EpitopePredictions vs MHCnuggets

How they work and how they differ.

# Introduction

[RJCB: Well done! This section is done!] We have two programs (MHCnuggets and Epitope Predictions) which both predict how likely a certain epitope is to be shown on the outside of the cell membrane with different haplotypes. This is important in the recognition of one's own body cells. If a cell presents an epitope that is foreign to the immune system, it will be killed, which is beneficial for the body, as the cell could be malfunctioning or infected by a virus. This presentation of epitopes also plays a key role in vaccine development. In order for a vaccine to be effective you want the epitopes to be presented as often and in as many cases as possible, therefore the vaccine has to contain epitopes for all immune types. Thus the immune system can quickly detect it and develop antibodies against the pathogen. Also, this can be used to make sure that this happens on every different haplotype, so people with different immune systems all get immune quickly. However, these programs output some very different results and thus it is unknown if the given predictions are trustworthy or not. In this paper we discuss the differences between these programs, why they are caused, and how this affects the usefulness of the results.

# 

# Hypothesis

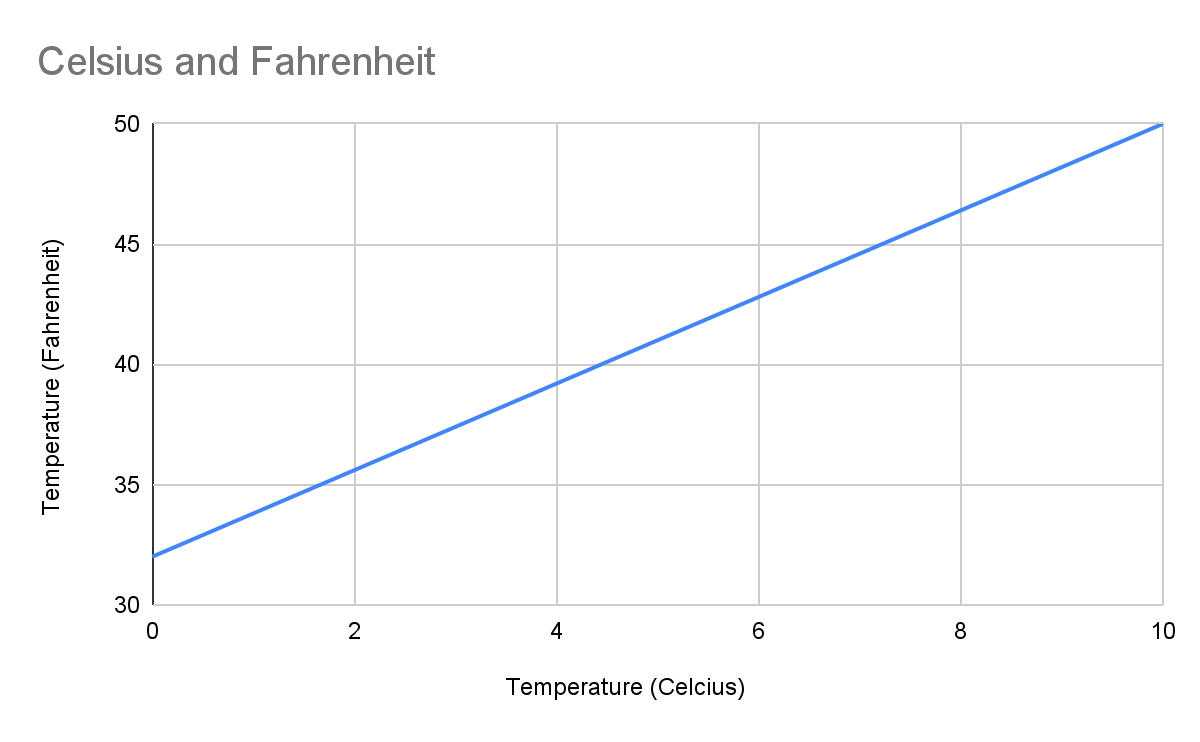
[RJCB: Well done! This section is done! Do read my question] Since both programs are trying to predict the same thing, we expected the results to be equal, or at least very similar. We expected the results to be fairly close together. However, as will be shown in the results section of this paper, both programs output different results, they are not always equal nor are they always fairly close together. We knew this could be caused by the use of different scales for both programs, just like thermometers can give temperature in celsius or fahrenheit and give thus different outputs. In this case the numbers differ, but the temperature stays the same. We thought the same might be happening here. If this is happening, we expect the highest results of EpitopePredictions to also be the highest result from MHCnuggets, and vice versa [RJCB: the conclusion on this must be in the conclusion. I expect a sentence there like ‘We expected the highest results of EpitopePredictions to also be the highest result from MHCnuggets, and vice versa. Based on our findings we conclude that ... [this is true/false]’].

# Methods

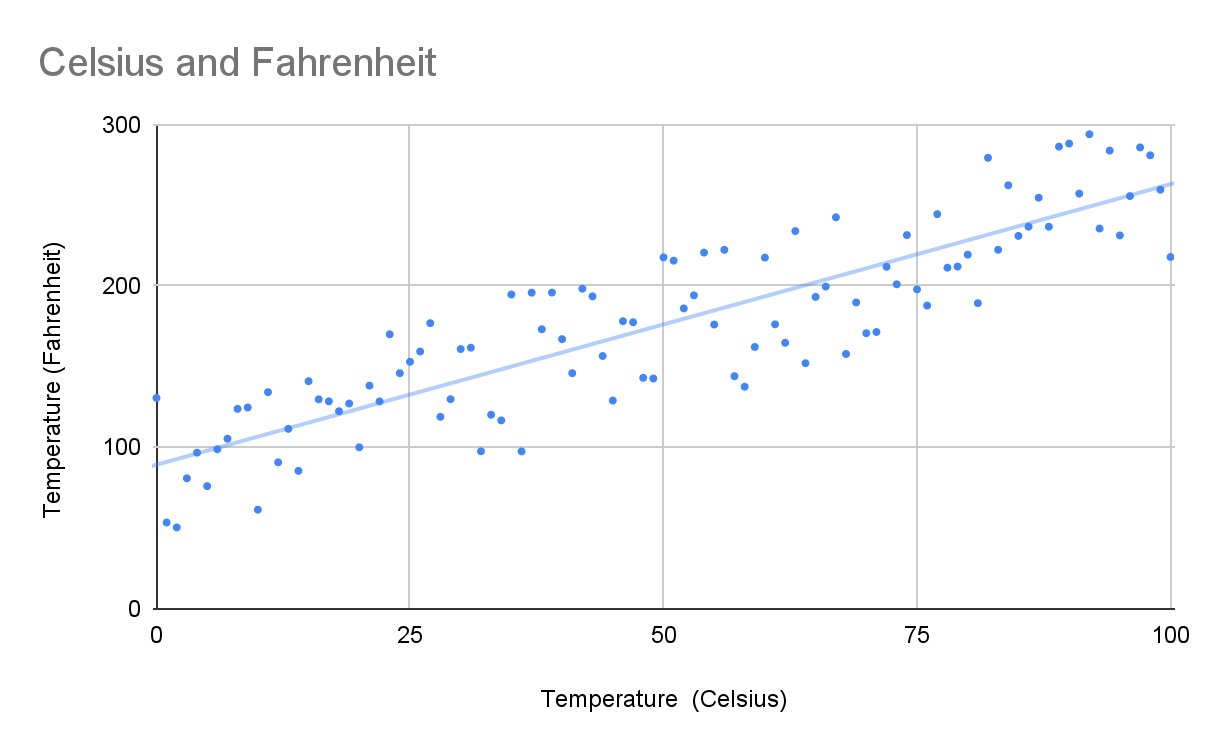
### Correlation

[RJCB: Well done! This section is done!] Here we will be showing our methods for determining whether or not there is a correlation between the predicted results from both programs. We demonstrate this using some examples.

In this table we have the temperature in two different units: Celsius and Fahrenheit. If we were to plot this table, we would get the graph below. The x-axis shows the temperature in Celsius and the y-axis shows the temperature in Fahrenheit. In this example it is easy to see the correlation: as we increase the value of the x-axis, the y-axis follows. While such a correlation is easy to spot, it becomes more difficult when there is noise involved. For this example we expect a trendline with the following formula: Temperature (**°**F) = (Temperature(**°**C) × 9/5) + 32. If we look at the graph we see it starts at 32, which results in the +32. The slope of the graph is equal to , which corresponds to the expected slope.



The example below is exactly the same as the previous example, but we added some noise to the measurements. Though, at first glance, it might not seem like there is a clear correlation between the two, a trendline shows that this assumption is false. We still have the same correlation between the temperature in Celsius and the temperature in Fahrenheit. Here, the trendline has an equation of . This slope (1.7391) is very similar to the expected slope of

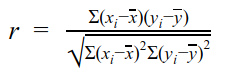


For our last example we have a graph with random data. Here we see there is no correlation: as we increase the value on the x-axis, the y-axis stays the same, this can be seen using the trendline, which has an equation of . This slope is very close to 0, indicating there is no correlation.

We also have a way to quantify this correlation, using the Pearson Correlation Coefficient.

### ChartPearson Correlation

Using the Pearson Correlation Coefficient we are able to find a correlation between the results of MHCnuggets and EpitopePredictions for each different haplotype. See the formula down below.



= correlation coefficient

= values of the x-variable in a sample

= mean of the values of the x-variable

= values of the y-variable in a sample

= mean of the values of the y-variable

The Pearson correlation always has an output ranging from -1 to 1. A value of -1 meaning a fully negative correlation, for example an equation with a slope -1, -0,342 or -(3,5\*1034). The opposite is true for a slope of 1, this means a fully positive correlation, for example an equation with a slope of 1, 0,342 or 3,5\*1034.

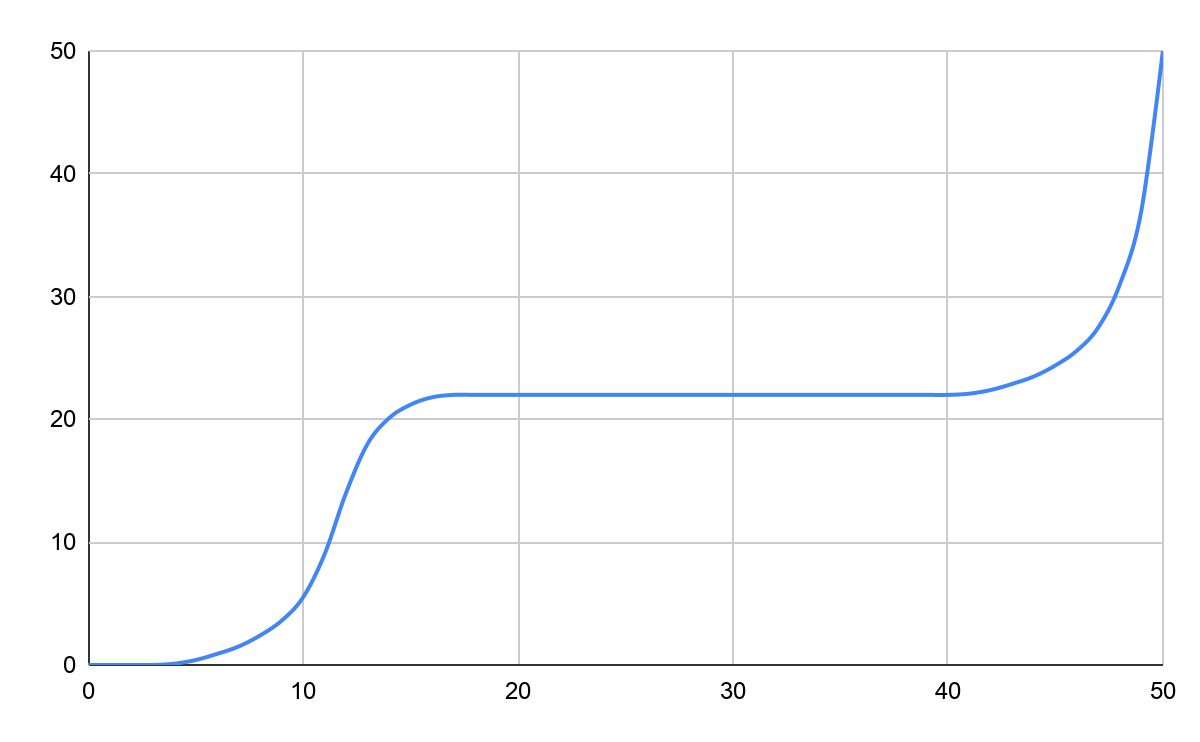
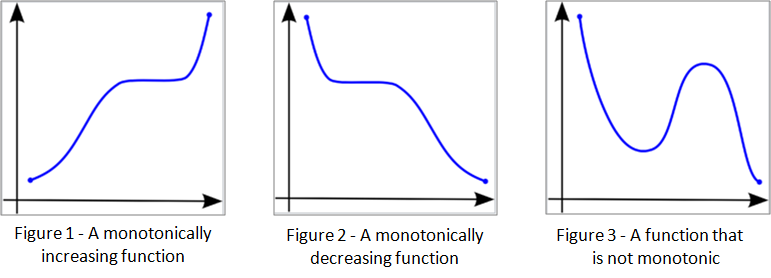
If we were to calculate the r values of above graphs we would get the following results (here we used the values of celsius from 0 to 100 instead of 0 to 10 to have more data points):

* Celsius and Fahrenheit: 1
* Celsius and Fahrenheit with noise: 0.8744
* Noise: 0.0227

As we can see here the values from the first graph have a perfect positive correlation, the slope between 2 values is the same for every value. The second graph, with noise, has a lower correlation due to the many fluctuations in it’s slope. The third graph that is supposed to be random has a tiny positive correlation, this is probably due to the graph being not completely random and due to noise as the trendline also has a slight positive slope (0,0036).

### Spearman Correlation

[RJCB: Well done! This section is done!] While it seems as if this coefficient could provide solid evidence as to whether there is a correlation or not, the Pearson correlation coefficient has some drawbacks. To illustrate this we use the following graph:

It is clear there is a correlation between the variables, but this correlation is non-linear. If we were to input these values into the Pearson formula, we get a correlation of 0.843, which is still considered quite strong, but the Pearson Correlation Coefficient is not meant to be used on a non-linear graph like this one. To work around this problem, we will also be testing if there is a non-linear correlation. For this we would need a coefficient to determine correlation between monotonic, or monotone, sets of data. Monotonic meaning they both tend to go up, or down, together.

This is where the Spearman Correlation comes into play. Using the following formula, we are able to determine how strong a monotonic increase or decrease is between the two datasets, if there is any.



*rs* = correlation coefficient

*N* = the number of data points

*Di* =

When we input the same graph in the Spearman formula, we get a correlation of 0.946. Which is very close to one, meaning there is a very strong correlation between the variables in the graph. If this graph were less perfect, and predictable, the Pearson Correlation could indicate there is no correlation, or a very weak one, while there may still be a monotonic relation present. That is why we will also be testing every haplotype to see the Spearman correlation, to determine if there is a monotonic relationship.

And if we were to use the Spearman correlation on the same values as we did with Pearson’s we get the following results:

* Celsius and Fahrenheit: 1
* Celsius and Fahrenheit with noise: 0.8766
* Noise: 0.0310

As can be seen the values are fairly similar, meaning that Spearman’s correlation can still be used reliably on linear graphs.

But this is not where it ends. As there are non-linear graphs that are not monotonic, like a parabola. For this type of correlation we will be plotting the haplotypes as well, to see if there is a different type of correlation visible.

# Biology

[RJCB: Well done! This section is done!] Before we dive into the two models and the results it is first necessary to understand what kind of values these models actually predict and what those values mean in the physical world. Therefore we will dive into the biological side of EP and MCHn.

## How are antigens presented to the immune system?

[RJCB: Well done! This section is done!] The immune system determines whether a cell in the body is invasive, like a bacteria, and should thus be killed, or a body’s own cell, like a blood cell or a liver cell.

This is usually accomplished by checking whether the antigen that is presented on a MHC-I (Major Histocompatibility Complex) or a MHC-II molecule. These two molecules have nearly the same name, but differ a fair amount. MHC-I molecules are found on the surfaces of all nucleated cells, cells with a nucleus. At that place they present epitopes found inside the cell itself. MHC-II molecules are only found on the surfaces of antigen presenting cells (APCs) like macrophages or phagocytes. Thus MHC-I is used for the detection of infected cells by, for example viruses and MHC-II is used for presenting the antigens of a virus or bacterium to other cells in the immune system, for example, T helper cells or B cells.[4]

For viruses this can be determined by verifying the antigens on the virus’ viral envelope or when a certain cell is infected and is showing a foreign antigen on one of it’s MHC-I proteins. The viral envelope is a defensive shell to protect the DNA inside it.[14] When, for example, a Tc cell detects one of those antigens as presented by MHC-I it will determine whether it is foreign or not, if that is the case the Tc cell will kill the infected cell by disintegrating it’s membrane with proteins. It will also start cloning itself rapidly to be able to more quickly notify other cells necessary in the defense against the virus like B-cells or macrophages.

For bacteria this process is fairly similar, the differences stem from the fact that usually a bacterium’s antigens aren’t directly presented on an infected cell’s membrane. But only on the *Bacterium’s* membrane, thus a Tc cell can’t detect so bacteria can’t be detected by the immune system when they are inside of a cell.[16]

## How does IC50 relate to this?

[RJCB: Well done! This section is done!] IC50 (half maximal inhibitory concentration) is a biological unit that describes how much of a certain substance is required to induce a certain biological processor component by 50%. This could be, for example, a drug, enzyme or cell. The scale often used for expressing IC50 values is molar concentration (mol/L)[15].

In our case the IC50 value corresponds with how well a certain epitope binds to MHC-I. IC50 refers to the amount of a certain substance, so a lower IC50 value corresponds with an epitope being more likely to be presented.

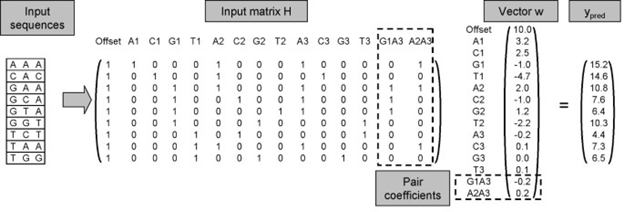
# Epitope Predictions

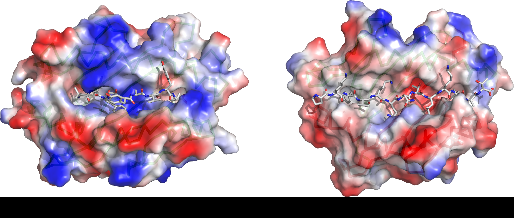
[RJCB: impressive! Except for the one thing I have below, no need to spend tme on this section anymore!] EpitopePredictions is one of the programs used to predict the IC50 value of a randomly generated epitope on a chosen human haplotype, but it can also do this for alleles from mice, chimpanzees and rhesus macaques. To do this it uses the so called stabilized matrix method (SMM), developed by Bjoern Peters and colleagues and uses Peptide:MHC binding energy covariance (PMBEC) as a Bayesian prior, which, in turn was developed by Johannes Textor and colleagues to improve current peptide:MHC binding prediction methods.

**Stabilized matrix method (SMM)**

The stabilized matrix method is one of the methods used for the prediction of MHC binding. Other methods include neural networks, like MHCnuggetsr or NetMHC, which was shown to be the current best prediction program, according to a large-scale benchmark [RJCB: reference here. Also, if MHCnuggets claims it is the best, maybe it is a bit biased :-)]. The program uses XML files for input and output, where each amino acid in an epitope is encoded as a binary vector with a length of 20, because there are 20 amino acids. Every position is set to 0 except for the one coding for the specific amino acid. An example of this could be 00001000000000000000, which, depending on the order in which the amino acids are ranked (in this case alphabetically by their one letter code), would indicate that phenylalanine is present here. These vectors are then stacked up and form a so-called input matrix which is multiplied by a weight matrix, w, resulting in a new matrix, the prediction matrix.

An example for this can be found in the SMM article. Here, instead of amino acid sequences, nucleic acid sequences were used. These can be seen under Input sequences. The letters correspond to each of the four nucleic acids found in DNA, T for Thymine, A for Adenine, C for Cytosine and G for Guanine. The input matrix turns these sequences into a binary vector. For each position, all nucleic acids are set to 0, except for the one present at that position. This nucleic acid is represented by a 1. When we look at the first row of Input matrix H, we see A1 is set to 1, indicating an A, Adenine molecule is present at position one. C1, G1 and T1 are set to 0 which indicates they are not present at position one in the Input sequence. This is done for each position resulting in Input matrix H. This matrix is then multiplied by weight vector w, outputting the ypred, or predictions, matrix.



The weight vector w is derived from the hydrophobicity of each of the amino acids. Hydrophobicity is the physical property of a molecule that is seemingly repelled from a mass of water. An every-day example for this is oil. Oils are hydrophobic, meaning they do not mix with water and are repelled by it. This is also the case for some amino acids. This is where MHC class I is important. The MHC I molecule is built up of α- and β-chains [], which differ between haplotypes. These chains form a groove, in which a peptide can bind. The MHC I groove is closed and because of this, mostly short epitopes can properly bind to it, though research has shown longer epitopes do sometimes bind to MHC I []. These epitopes are mostly between 9 and 11 amino acids long. However, epitopes with different lengths often use alternative binding grooves, which complicates predictions []. Because of this, Epitope Predictions uses epitopes that are 9 amino acids long, because they are most common [3], and to simplify predictions.

The MHC-I molecule is partly hydrophilic, meaning it mixes well with, and is attracted to, water, and partly hydrophobic. This means that a peptide consisting of hydrophobic amino acids is repelled by the hydrophobic cleft in the MHC-I molecule. The distribution of hydrophobic and hydrophilic amino acids and their positions determines how well a certain epitope can bind to MHC-I. The weight vector is chosen in such a way that the predicted values correspond to measured values. This is done with multiple sets of data and then the average weight value of all these tests is used in the final matrix.

The stabilized matrix method also has a built-in method to suppress the effects of noise in input data. A positive scalar is added, which is determined by taking the average of multiple training sets. A scalar can be defined as something which only has a magnitude but no direction. Examples of scalars are temperature and pressure. A certain point in space has a temperature value (magnitude) but no direction, since temperature does not work in a direction. This scalar shifts all optimal entries in w closer to 0.

We use the following example. Say Ypred, the predicted value is equal to 4 and the measured value, ymeas is equal to 5 and the scalar is set to 0. In this case w needs to be adjusted to minimize the difference between these values since we want the predicted results to be as close to the measured results as possible. If we change the value of our scalar to a positive value that is not equal to 0, this shifts all optimal entries for w (the entries that minimize difference between ypred and ymeas) closer to 0. When values of w are lower, this means noise is multiplied by a smaller number and is thus reduced. This process is called regularization.

**Peptide:MHC binding energy covariance (PMBEC)**

[RJCB: Excellent! This section is done!] PMBEC is a new amino acid similarity matrix and was created using experimental data. Like many other matrices used for the same or similar purposes, PMBEC takes into account well-known physiochemical (chemistry of organs and tissues of the body) properties of amino acid residues. Amino acid residues are two or more amino acids linked together, a peptide, where the elements of water have been removed. PMBEC differs in cases where a charged amino acid is changed for an oppositely charged amino acid. An example for this is substitution of Glutamic Acid with Arginine. Glutamic Acid has a negatively charged chain in its molecular structure, while Arginine has a positively charged chain.

EpitopePrediction uses PMBEC as a Bayesian prior for SMM. A Bayesian prior, short for prior probability distribution, refers to the Bayesian inference, which is a method of statistical inference in which Bayes’ theorem is used. Bayes’ theorem “describes the probability of anevent, based on prior knowledge of conditions that might be related to the event.” This is used to update the probability for a hypothesis as more information is available. Meaning the prediction method can be refined by benchmarking it against measured data.

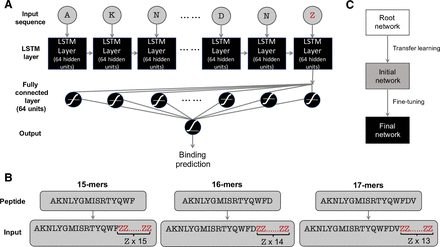
The researchers compared results from PMBEC to 10 other prediction matrices, one of which was BLOSUM50. PMBEC and BLOSUM50 had a Pearson Correlation of 0.64 between each other, while BLOSUM50 had correlation of more than 0.93 with seven of the other matrices, indicating PMBEC is different from most algorithms.

The article about PMBEC describes how SMMPMBEC has significantly better prediction accuracy than SMM, especially when the amount of data was smaller. They also found the average difference in performance between SMMPMBEC and NetMHC, which is currently the best method for prediction peptide binding to MHC-I is not statistically significant. Meaning they are almost identical in results.

The main benefit of SMMPMBEC over NetMHC is that SMMPMBEC is easier to understand and way simpler, while performance is about equal.

# MHCnuggets

[RJCB: This section is done, I am more interested in the conclusion today :-)] MHCnuggets is another program which predicts an IC50-value. MHCnuggets is developed to to work around the limitations other predictors had, by using a neural network that predicts the peptide-MHC bindings. It uses a method called LSTM(long short-term memory) which is good at processing peptides which can be of any length. The neural network got trained with a method called transfer learning, which in this case means the network uses the data(binding affinity, IC50-value) from other alleles to predict the IC50-value of an unknown (to the program) allele.



visualisation of the neural network

# 

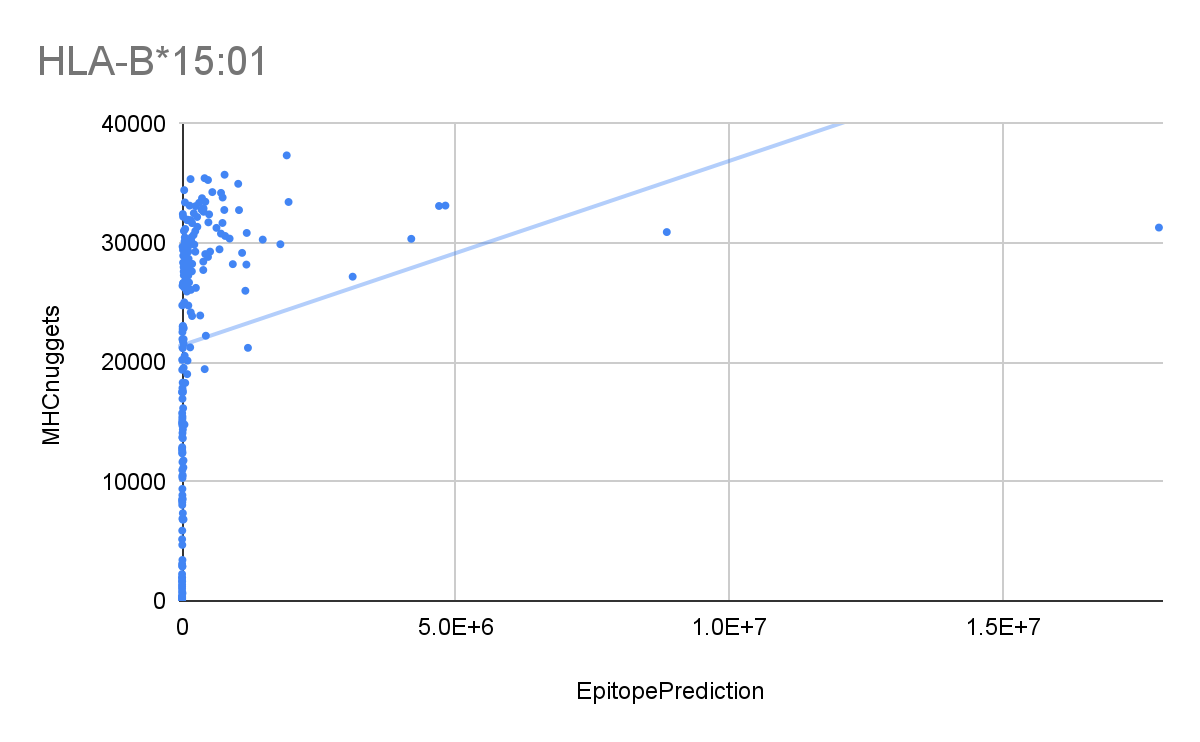
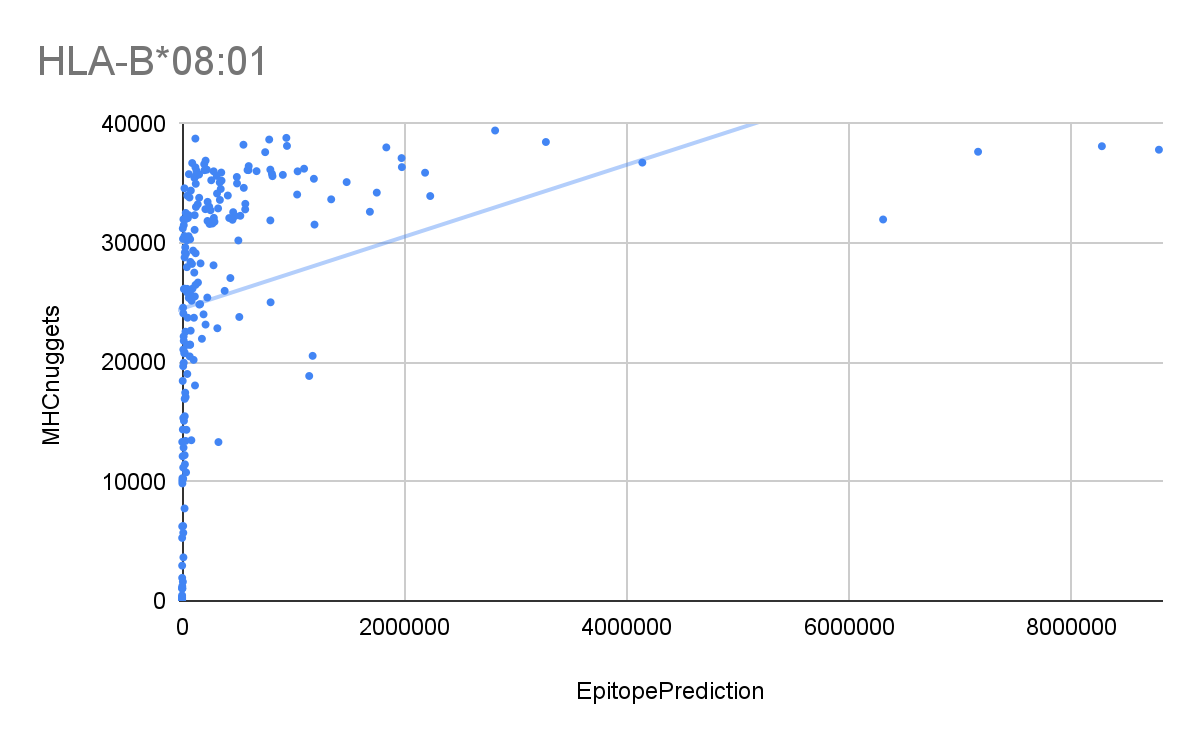
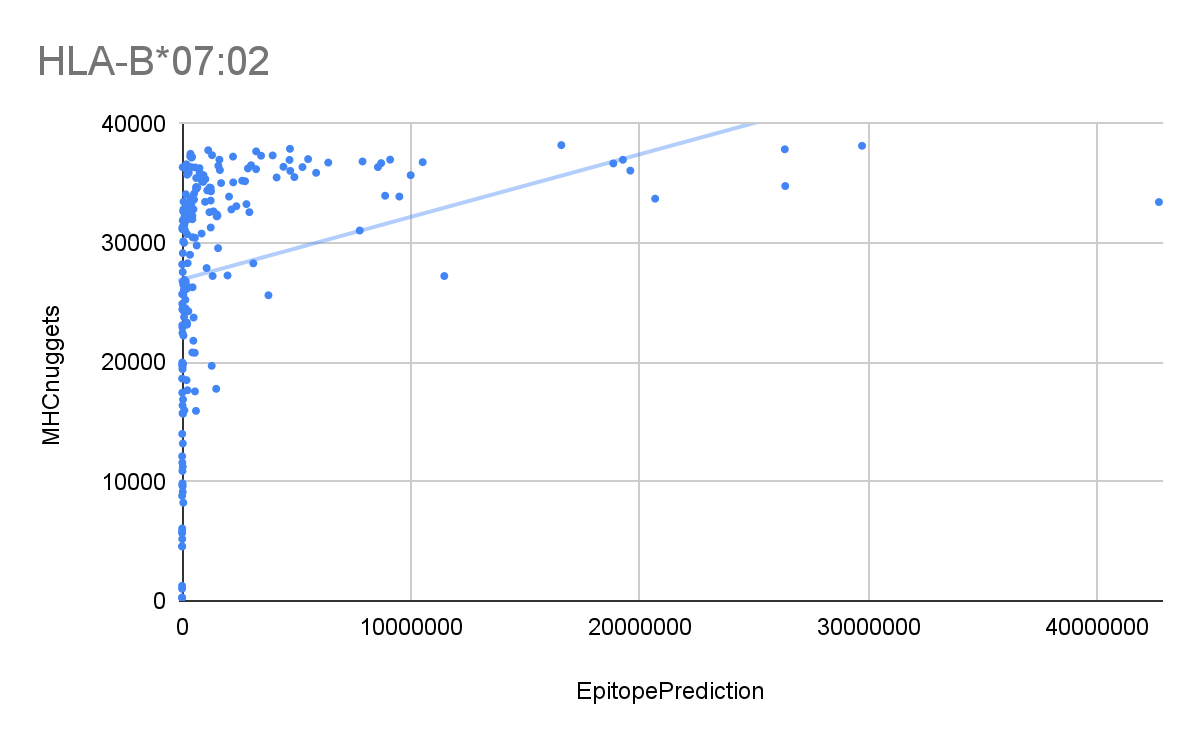
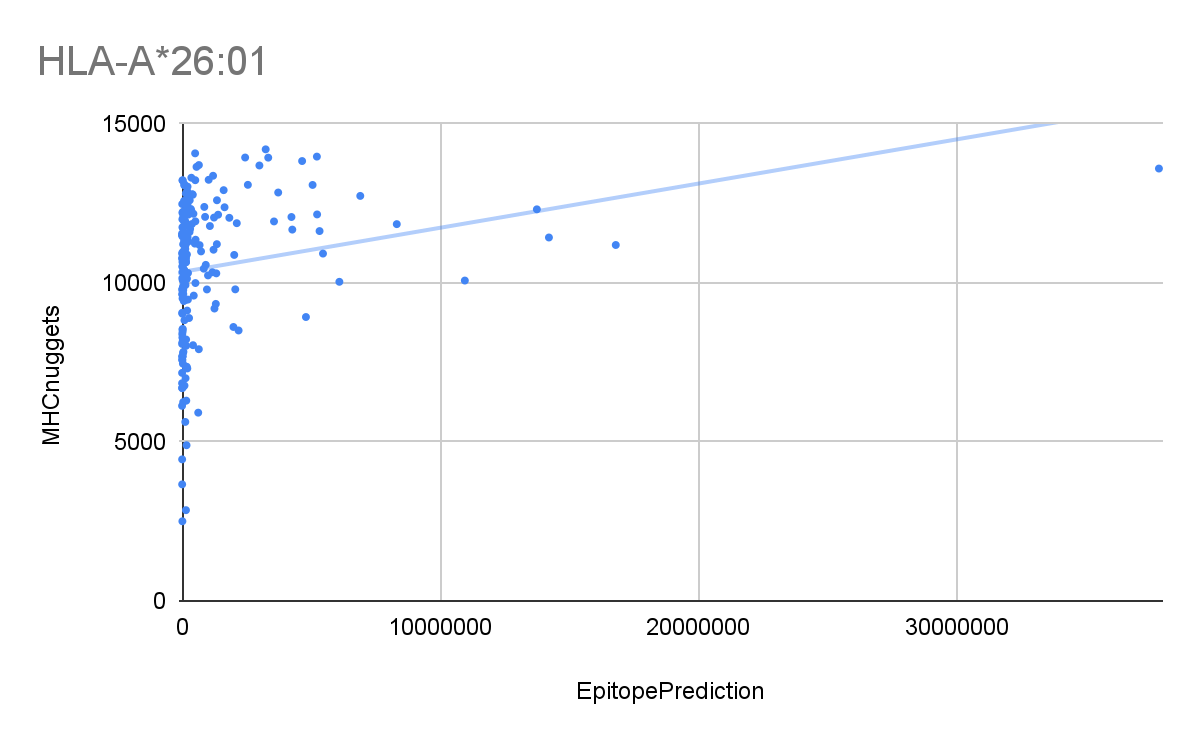
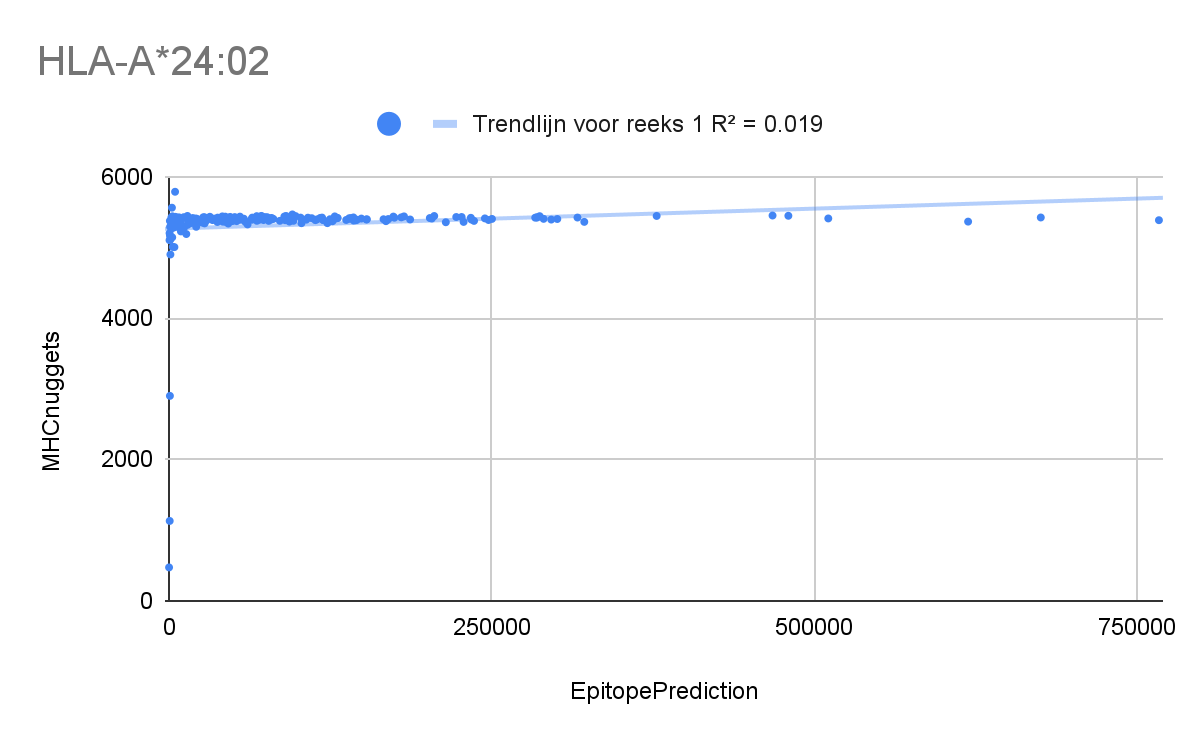
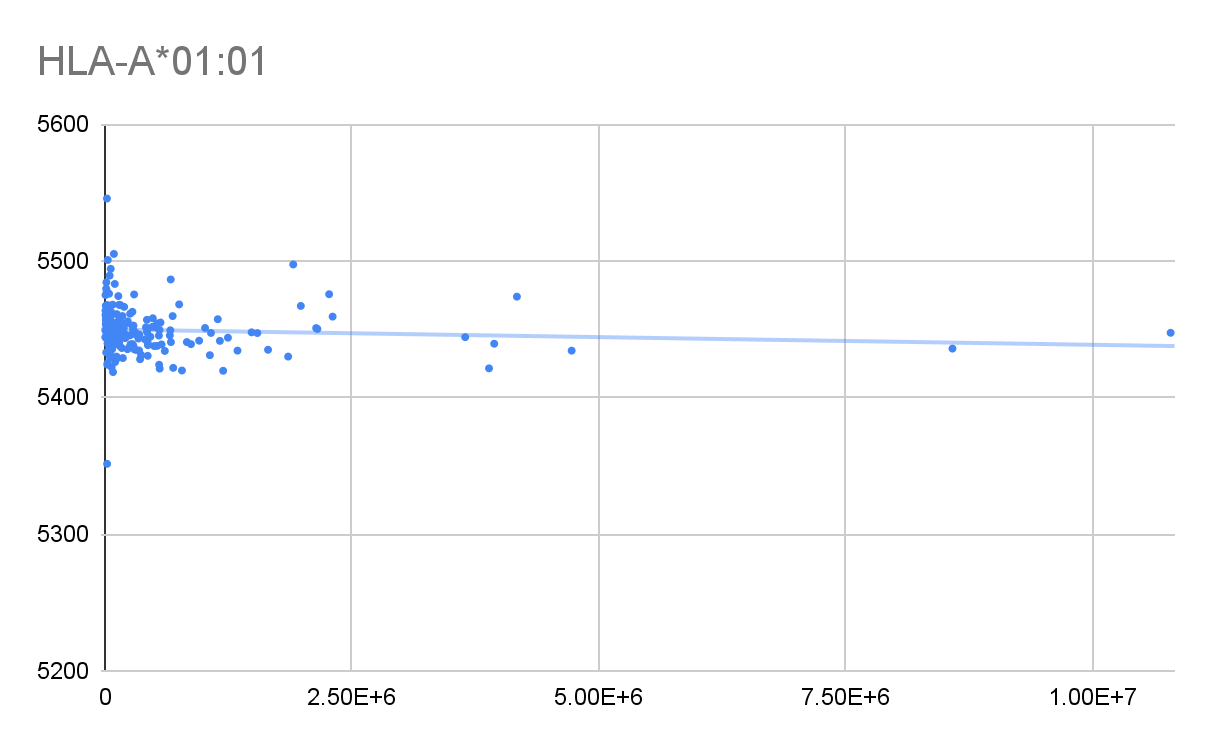
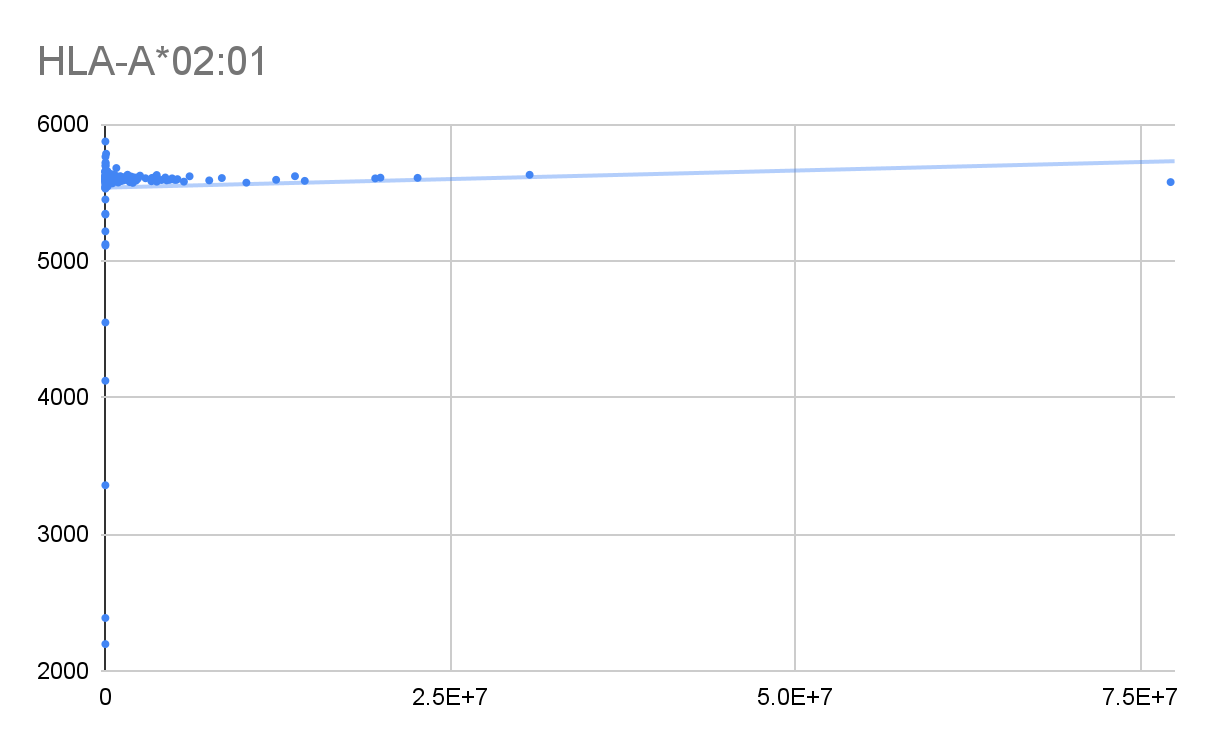
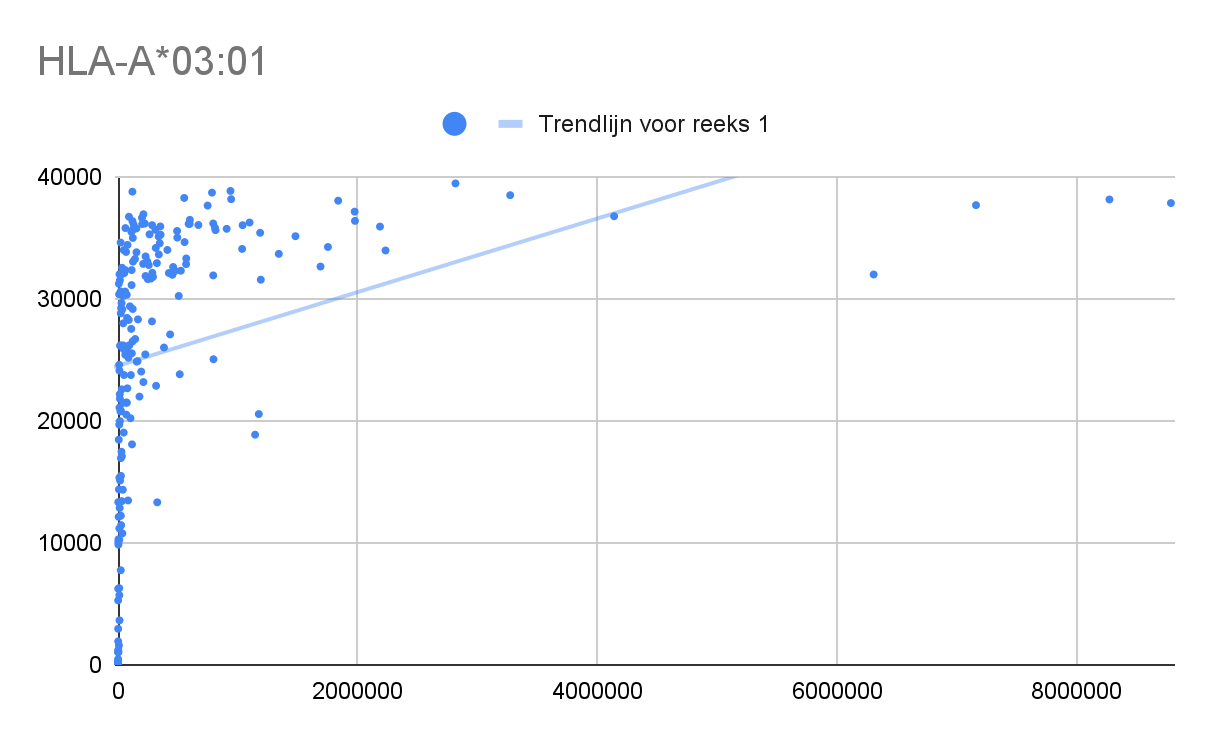
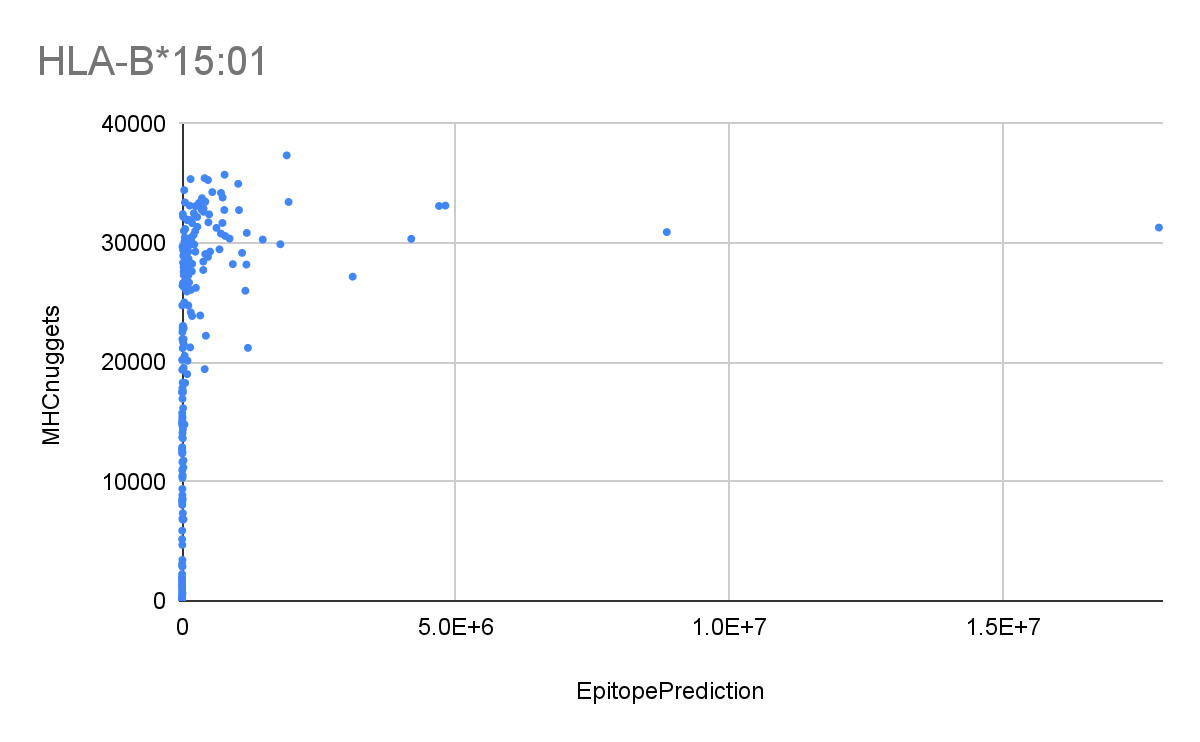
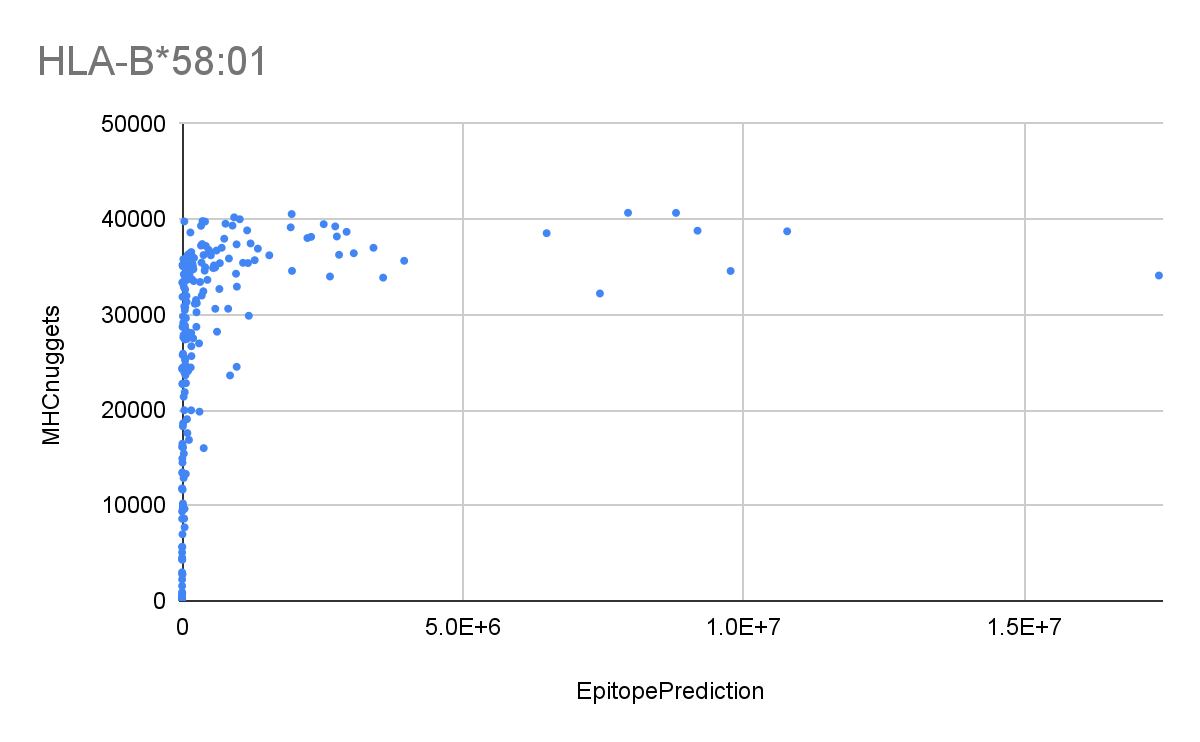
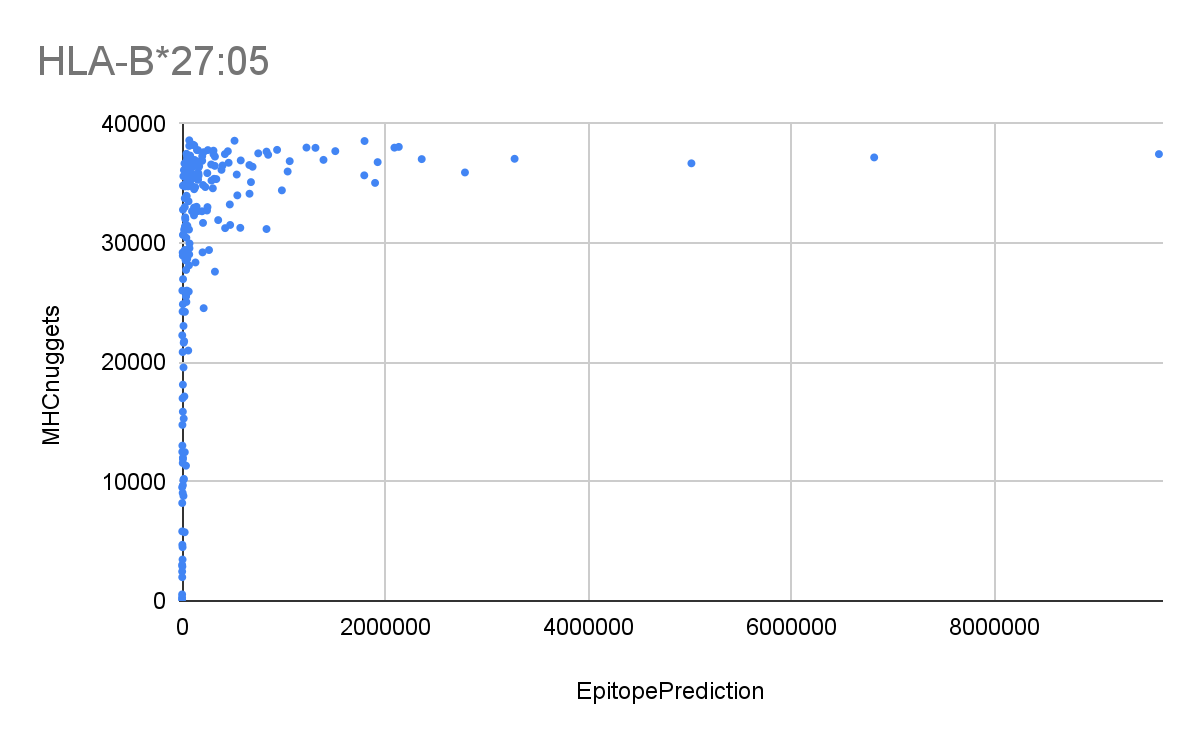
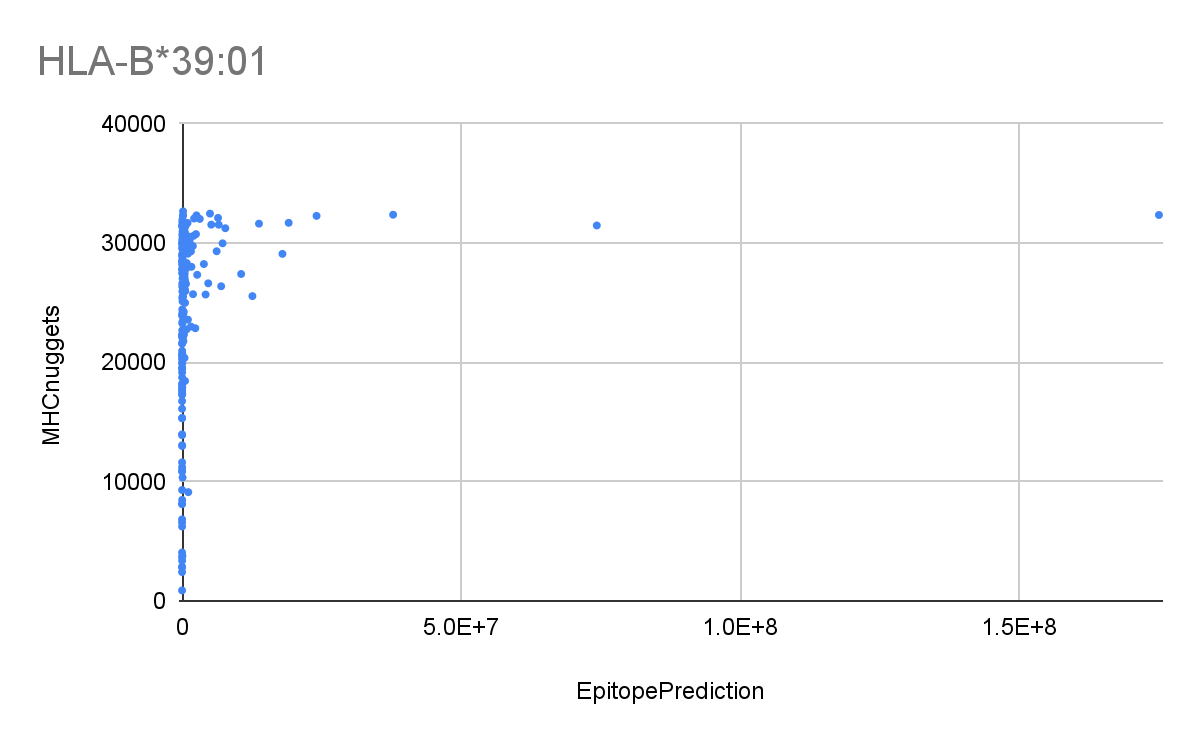
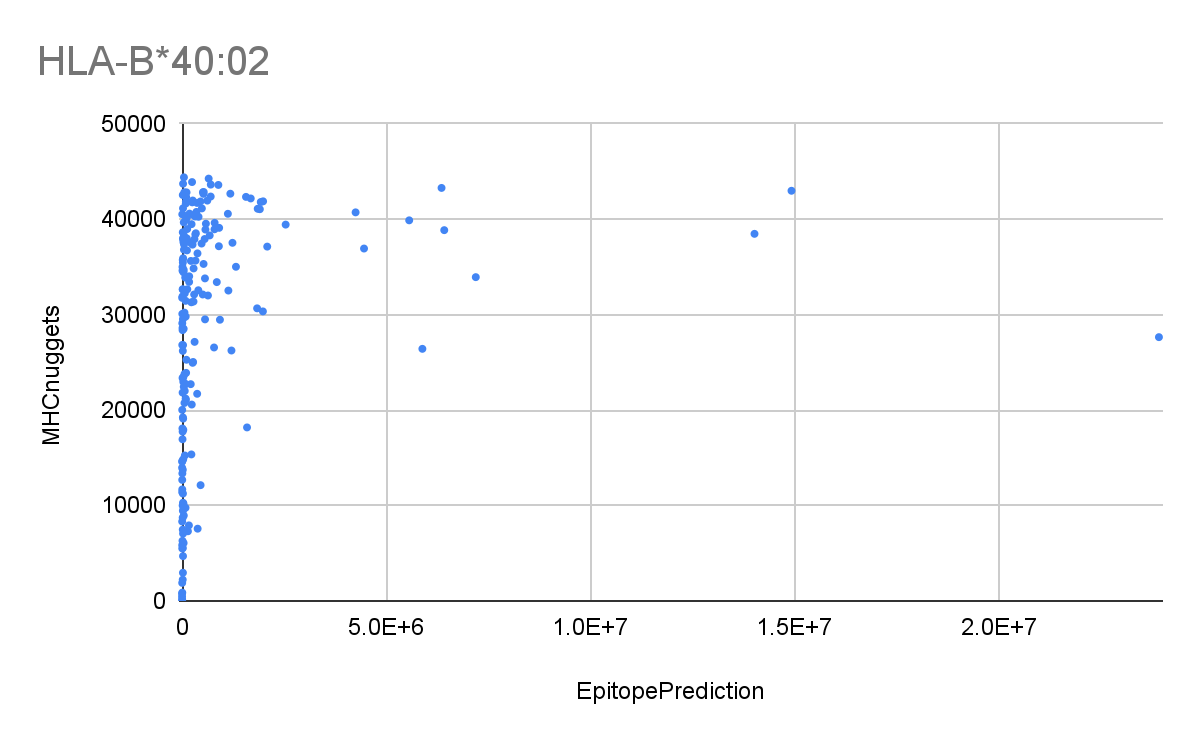
# R-code

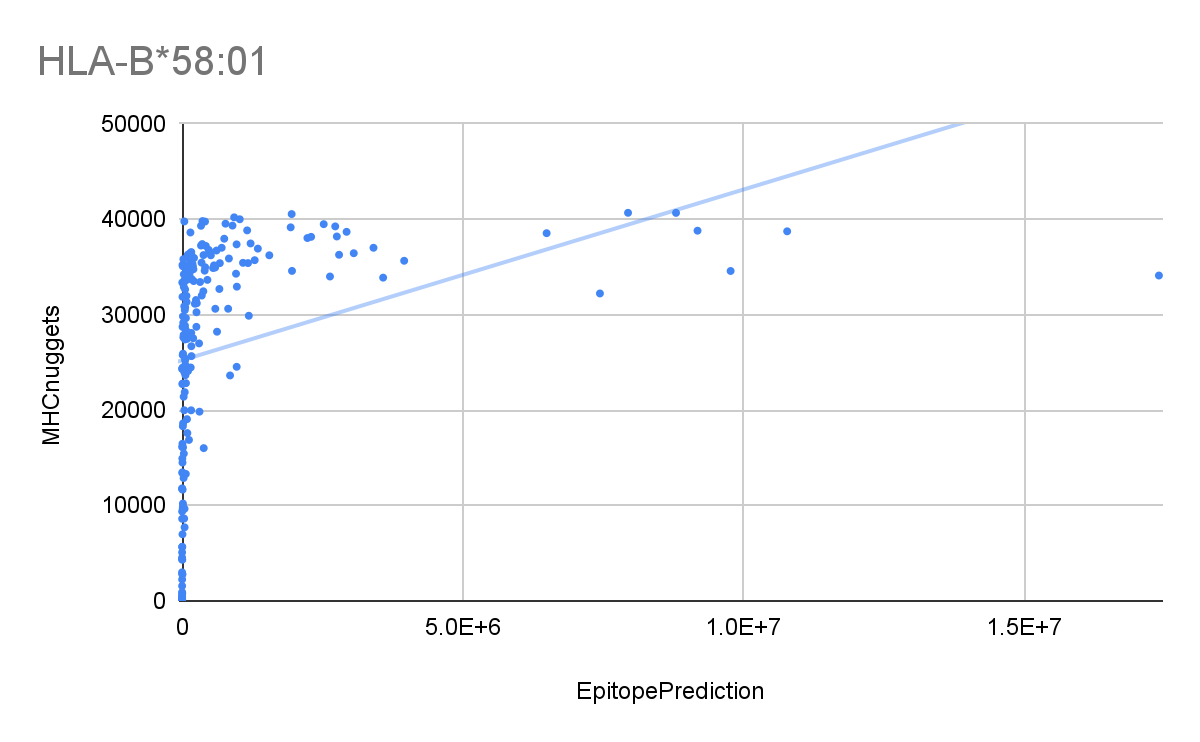
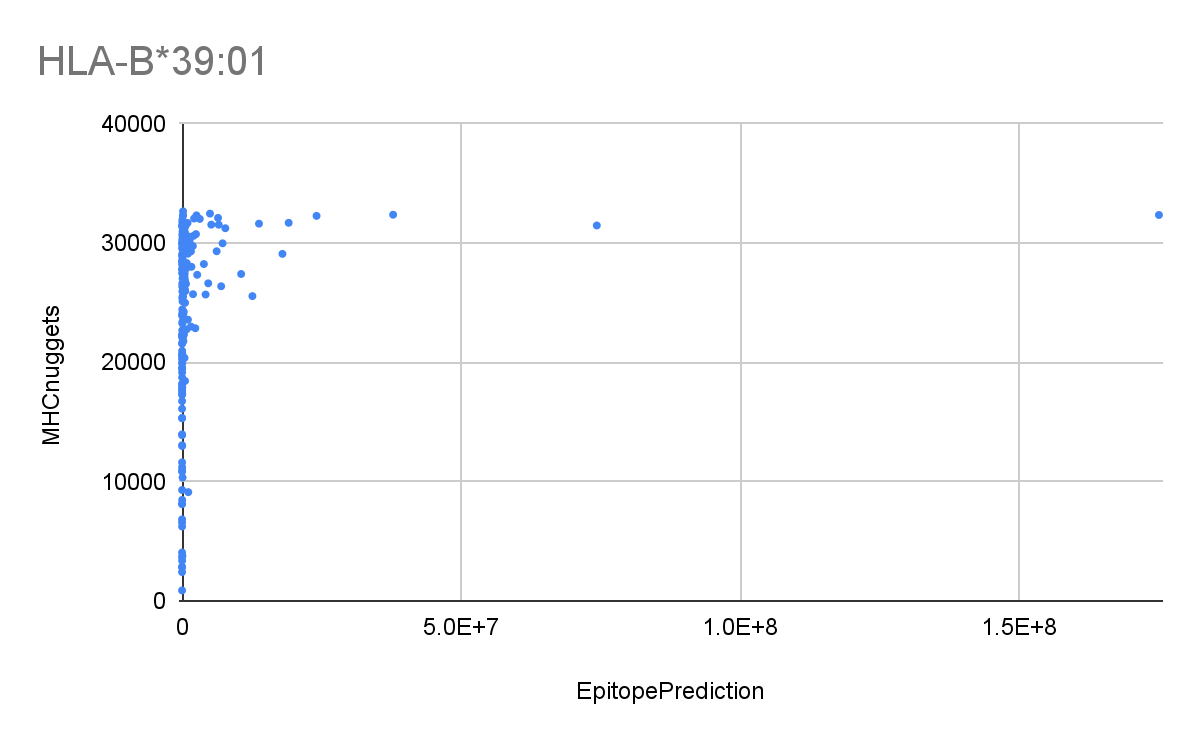
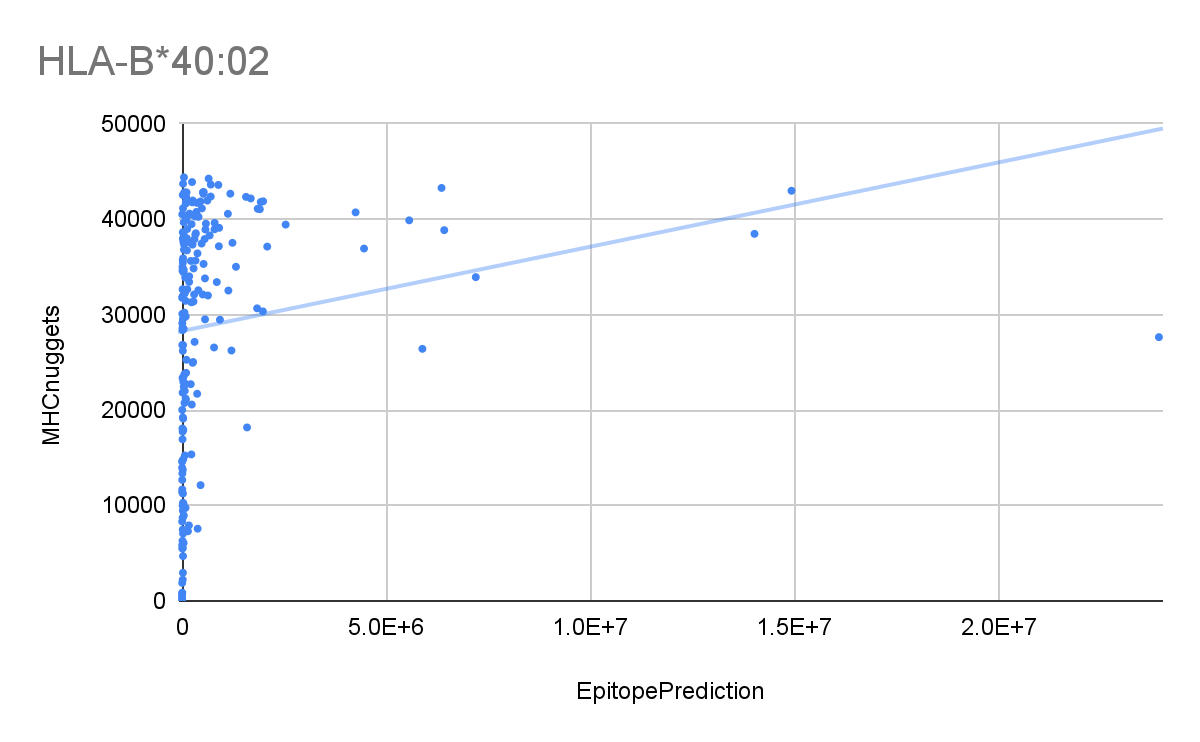
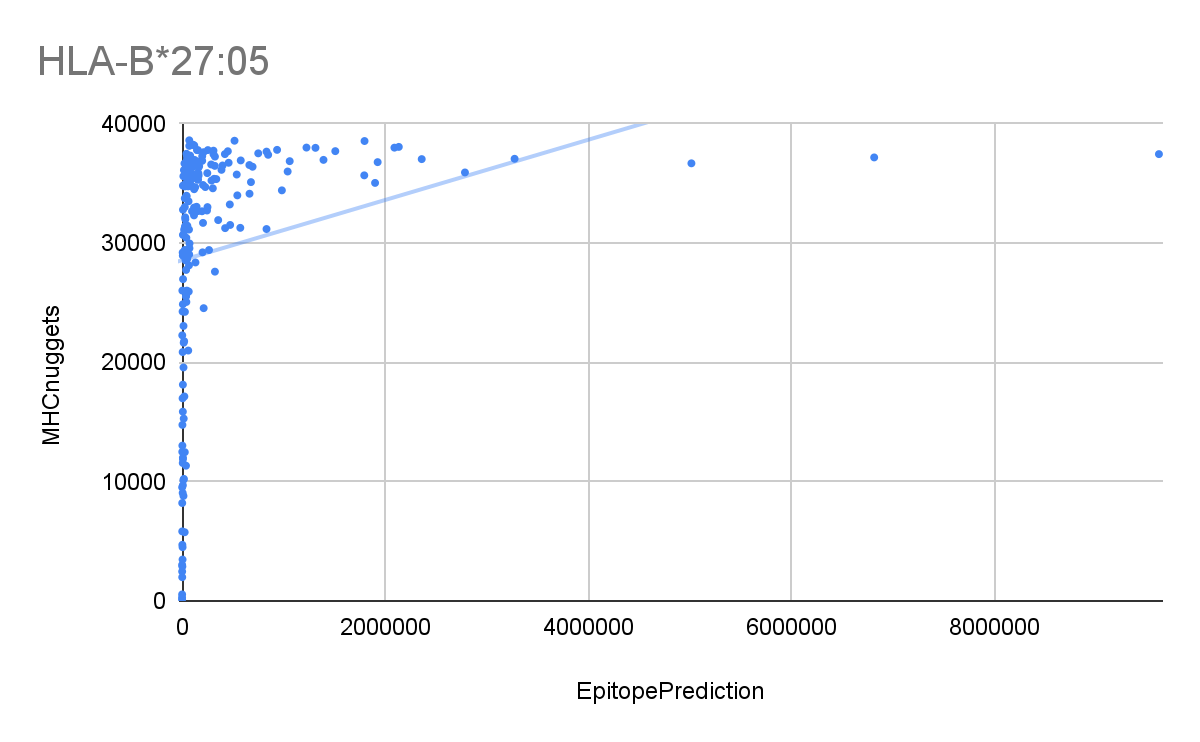
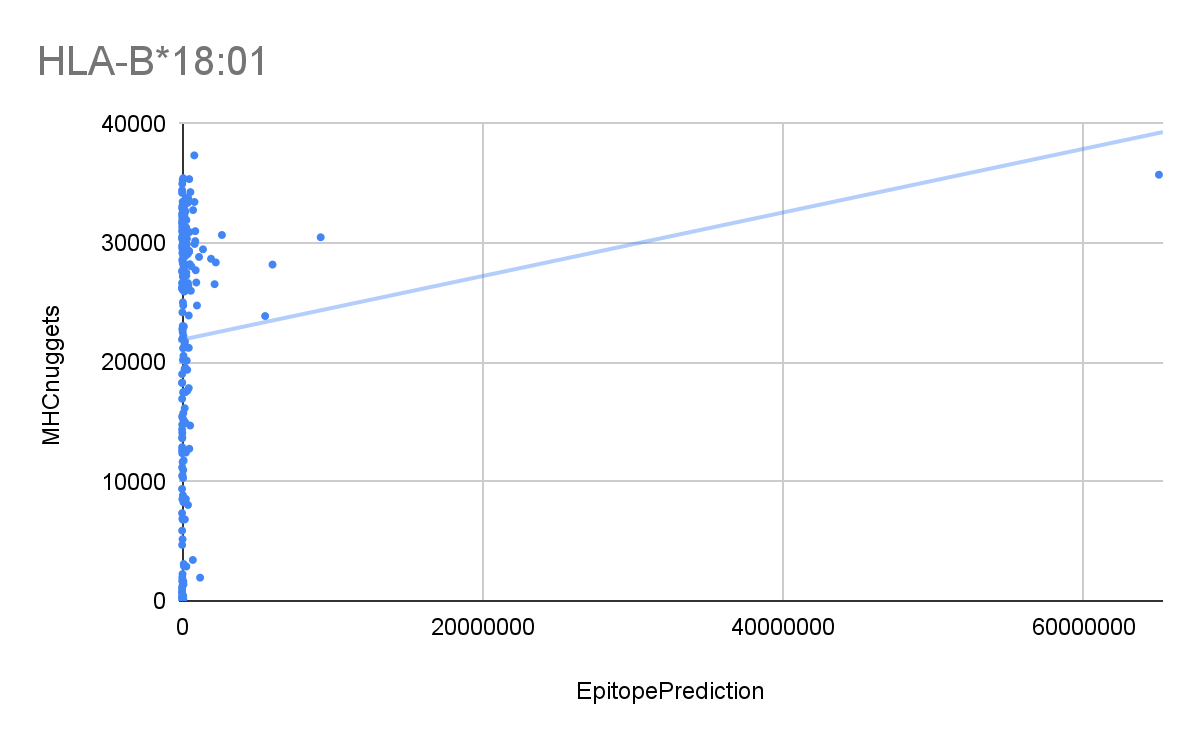
[RJCB: This section is done, I am more interested in the conclusion today :-)] For the generation of the misbehaving IC50 values that were the motivation for this research 2 R scripts were used. Those scripts can be found here: <https://github.com/richelbilderbeek/ep_vs_mhcn>. They roughly work as follows: the first script, <https://github.com/richelbilderbeek/ep_vs_mhcn/blob/master/create_dataset.R>, is used to generate random peptides (line 20 to 31) and after that the two programs MHCnuggets and EpitopePrediction are used to predict the ic50 values of the randomly generated peptides (line 33 to 41), these are stored in the file “ep\_vs\_mhcn.csv” (line 43).

# 

# Results

|  |  |  |
| --- | --- | --- |
| **Haplotype** | **Pearson Correlation** | **Spearman Correlation** |
| **HLA-A\*01:01** | -0.074 | -0.251 |
| **HLA-A\*02:01** | 0.042 | 0.043 |
| **HLA-A\*03:01** | 0.055 | 0.08 |
| **HLA-A\*24:02** | 0.138 | 0.469 |
| **HLA-A\*26:01** | 0.218 | 0.434 |
| **HLA-B\*07:02** | 0.309 | 0.734 |
| **HLA-B\*08:01** | 0.329 | 0.762 |
| **HLA-B\*15:01** | 0.227 | 0.812 |
| **HLA-B\*18:01** | 0.119 | 0.221 |
| **HLA-B\*27:05** | 0.252 | 0.73 |
| **HLA-B\*39:01** | 0.156 | 0.623 |
| **HLA-B\*40:02** | 0.171 | 0.556 |
| **HLA-B\*58:01** | 0.31 | 0.785 |





Here we have the correlation-values as calculated using both Pearson’s and Spearman’s formulas. The Pearson Correlation gave a value between -0.074 and 0.309 and the Spearman Correlation ranged from -0.251 to 0.812. The first three values are very close together [As haplotypes are unrelated to one another, this statement is as useful as saying ‘The first three colors of the rainbow have names (‘red’, ‘yellow’, ‘green’) of similar lengths’].

A notable exception is that the first haplotype has a slight negative correlation, contrary to all the others which have a slight or moderate positive correlation. This means that on average the graph has a slightly negative slope [RJCB: what does this mean for the conclusion?].

Another interesting observation is that the values calculated by Pearson’s coefficient are consequently lower than those calculated by Spearman’s coefficient. However, in haplotype HLA-A\*02:01 this is not the case, but the correlation for that haplotype is so low that this could easily be due to statistical errors, like a rounding error [RJCB: you really think that is likely? If yes, is there evidence to think so?].This indicates that the haplotypes are in a monotonic relationship, due to monotonic relationships always having linear parts that will be detected by Pearson’s coefficient. Spearman’s coefficient however, will detect the whole monotonic relationship instead of only it’s linear parts, therefore resulting in higher values for Spearman’s coefficient, but still moderate values for Pearson’s [RJCB: what does this mean for the conclusion?].

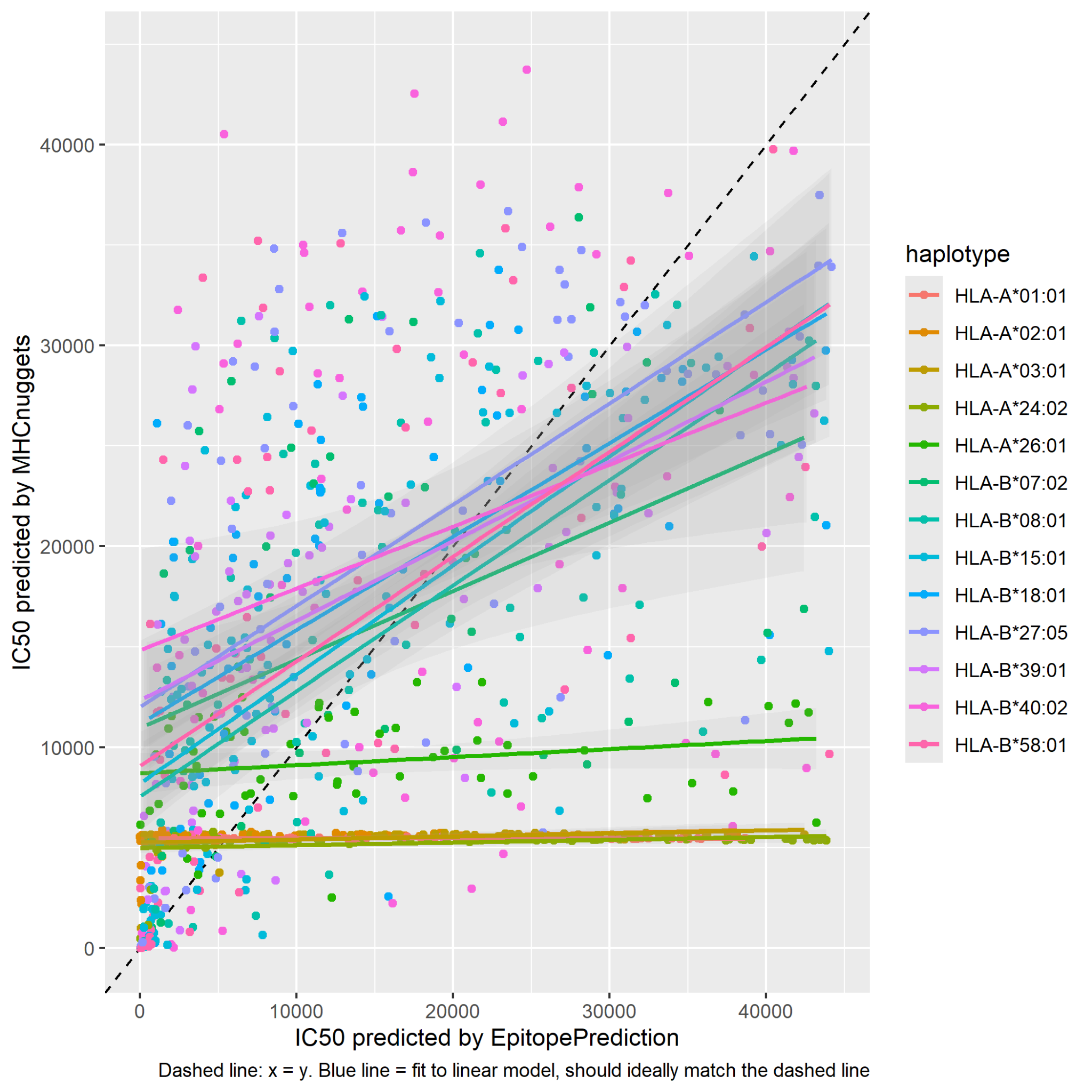
# 

# Discussion

These results probably mean that a monotonic correlation is present in *some* haplotypes, the ones with higher levels of correlation calculated by Spearman’s coefficient. That could also mean that the results for those haplotypes calculated by both programs are more trustworthy because the programs more or less agree with one another. But it is much harder to make such claims for the haplotypes with lower levels of correlation, for those the programs do not agree as much. Thus those results are more likely to be untrustworthy [RJCB: I enjoy such clear words that draw a clear conclusion! I would enjoy a table with haplotype names and verdict, e.g ‘trustworthy’, ‘untrustworthy’. Using a self-made-up scale for trustworthiness would be amazing].

As can be seen in the results (figure x), there appears to be a slight correlation according to Pearson’s coefficient (an average r-value of 0,1732). However, this correlation is very low, so it cannot be fully trusted. But due to the value being above zero it does indicate a slightly positive linear graph. [RJCB: with such a low PCC, wouldn’t you say the programs say something close-to-completely different? Which PCC is the threshold?]

Interestingly, the average correlation calculated by Spearman’s coefficient is much higher, namely 0,4614. This result is still fairly low, but a lot higher than the average calculated by Pearson’s coefficient [RJCB: in principle, the PCC and SC values are unrelated, i.e. they have different units, as they measure different things. A high SC allows a trustworthy PCC, but comparing these values makes no sense. Either don’t compare are enlighten me that one can compare PCC and SC], because of that his result is more trustworthy. It indicates a fairly monotonic graph with some linear parts.

It is also striking [RJCB: Well observed, but this is not striking: if there is a correlation found by method A it is not striking to see that a correlation is found by methods B. Striking would be if there are contradicting results] to see that the correlations of different haplotypes appear to correlate with how close a trendline’s slope in the right graph is to y=x (the dashed line in figure x). This was to be expected as that describes some form of correlation too. This same effect is also visible in other calculations done on this data, for example [the relative values](https://github.com/richelbilderbeek/ep_vs_mhcn/blob/master/ep_vs_mhcn_perc.png?raw=true) and [the sorted values](https://github.com/richelbilderbeek/meesterproef_joshua/blob/master/ep_vs_mhcn_sort.png?raw=true). [RJCB: either remove that figure or do something useful to it. AFAICS, it adds no value to the PCC and SC story]

# Conclusions

# Epilogue

# Acknowledgements

# Definitions

Epitope: the part of an antigen molecule to which an antibody attaches itself. An epitope is a specific protein on the surface of an antigen and is used for recognition of cells.

IC50: a quantitative measure that indicates how much of a particular inhibitory substance (e.g. drug) is needed to inhibit, in vitro, a given biological process or biological component by 50%. The biological component could be an enzyme, cell, cell receptor or microorganism. For our research this means a lower IC50 corresponds to an epitope being more likely to be presented.

Haplotype: a group of genes within an organism that was inherited together from a single parent. A different haplotype can correspond to a different MHC I-complex, which in itself means that different epitopes are more or less likely to be presented to the immune system.

# 

# References

[1] Peters, Bjoern, and Alessandro Sette. "Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method." *BMC bioinformatics* 6.1 (2005): 1-9.

[2] Sanchez-Trincado, Jose L., Marta Gomez-Perosanz, and Pedro A. Reche. "Fundamentals and methods for T-and B-cell epitope prediction." *Journal of immunology research* 2017 (2017).

[3] Trolle, Thomas, et al. "The length distribution of class I–restricted T cell epitopes is determined by both peptide supply and MHC allele–specific binding preference." *The Journal of Immunology* 196.4 (2016): 1480-1487.

[4] Janeway, Charles A, and Jr. “The major histocompatibility complex and its functions.” *Immunobiology: The Immune System in Health and Disease. 5th edition.*, U.S. National Library of Medicine (1970).

[5] Research done by Joshua van Waardenberg

[richelbilderbeek/meesterproef\_joshua](https://github.com/richelbilderbeek/meesterproef_joshua)

[6] Initial research done by Richel Bilderbeek

[richelbilderbeek/ep\_vs\_mhcn](https://github.com/richelbilderbeek/ep_vs_mhcn)

[7] Spearman’s vs Pearson’s correlation coefficient

<https://statisticsbyjim.com/basics/spearmans-correlation/>

[8] Various calculators for correlation coefficients

<https://www.socscistatistics.com/>

[9]MHC class I

<https://en.wikipedia.org/wiki/MHC_class_I>

[10] hydrophobicity

<https://en.wikipedia.org/wiki/Hydrophobe>

[11] Epitope Predictions

<https://github.com/jtextor/epitope-prediction>

[12] Stabilized matrix

<https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-6-132>

[13] Interpretation of Pearson’s coefficient

<https://www.questionpro.com/blog/pearson-correlation-coefficient/>

[14] Viral Envelope

<https://en.wikipedia.org/wiki/Viral_envelope>

[15] IC50

<https://en.wikipedia.org/wiki/IC50>

[16] Our biology textbook

<https://www.noordhoff.nl/voortgezet-onderwijs/biologie/nectar>

### 

### Images:

[] Cell membrane

<https://biologydictionary.net/cell-membrane/>

Visualisation of the neural network

<https://cancerimmunolres.aacrjournals.org/content/8/3/396.long>

MHCnuggets

<https://cancerimmunolres.aacrjournals.org/content/8/3/396.long>

LSTM

<https://www.researchgate.net/publication/13853244_Long_Short-term_Memory>

# 

# Notes

Epitope Predictions gebruikt een stabilized matrix, waarbij de hydrofobiciteit van de aminozuren wordt opgeteld. Epitope Predictions is sneller dan MHCnuggets, maar misschien minder betrouwbaar. JTextor

MHCnuggets gebruikt een getraind neuraal netwerk en is langzamer, maar misschien nauwkeuriger.