Comparitive study on prediction power of PSSM, SMM, and ANN

Results Analysis & Visualisation

Contents

1) Introduction

In this results analysis & visualisation, we will compare the results for 3 types of algorithms, for predicting binding motifs in MHC Class I proteins. Performance in this case is measured with PCC (Pearson Correlation Coefficient) - that is, the discrepancy between predicted motifs' binding coefficient, vs the actual binding coefficient.

We also attempted to optimise algorithm parameters, achieve the highest performance for each method. These will also be evaluated in each algorithms' respective section.

For a nice summary of the results, skip to the final 'Comparison' and 'Meta-study' sections.

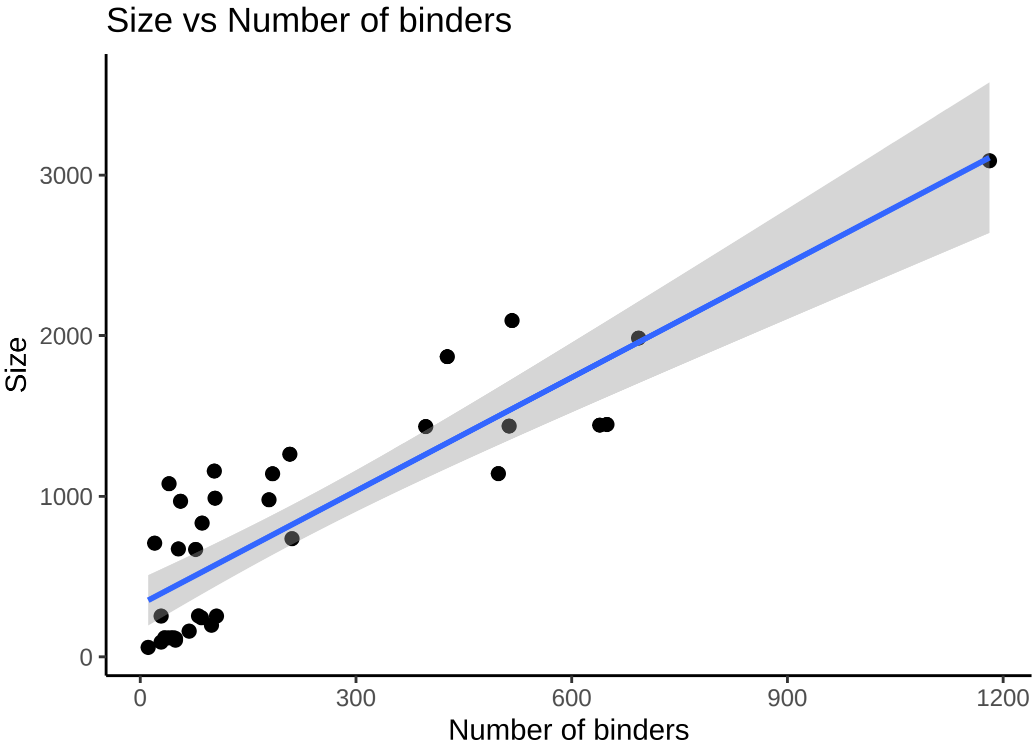
2) Dataset statistics

In addition to the actual protein sequences, an accompanying dataset called `count` was included, which includes protein ID's, along with their respective alleles, and number of binders. There are 35 alleles in total, here the first 6 are displayed.

```
##   Allele Size Number of binders
## 1  A0101 1157               103
## 2  A0201 3089              1181
## 3  A0202 1447               649
## 4  A0203 1443               639
## 5  A0206 1437               513
## 6  A0301 2094               517
```

2.1 No. binders vs size

The number of binders clearly scales linearly with the size of the dataset.

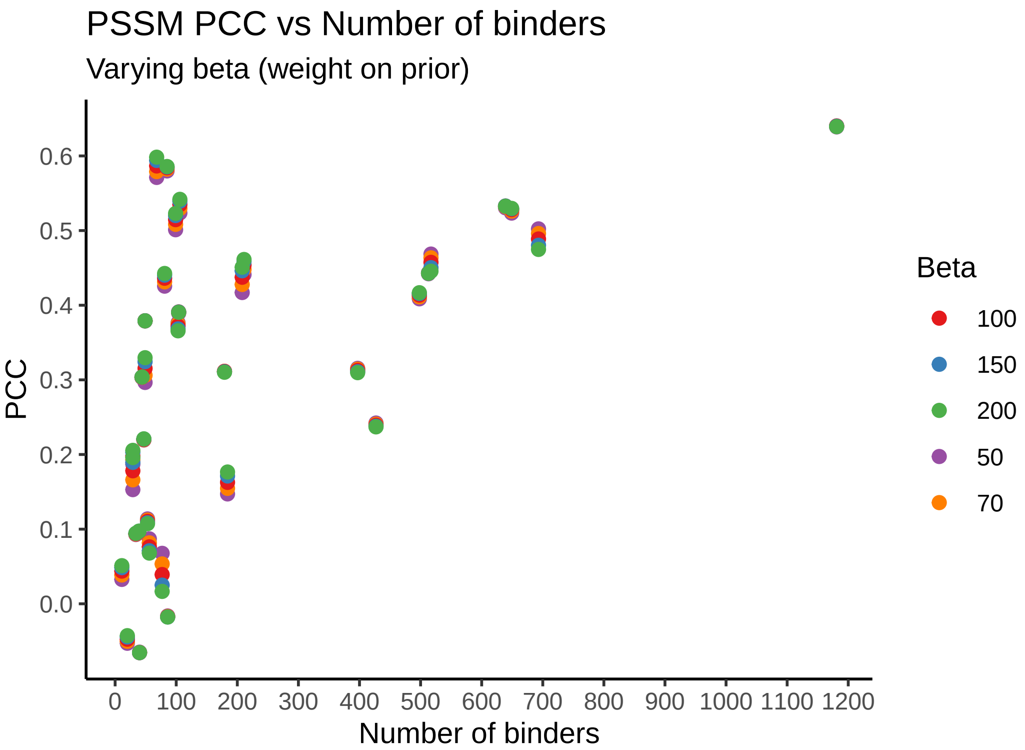


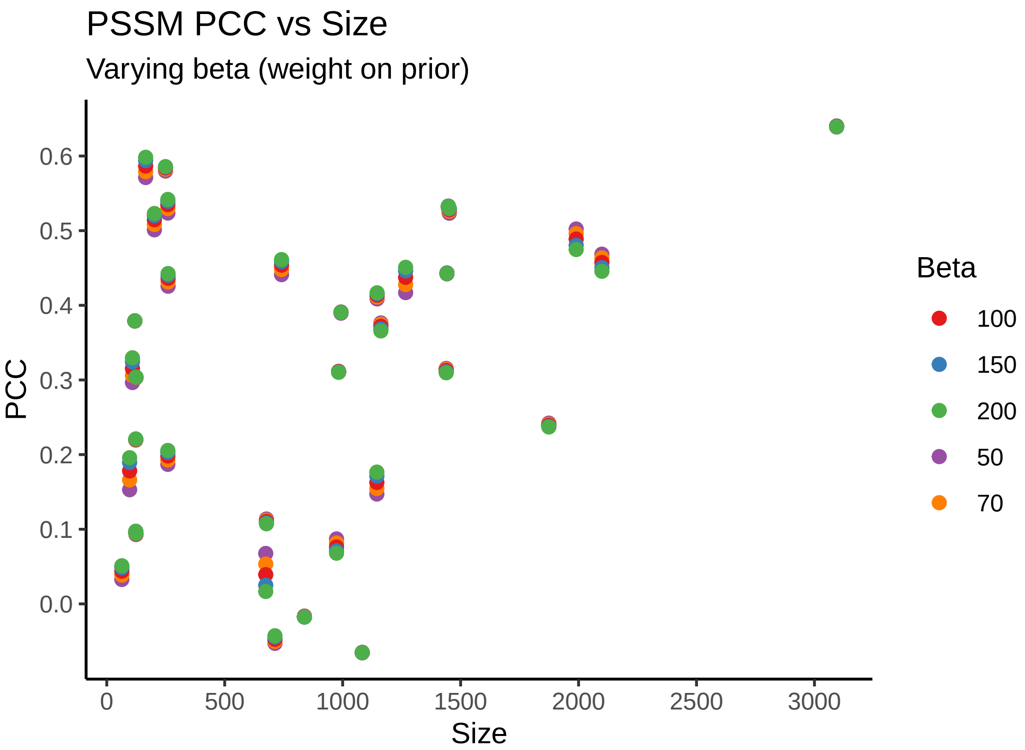
While this study is mainly about the prediction power of different algorithms, a small meta-study will be conducted, to determine how the dataset size (i.e. size of the protein allele) affects performance.

3) Position-Specific Scoring Matrix (PSSM)

3.1 Optimising beta

From the graphs, PCC performance between beta values is minimal.





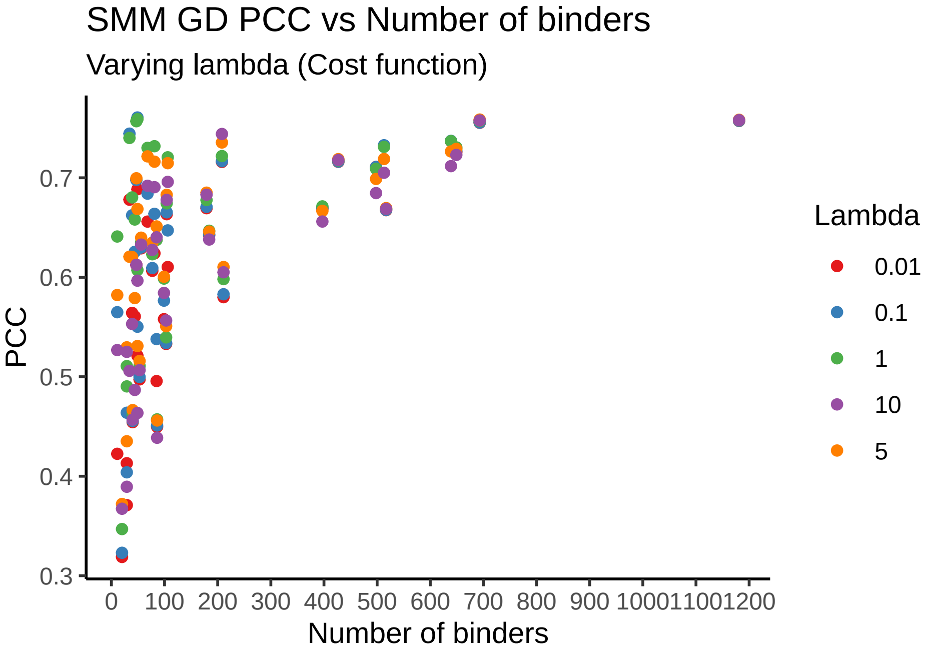
Due to beta not having any significant impact on performance, beta = 100 was arbitrarily

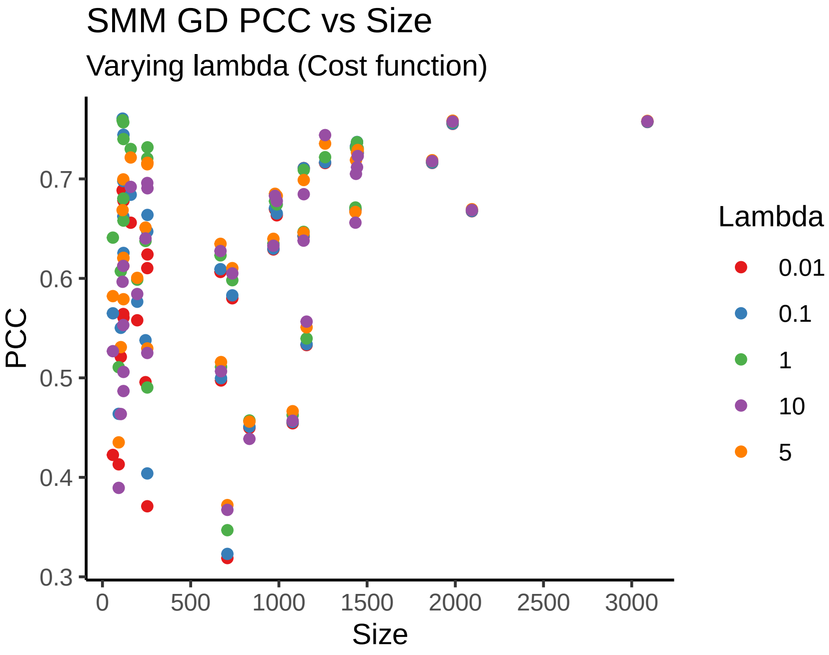
selected for the final comparison.

4) Stabilization Matrix Method (SMM)

We have assessed two implementations of the stabilization matrix method: Gradient Descent, and Monte Carlo. However for MC, no optimisations were made, due to long execution times, paired with low early-onset performance. Therefore, the initially run values are selected for the final comparison

4.1 Optimising lambda





4.2 Assessment of lambda

Lambda = 1 has best overall performance - by graph, and it has highest average.

```
## # A tibble: 1 x 1
##   `Lambda = 0.01 mean PCC`
##                      <dbl>
## 1                    0.600
```

```
## # A tibble: 1 x 1
##   `Lambda = 0.1 mean PCC`
##                     <dbl>
## 1                   0.624
```

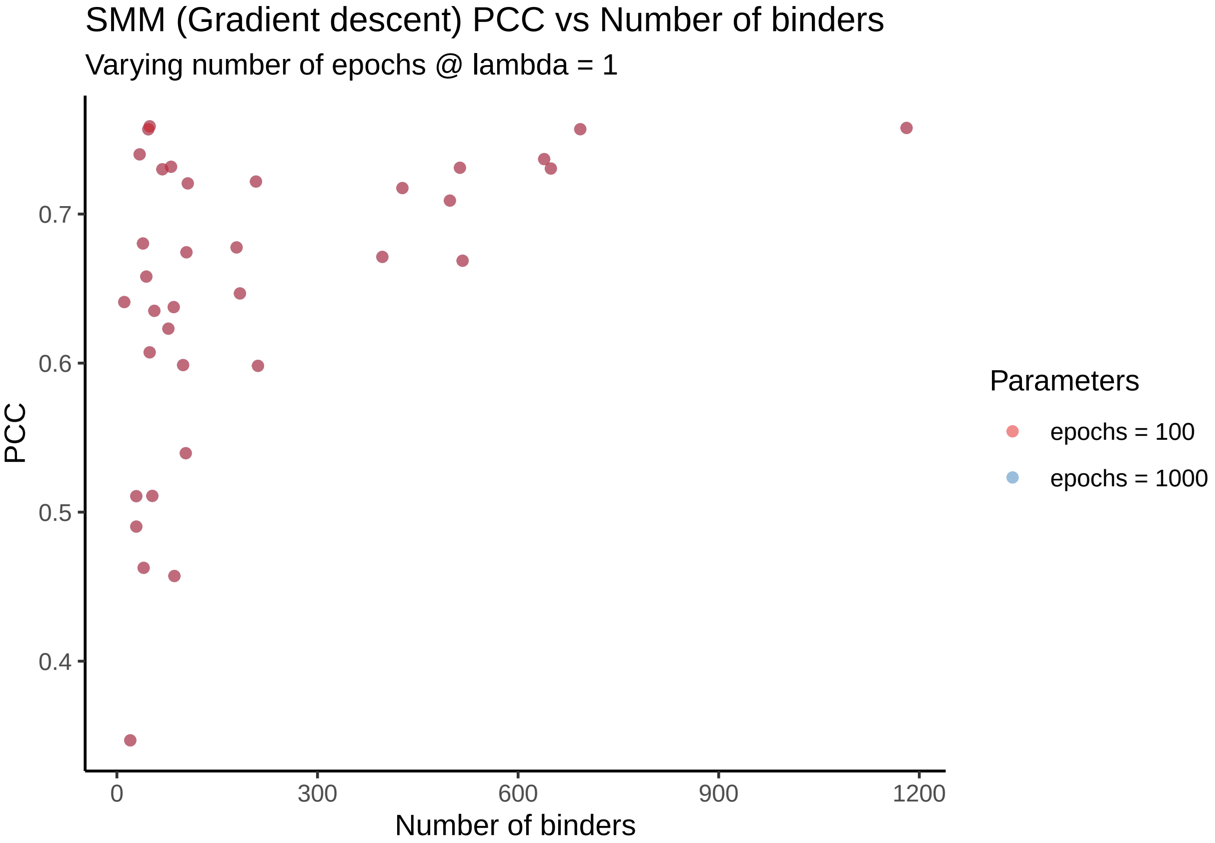
```
## # A tibble: 1 x 1
##   `Lambda = 1 mean PCC`
##                   <dbl>
## 1                 0.647
```

```
## # A tibble: 1 x 1
##   `Lambda = 5 mean PCC`
##                   <dbl>
## 1                 0.632
```

```
## # A tibble: 1 x 1
##   `Lambda = 10 mean PCC`
##                    <dbl>
## 1                  0.608
```

4.3 Optimising Epochs

This this last optimisation section was an attempt to run with increased number of epochs (100 -> 1000), @ lambda = 1 (the previously established optimal lambda parameter). The results were exactly the same - evidenced by the fact that a composite graph simply overlayed the points over each other, and the average PCC was identical. The ideal would have been to vary epsilon instead, but we declined, due to time constraints.

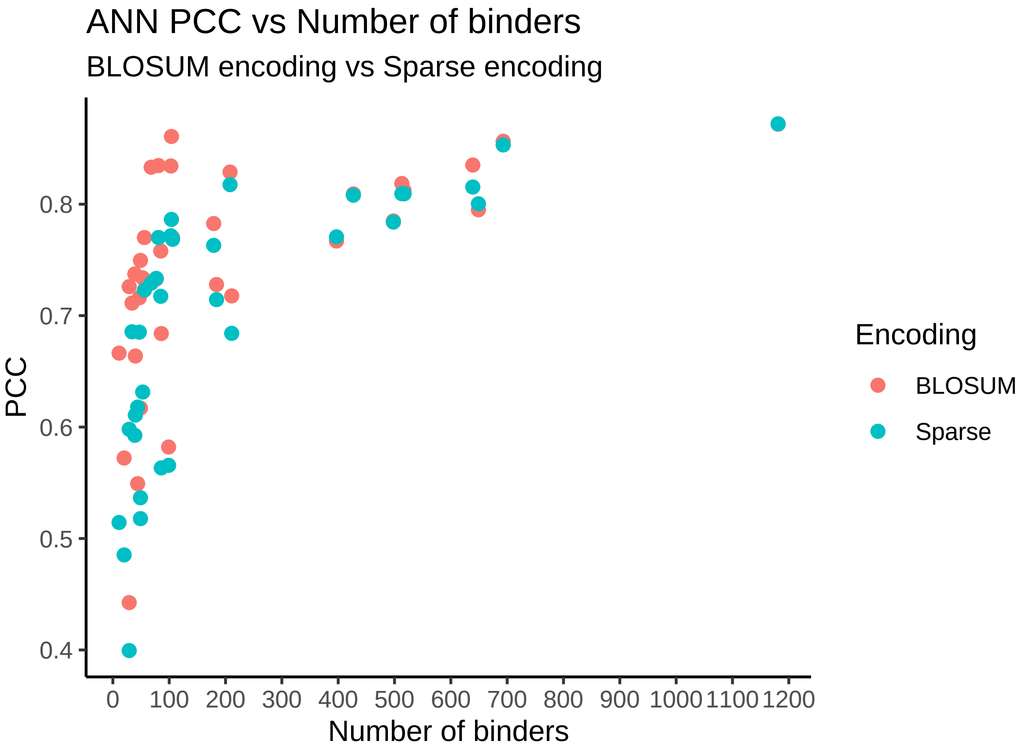


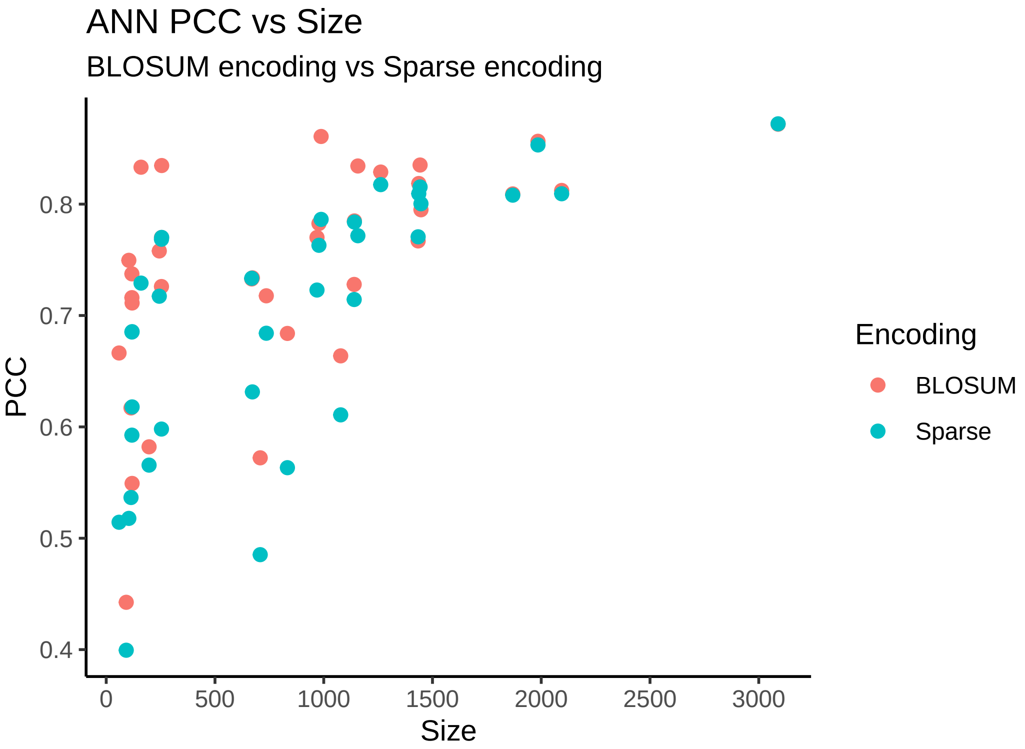
Due to the lack of difference between epochs, we arbitrarily selected epochs = 100 as our 'optimal'

5) Artificial Neural Network (ANN)

5.1 BLOSUM vs Sparse encoding

For ANN, there is the option of running with two parameters: sparse, or BLOSUM encoding. Here, we assess which option leads to the best results.





While it is apparent from the graph that BLOSUM encoding outperforms sparse encoding, the mean performances also support this.

```
## # A tibble: 1 x 1
##   `BLOSUM mean PCC`
##               <dbl>
## 1             0.741
```

```
## # A tibble: 1 x 1
##   `sparse mean PCC`
##               <dbl>
## 1             0.694
```

6) Comparison of all algorithms' performance

ALL results are compared using the optimal parameters.

- **PSSM**: beta = 100 - beta had no significant on performance, so we just use the original beta value
- **SMM: Gradient descent**: lambda = 1, epsilon = 0.05, epochs = 100 - optimised, from lambda 0.01 originally
- **SMM: Monte Carlo**: lambda = 0.01, epochs = 1000
- **ANN**: BLOSUM encoding

6.1 Average PCC

ANN and SMM Gradient descent are the highest overall performers, whereas PSSM and SMM Monte Carlo are the poorest performers

```
## # A tibble: 1 x 4
##   `PSSM mean PCC` `GD mean PCC` `MC mean PCC` `ANN mean PCC`
##             <dbl>         <dbl>         <dbl>          <dbl>
## 1           0.308         0.647         0.393          0.741
```

6.2 Max PCC

Interestingly enough, maximum performance shows far less discrepancies.

```
## # A tibble: 1 x 4
##   `PSSM max PCC` `GD max PCC` `MC max PCC` `ANN max PCC`
##            <dbl>        <dbl>        <dbl>         <dbl>
## 1           0.64        0.759        0.779         0.872
```

6.3 Negative PCC values

```
## # A tibble: 1 x 1
##   `No. negatives in PSSM`
##                     <dbl>
## 1                       3
```

```
## # A tibble: 1 x 1
##   `No. negatives in GD`
##                   <dbl>
## 1                     0
```

```
## # A tibble: 1 x 1
##   `No. negatives in MC`
##                   <dbl>
## 1                     3
```

```
## # A tibble: 1 x 1
##   `No. negatives in MC`
##                   <dbl>
## 1                     0
```

3 negative values in PSSM and MC. Interestingly enough, there are 3 different alleles in each case.

PSSM

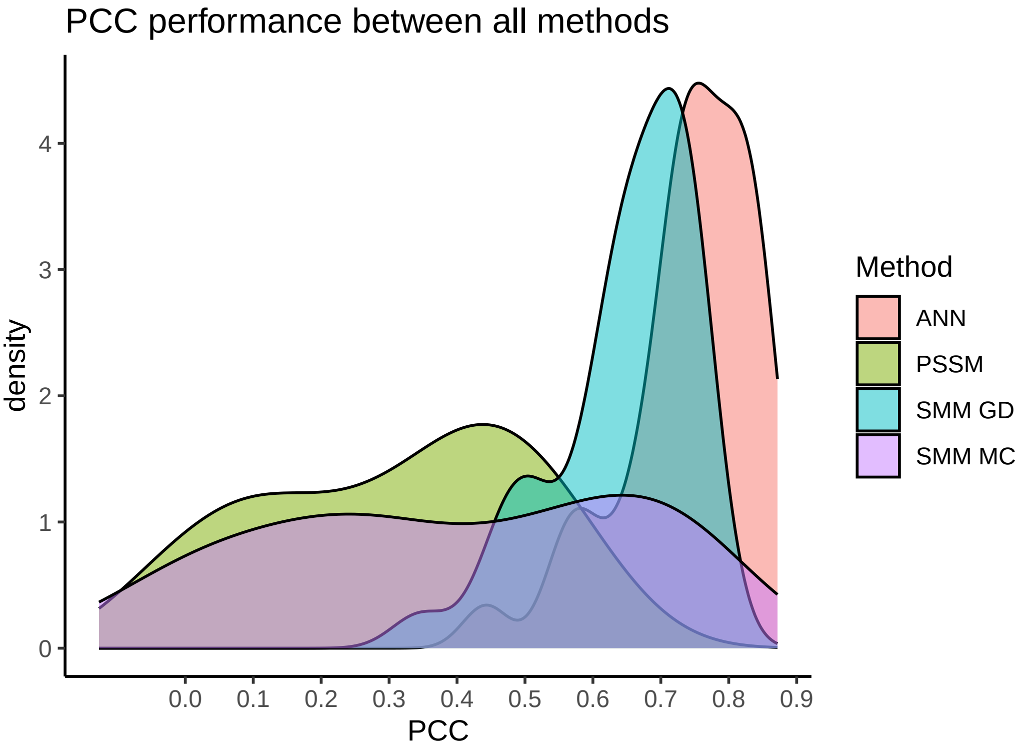
```
## # A tibble: 3 x 4
##   Allele  Size     PCC `Number of binders`
##   <chr>  <dbl>   <dbl>               <int>
## 1 A6901    838 -0.0175                 86
## 2 B0801    713 -0.0475                 20
## 3 B4001   1083 -0.0655                 40
```

MC

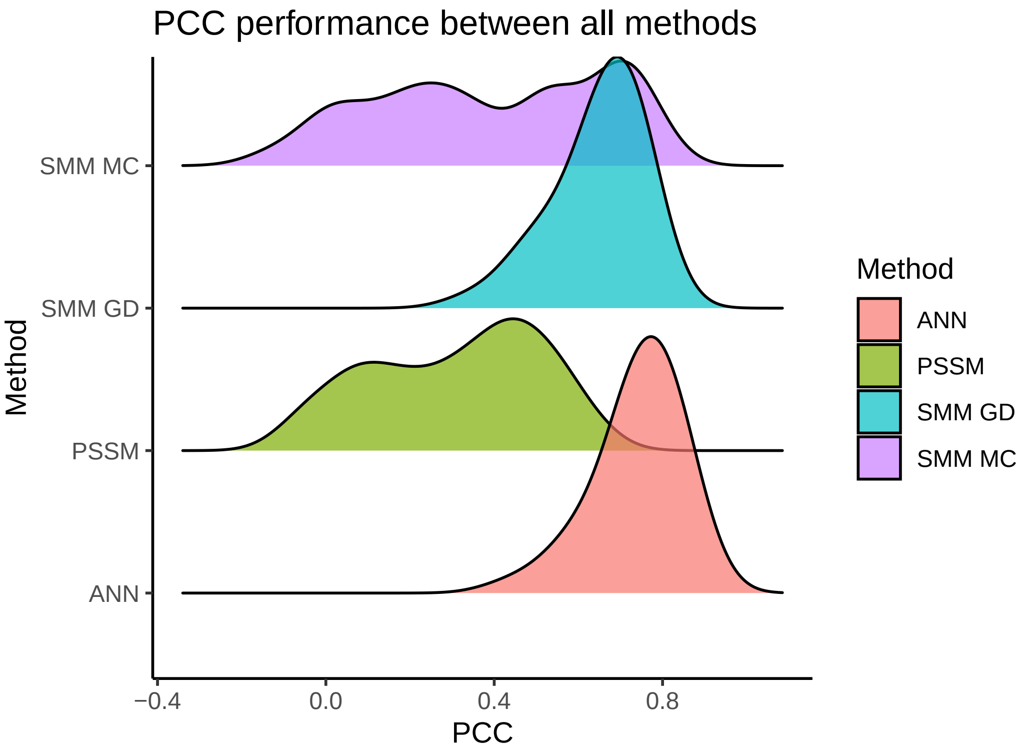
```
## # A tibble: 3 x 4
##   Allele  Size      PCC `Number of binders`
##   <chr>  <dbl>    <dbl>               <int>
## 1 B4002    118 -0.127                    39
## 2 B4403    119 -0.00242                  34
## 3 B5701     59 -0.0316                   11
```

6.4 Visualise performance for ALL methods

We have used a desntiy plot to visualise all methods' performances. ANN and gradient descent (SMM GD) achieve the highest PCC densities, with more pronounced peaks - indicating a more consistent performance throughout the dataset. PSSM appears to have a median PCC of ~0.45 but drops rapidly past this point and therefore less consistent. Monte Carlo's performance is the most evenly distributed - and therefore the most unaffected by size - but achieves middling results.



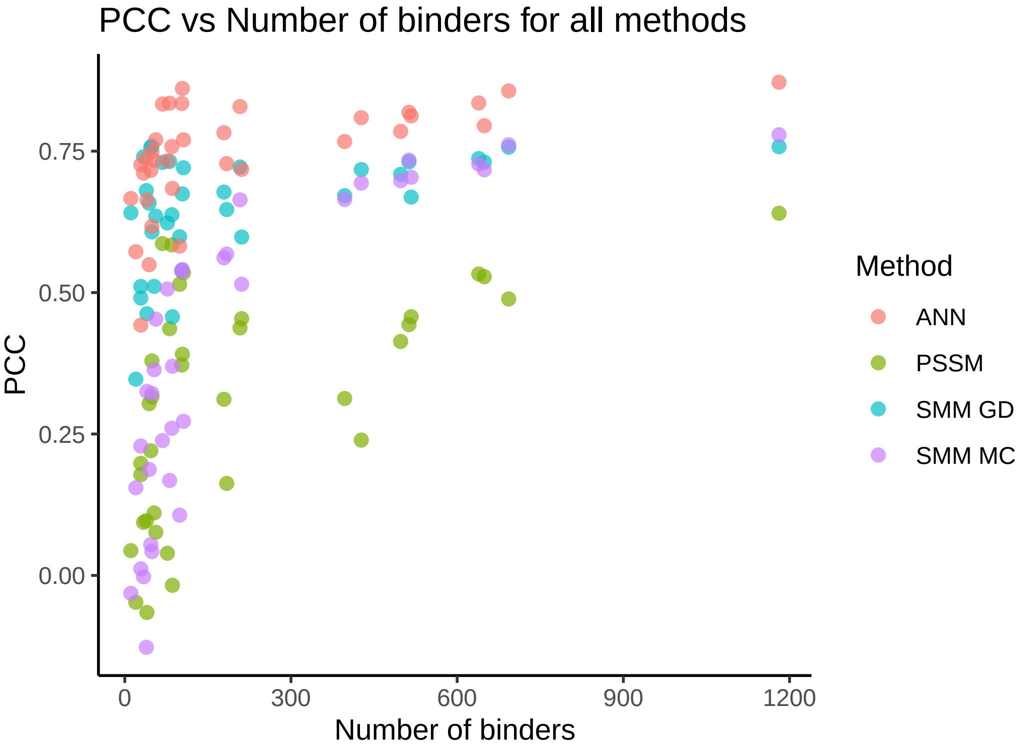
Alternate plot (ridgeline) This ridgeline plot of the densities helps distinguish the PCC densities. While the shapes are slightly different, they tell a similar story. ANN and SMM GD are have the highest and most consistent performance, and MCC and PSSM having the middling performances.

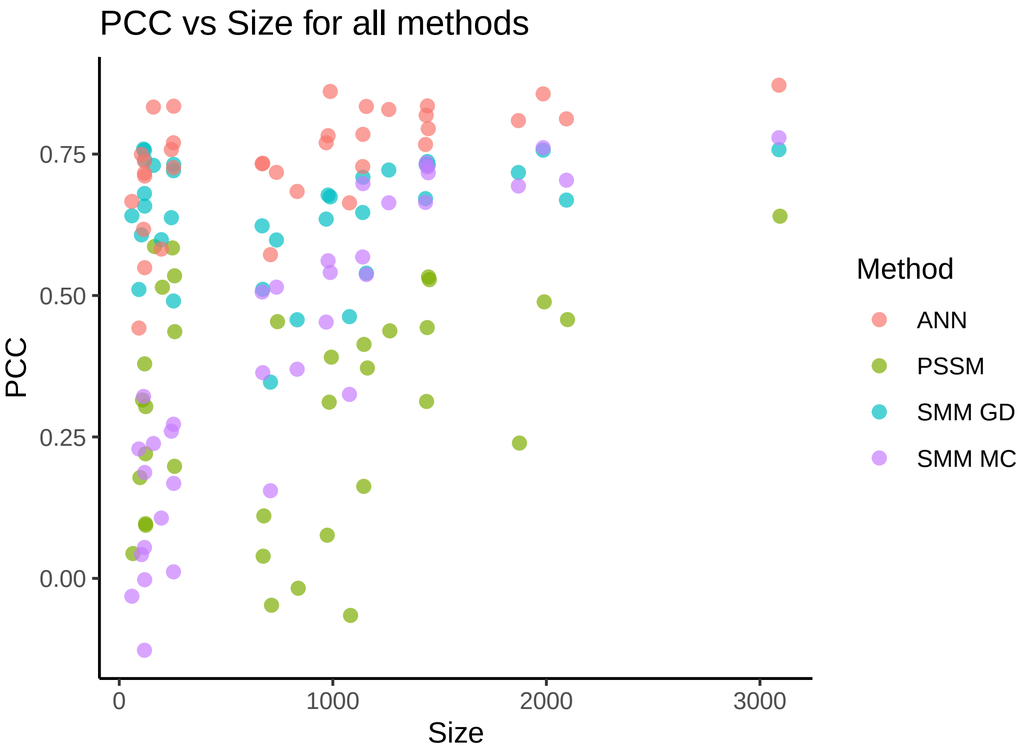

PCC performance between all methods
Method
SMM MC
SMM GD
PSSM
ANN
−0.4
0.0
0.4
0.8
PCC
Method
ANN
PSSM
SMM GD
SMM MC

7) Meta-study: Affect of dataset size and number of binders on performance

When plotting the PCC values against number of binders, it appears that a critical threshold ~300 binders must be met, before performance is maintained consistently throughout different numbers of binders between alleles.

On the other hand, this trend is much less pronounced for a plot of PCC values against dataset size.

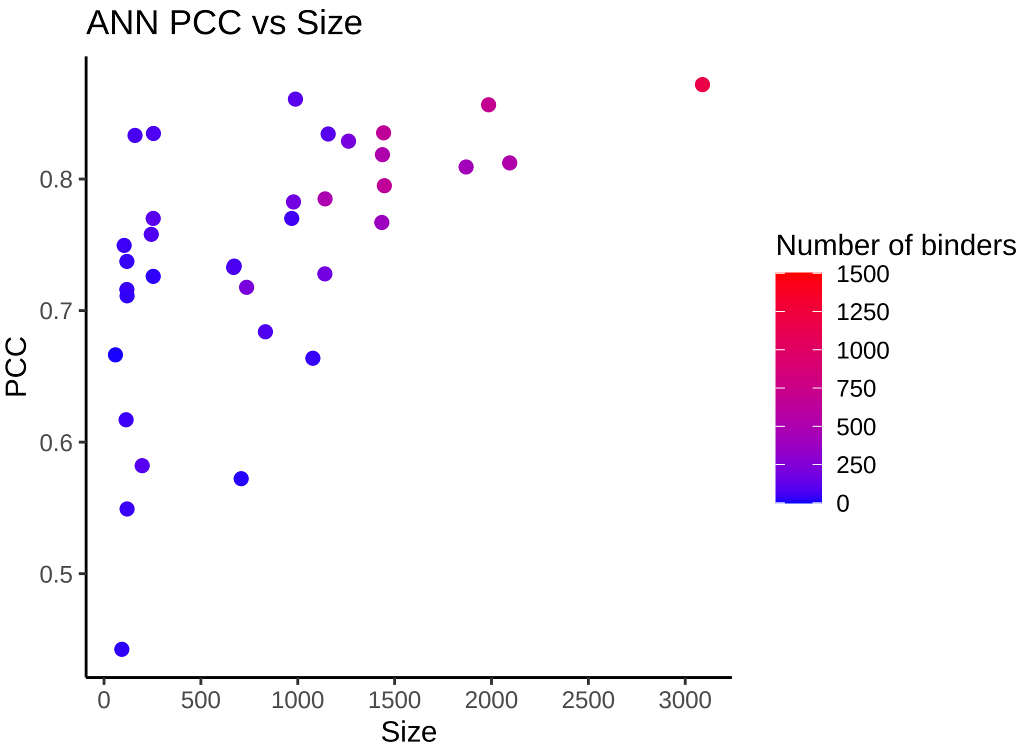




From these findings, it appears that the number of binders appears to be a more consistent predictor of performance.

For more quantitative findings and insights on this meta-study, please refer to the `ProjectReport.pdf`.

In addition, graphs for PCC vs number of binders stratified against size (and vice versa) were plotted - example:



For the sake of preserving space, the rest of them have not been included in this report. However they are featured and discussed in in the `ProjectReport.pdf`, and high quality exports for all methods are in the `plots` directory.