΩ = Ω)
N = length(μ)
invsig=inv(Σ)
(m,n)=size(invsig)
logdetinvsig = logdet(invsig)

###Calculate scores of test databases
#Scores of human dataset
Nseqsh, Lseqh = size(testhuman)
Ph=zeros(Nseqsh)
```

```
for i in 1:Nseqsh
   x = asciitobinary(testhuman[i, :]) #converts character matrix into its binary representation
   logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig) #returns the natural logarithm of the
#probability of the peptide in the statistical model (a Student's t distribution)
   Ph[i]=logP #array with score of each peptide of the human test dataset
end
#Scores of mouse dataset
Nseqsm, Lseqm = size(testmouse)
Pm=zeros(Nseqsm)
for i in 1:Nseqsm
   x = asciitobinary(testmouse[i, :])
   logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig)
   Pm[i]=logP #array with score of each peptide of the human test dataset
end

###Perform ROC analysis
r=roc(Pm,Ph) #computes statistics for evaluation of the performance of a two-class classifier.
AUC=auc(r) #calculates Area Under the Curve
push!(AUC_output, AUC) #add each new AUC to the array
end
writedlm("newfilename.txt", AUC_output) #save results in txt file

###Sort AUC values just when only 1 column was deleted each time
AUCcols=readdlm("newfilename.txt", Float64) #read file with AUC results
AUCsorted=sortperm(AUCcols[:,1],rev=true) #place in order from high to low the array's indices
writedlm("newfilename.txt", AUCsorted) #save results in txt file

###Plot
S=readdlm("AUC_Entropy.txt")[:,1] #file with AUC according to S entropy
D=readdlm("AUC_Kullback.txt")[:,1] #file with AUC according to KL divergence
AUC=readdlm("AUC_by_deleting.txt")[:,1] #file with AUC according to deletion one by one
Phi=readdlm("AUC_Phi.txt")[:,1] #file with AUC according to φ
x=[1:1:298;] #Columns in the MSA
plot(x,S,xticks=0:28:298, yticks = 0:0.1:1, label="Information Entropy",legend = :bottomleft,
ylabel = "Area Under the Curve", xlabel = "Number of columns deleted", background_color =
:ivory) #S entropy
plot!(x,D,xticks=0:28:298,label="Kullback-Liebler divergence", yticks = 0:0.1:1, ylabel = "Area
Under the Curve", xlabel = "Number of columns deleted", background_color = :ivory) #KL
plot!(x,AUC,xticks=0:28:298,label="AUC", yticks = 0:0.1:1,
ylabel = "Area Under the Curve", xlabel = "Number of columns deleted",
background_color = :ivory) #AUC
plot!(x,Phi,xticks=0:28:298,label="hj-vj", yticks = 0:0.1:1,
ylabel = "Area Under the Curve", xlabel = "Number of columns deleted",
background_color = :ivory) #ϕ
savefig("newfigurename.png") #save in png format
```

Julia code to study the model as a function of the learning database

```
#Code to calculate Area Under the Curve after reducing number of sequences in the learning MSA

###Packages
using LinearAlgebra
using SpecialFunctions
```

```
using DataFrames
using DelimitedFiles
using StatsBase
using Statistics
using ROCAnalysis
using Plots

###Arguments
filename_human_learning = ARGS[1] #Name of the human learning database MSA file
filename_human_test = ARGS[2] #Name of the human test database MSA file
filename_mouse_test = ARGS[3] #Name of the human test database MSA file
λ = parse(Float64, ARGS[4]) #Value of λ
#dataset VHVL λ=0.1    only VH λ=0.3    only VL λ=0.1
#dataset VH λ=0.4
#dataset VL λ=0.1
Ω = parse(Float64, ARGS[5]) #Value of Ω
#dataset VHVL Ω =0.4898    VH Ω =0.5102    VL Ω =0.4286
#dataset VH  Ω =0.4286
#dataset VL   Ω =0.2250

###Reduce randomly the number of sequences in the learning database
#return matrix of characters in dataframe format
learningdata = DataFrame(inputtoascii("filename_human_learning"))
deleteseqs=[0:20:1300;] #array with range of sequences to delete from learning database
timesdelete=repeat(deleteseqs,20) #array with sequences to delete multiple times (20)
(k,l)=size(learningdata) #rows and columns of the learning database
AUC_output=[] #array to save results
for i in timesdelete
s=k-i #Number of sequences sample must contain
reduceddata=learningdata[sample(axes(learningdata, 1), s; replace = false), :] #Select randomly
#d rows
learnhuman=convert(Matrix,reduceddata)
#Read test databases
testhuman= inputtoascii("filename_human_test")
testmouse= inputtoascii("filename_mouse_test")

###Generate new parameters
M = size(learnhuman, 1)
μ, Σ = posterior(learnhuman, λ = λ  , customΩ = Ω)
N = length(μ)
invsig=inv(Σ)
(m,n)=size(invsig)
logdetinvsig = logdet(invsig)

###Calculate scores of test databases
#Scores of human dataset
Nseqsh, Lseqh = size(testhuman)
Ph=zeros(Nseqsh)
for i in 1:Nseqsh
   x = asciitobinary(testhuman[i, :]) #converts character matrix into its binary representation
   logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig) #returns the natural logarithm of the
#probability of the peptide in the statistical model (a Student's t distribution)
   Ph[i]=logP #array with score of each peptide of the human test dataset
end
#Scores of mouse dataset
```

```
Nseqsm, Lseqm = size(testmouse)
Pm=zeros(Nseqsm)
for i in 1:Nseqsm
   x = asciitobinary(testmouse[i, :])
   logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig)
   Pm[i]=logP #array with score of each peptide of the human test dataset
end

###Perform ROC analysis
r=roc(Pm,Ph) #computes statistics for evaluation of the performance of a two-class classifier.
AUC=auc(r) #calculates Area Under the Curve
push!(AUC_output, AUC) #Save each AUC value calculated in an array
print("($i)","($AUC)")
end
writedlm("newfilename.txt", AUC_output) #save array in txt file

###Calculate the average
VHVL1=readdlm("VHVLfilename1.txt",Float64) #read AUC values as matrix
VHVL1=VHVL1[:,1] #to change to 1 dimensional array
#repeat for all n filenames: VHVLfilename2.txt, VHVLfilename3.txt,….VHVLfilenameN.txt
VHVL=hcat(VHVL1,VHVL2,VHVL3,…,VHVLN) #concatenate in one array so that each row contains
all AUC values for the corresponding size
ave_VHVL=mean(VHVL,dims=2) #calculates the mean of each row
std_VHVL=std(VHVL,dims=2) #calculates standard deviation of each row
#repeat for VH and VL

###Plot
plot(d,ave_VHVL,yerror=std_VHVL,xticks=0:100:1300, yticks = 0:0.1:1,
label="VHVL",legend=:bottomleft, ylabel = "Area Under the Curve", xlabel = "Number of
sequences deleted", background_color = :ivory) #VHVL line
plot!(d,ave_VH,yerror=std_VH,xticks=0:100:1300,label="VH", yticks = 0:0.1:1,ylabel = "Area
Under the Curve", xlabel = "Number of sequences deleted", background_color = :ivory) #VH line
plot!(d,ave_VL,yerror=std_VL,xticks=0:100:1300,label="VL", yticks = 0:0.1:1,ylabel = "Area
Under the Curve", xlabel = "Number of sequences deleted", background_color = :ivory) #VL line
savefig("newfigurename.png") #save figure in png format
```

Julia code to optimize Ω

```
#Code to optimize Ω

###Packages
using LinearAlgebra
using DelimitedFiles

###Arguments
Ωrange = range(0.001,stop=0.999,length=50) #construct a range of Ω values
get_norm(Σ) = norm(Σ,2) #function to compute the p-norm of the covariance matrix
file = "filename.seqs" #MSA filename
seqs = inputtoascii(file) #read file and return matrix of characters
seqsB = asciitobinary(seqs) #convert character matrix into its binary representation
outnorm = open("frobnorm_$file.csv","w") #create file to later save the Frobenius norm
outweight = open("weights_$file.txt","w") #create file to later save the weigths

###Calculate Frobenius norm for each Ω value
```

```
for Ω in Ωrange
        w = weight(seqs,customΩ=Ω, memoize=true) # calculate weight of each sequence
        writedlm(outweight, w') #save weights in file
        Σ = empiricalcovariance(seqsB, w) #compute empirical covariance by given weights
        #write results in files
        println(outnorm, file, ",", Ω, ",", get_norm(Σ))
        flush(outnorm)
        flush(outweight)
end

###Find the minimum 1-|Σ|
file=readdlm("frobnorm_filename.csv.seqs.csv", ',') #open file with Ω and its Frobenius norm
fronorm=file [:,3] #read column with frobenius norm values
one=ones(50)
score=one-fronorm #calculate score=1-get_norm(Σ)
(score_min, ind_ Ω) = findmin(score) #find value and position of minimum score
 Ωopt= Ωrange [ind_ Ω]  #find value of Ω of given index
#Results
#VH 0.4286
#VL 0.2250
```

Julia code to optimize λ

```
#Code to optimize λ

###Packages
using DelimitedFiles
using ROCAnalysis
using LinearAlgebra
using SpecialFunctions

###Arguments
filename_human_learning = ARGS[1] #Name of the learning database MSA file
filename_human_test = ARGS[2] #Name of the human test database MSA file
filename_mouse_test = ARGS[3] #Name of the human test database MSA file
Ω = parse(Float64, ARGS[4]) #Value of Ω previously optimized

###Study the classification performance for each λ value
λ_range  = exp10.(range(-3.5,stop=-0.01,length=10)) #construct a range of λ values
AUC_output=[] #array to save results
for λ in λ_range
#Generate new parameters
M = size(filename_human_learning, 1)
μ, Σ = posterior(filename_human_learning, λ = λ  , customΩ = Ω)
N = length(μ)
invsig=inv(Σ)
(m,n)=size(invsig)
logdetinvsig = logdet(invsig)

#Calculate scores of test databases
Nseqsh, Lseqh = size(filename_human_test)
Ph=zeros(Nseqsh)
for i in 1:Nseqsh
   x = asciitobinary(filename_human_test[i, :]) #converts character matrix into its binary
representation
```

```
    logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig) #returns the natural logarithm of the
#probability of the peptide in the statistical model (a Student's t distribution)
    Ph[i]=logP #array with score of each peptide of the human test dataset
end
#Scores of mouse dataset
Nseqsm, Lseqm = size(filename_mouse_test)
Pm=zeros(Nseqsm)
for i in 1:Nseqsm
    x = asciitobinary(filename_mouse_test[i, :])
    logP = prob_in_model(x, λ, M, μ, invsig, logdetinvsig)
    Pm[i]=logP #array with score of each peptide of the human test dataset
end

#ROC analysis
r=roc(Pm,Ph) #computes statistics for evaluation of the performance of a two-class classifier.
AUC[k]=auc(r) #calculates Area Under the Curve
push!(AUC_output, AUC) #add each new AUC to the array
end
writedlm("newfilename.txt", AUC_output) #save results in txt file

###Find the best λ
(AUC_max, ind_λ) = findmax(AUC_output) #find value and position of maximum AUC
λopt= λ_range[ind_λ]  #find value of λ of given index
#Results
#VH 0.4
#VL 0.1
```

Julia code to adjust MG-score

```
#Code to generate random sequences to correct MG-score

###Packages
using LinearAlgebra
using DataFrames
using DelimitedFiles
using StatsBase

###Arguments
filename= ARGS[1] #MSA file

###Calculate amino acid distribution over the whole alignment
seqmatrix = inputtoascii(filename)
(m,n) = size(seqmatrix)
all = collect(Iterators.flatten(seqmatrix)) #Collect all residues
aminos = deleteat!(all, all .== '-') #Delete gaps
d = proportionmap(aminos)  #Dictionary with frequencies of each amino acid
residues = collect(keys(d)) #Amino acids characters
weights = collect(values(d)) #Amino acids frequencies

###Replace amino acids with new distribution
aminosdict = ['A','C','D','E','F','G','H','I','K','L','M','N','P','Q',
'R','S','T','V','W','Y'] #amino acid dictionary
gapdict = [' ','_','-'] #gap dictionary
```

```
###GAP CORRECTION: Change only positions with amino acid characters
for i in aminosdict
    for k in 1:m
        for l in 1:n
            if i==seqmatrix[k,l] #only residues (gaps are left)
                #Generate random character according to given proportions
                seqmatrix[k,l]= sample(residues, Weights(weights))
            end
        end
    end
end

writedlm("gap_$filename", seqmatrix,',') #save in same format of the MSA file
```