

Real Data: 16s rRNA

Yichen Han

2024-08-13

After modifying IG implementation and testing it on synthetic data, we now want to work with real data. We choose a rather simple task as already done by the package `dvp-team`: 16S rRNA gene detection vs. bacterial genomes.

Pretrained Model

We load the best model as instructed by the online tutorial.

```
## Using checkpoint checkpoints/16S_vs_bacteria_checkpoints/Ep.005-val_loss0.01-val_acc0.999.hdf5

## [[1]]
## [[1]]$confusion_matrix
##           Truth
## Prediction  16s  bacteria
##    16s      1250         1
##    bacteria    0      1249
##
## [[1]]$accuracy
## [1] 0.9996
##
## [[1]]$categorical_crossentropy_loss
## [1] 0.004219269
##
## [[1]]$AUC
## [1] 1
##
## [[1]]$AUROC
## NULL
```

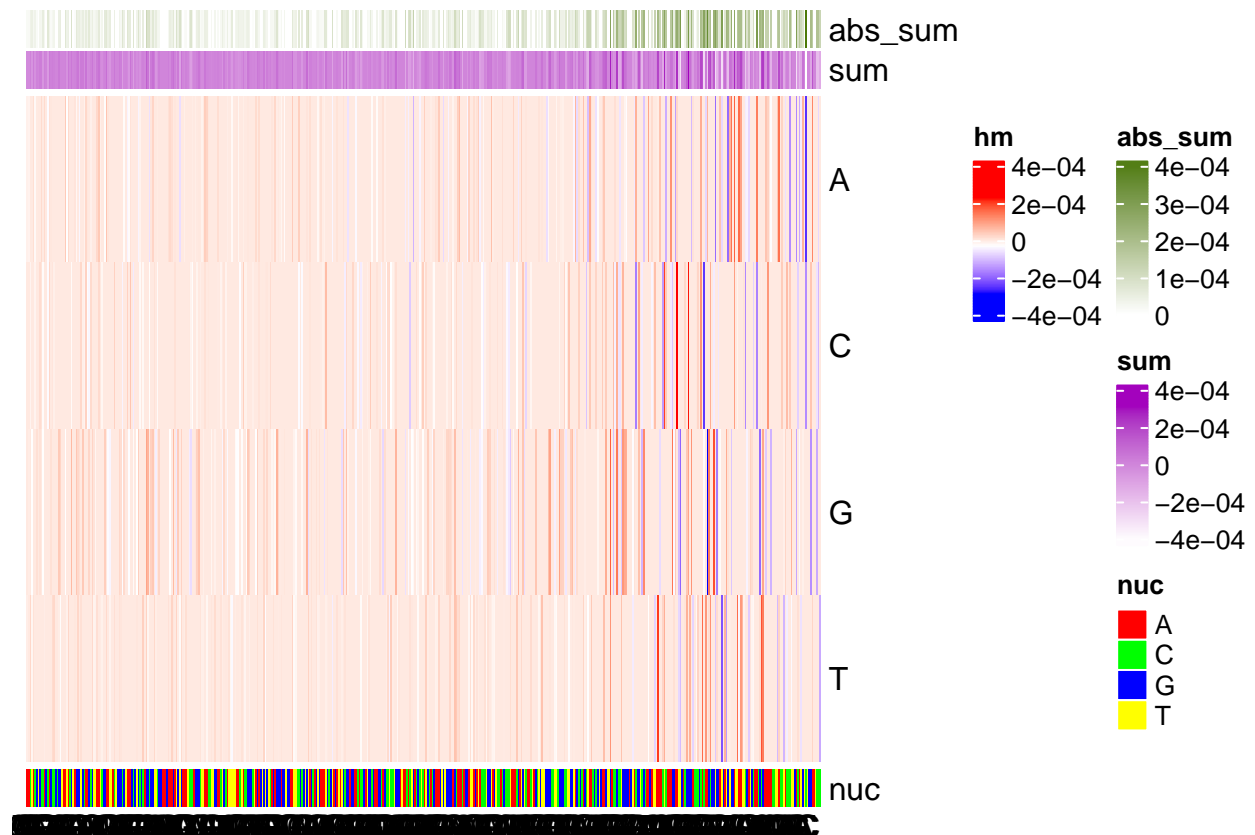
We see that the model has very high accuracy.

We then load a random 16S gene sequence from the validation data, and load a bacteria genome from the validation data as our baseline. We extract positions 499-998 (500 bp) for both sequences.

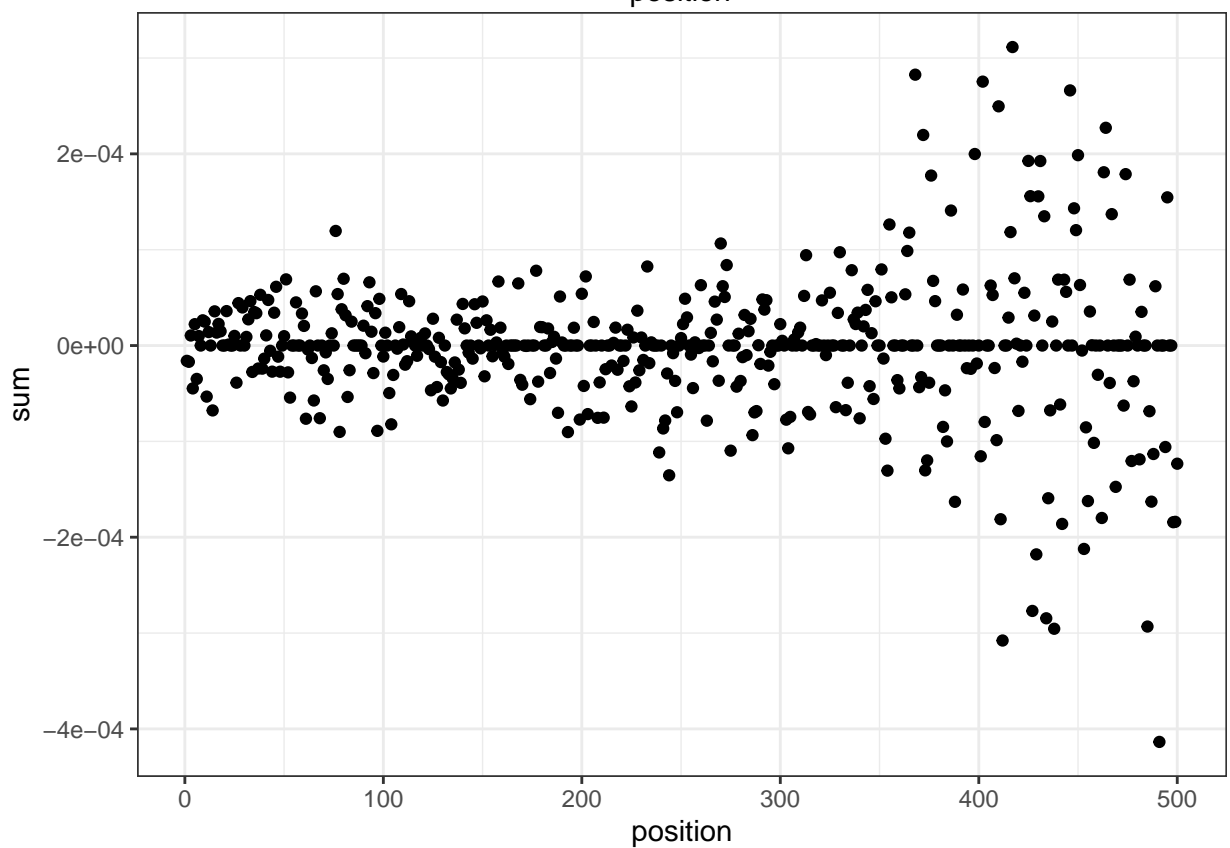
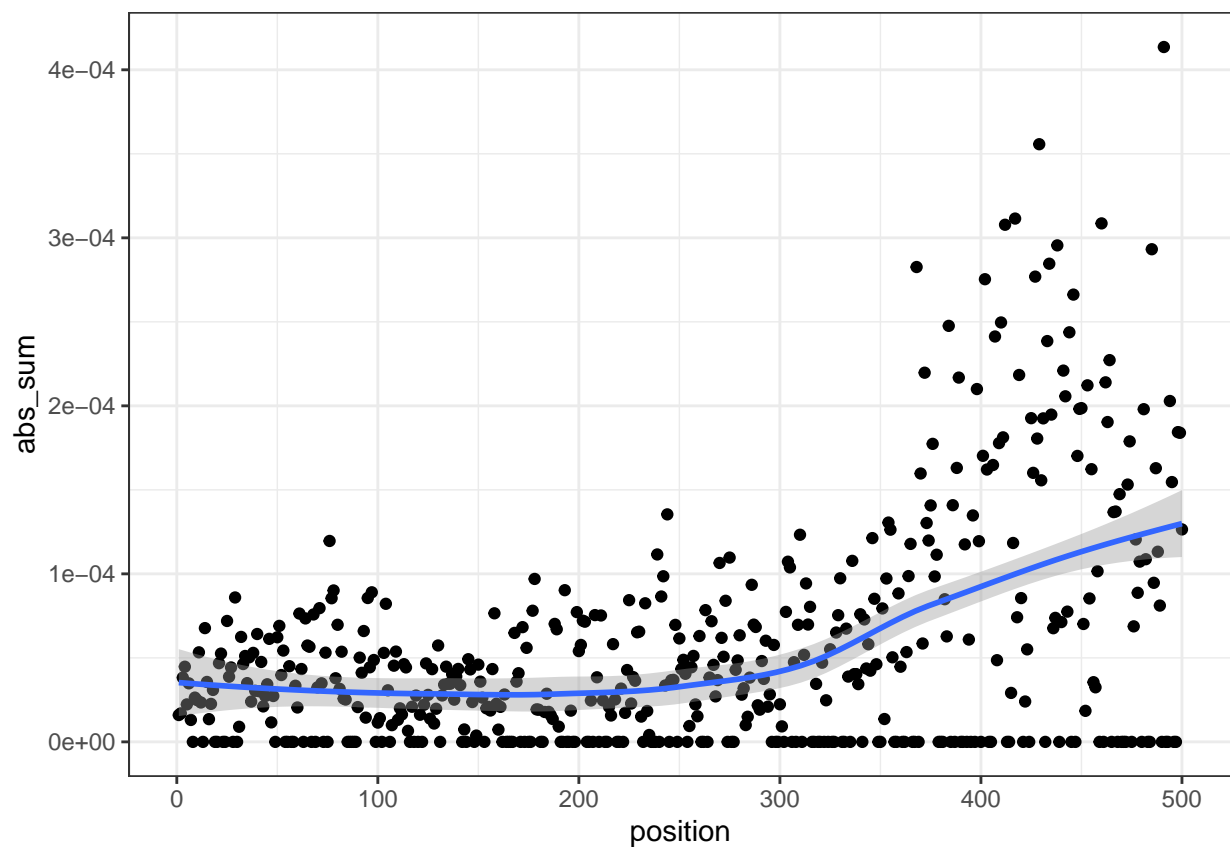
We then perform modified IG.

```
##           [,1]           [,2]
## [1,] 0.9987847 0.001215239

## [[1]]
```



```
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
```



We noticed approximately 0 importance assigned to the first 300 bp, and a monotonic rise in the last 200.

We repeat the process on different targets.

We discovered that the emphasis on the tail is recurrent whatever instance, baseline, or sequence starting index. This does not reflect the true traits of 16S rRNA gene, and is not logical.

We noticed the model specifications:

```
model <- create_model_lstm_cnn(  
  maxlen = 500, # not divisible by 3  
  layer_lstm = NULL,  
  layer_dense = c(2L),  
  vocabulary_size = 4,  
  kernel_size = c(12, 12, 12),  
  filters = c(32, 64, 128),  
  pool_size = c(3, 3, 3),  
  learning_rate = 0.001)  
  
train_model(train_type = "label_folder",  
  model = model,  
  path = c(path_16S_train, path_bacteria_train),  
  path_val = c(path_16S_validation, path_bacteria_validation),  
  vocabulary_label = c("16s", "bacteria"),  
  path_checkpoint = checkpoint_path,  
  train_val_ratio = 0.2,  
  run_name = run_name,  
  batch_size = 256,  
  steps_per_epoch = 25,  
  epochs = 8,  
  save_best_only = FALSE,  
  step = c(100, 500), # take sample every 100 step for 16S and every 500 for bacteria  
  proportion_per_seq = c(0.95, 0.05))
```

The length is not divisible by 3, and when the instance is introduced after `substr`, the original codon structure is broken, and the model did very likely not capture codon structure during the training at all.

We speculate the model to be not biologically trustworthy. We thus retrain a model.

```
model <- create_model_lstm_cnn(  
  maxlen = 600, # size divisible by 3  
  ...)  
  
train_model(train_type = "label_folder",  
  ...  
  step = c(6, 30), # smaller steps, divisible by 3  
  ...)
```

Retrained Model

We load the model at last epoch:

```
## Using checkpoint checkpoints/16S_vs_bacteria_full_2/Ep.008-val_loss0.13-val_acc0.991.hdf5  
  
## [[1]]
```

```
## [[1]]$confusion_matrix
##           Truth
## Prediction  16s  bacteria
##    16s      1198         1
##    bacteria   52      1249
##
## [[1]]$accuracy
## [1] 0.9788
##
## [[1]]$categorical_crossentropy_loss
## [1] 0.1410726
##
## [[1]]$AUC
## [1] 0.9999059
##
## [[1]]$AUPRC
## NULL
```

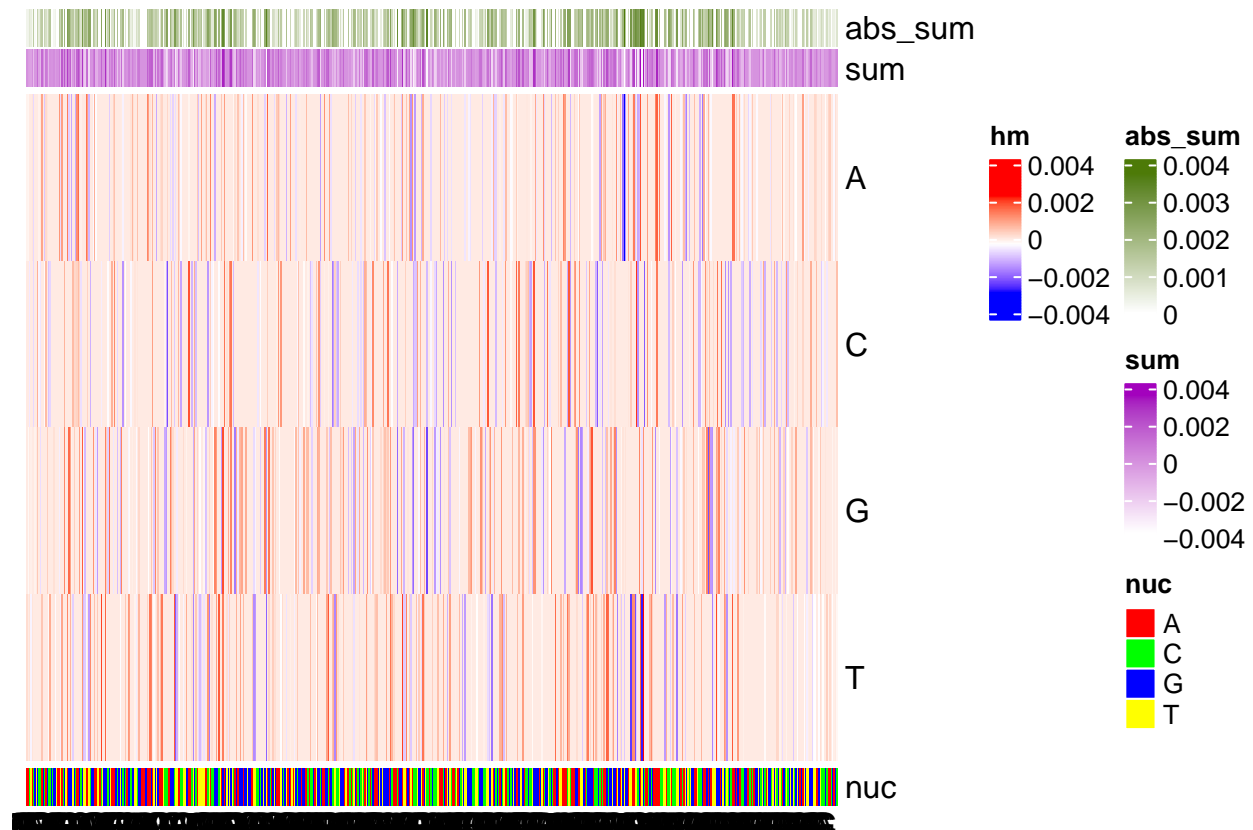
It still has a very high accuracy. We evaluate it again using modified IG.

Instance: GCF_001986655.1_ASM198665v1_genomic.16s.fasta 499-1098 bp.

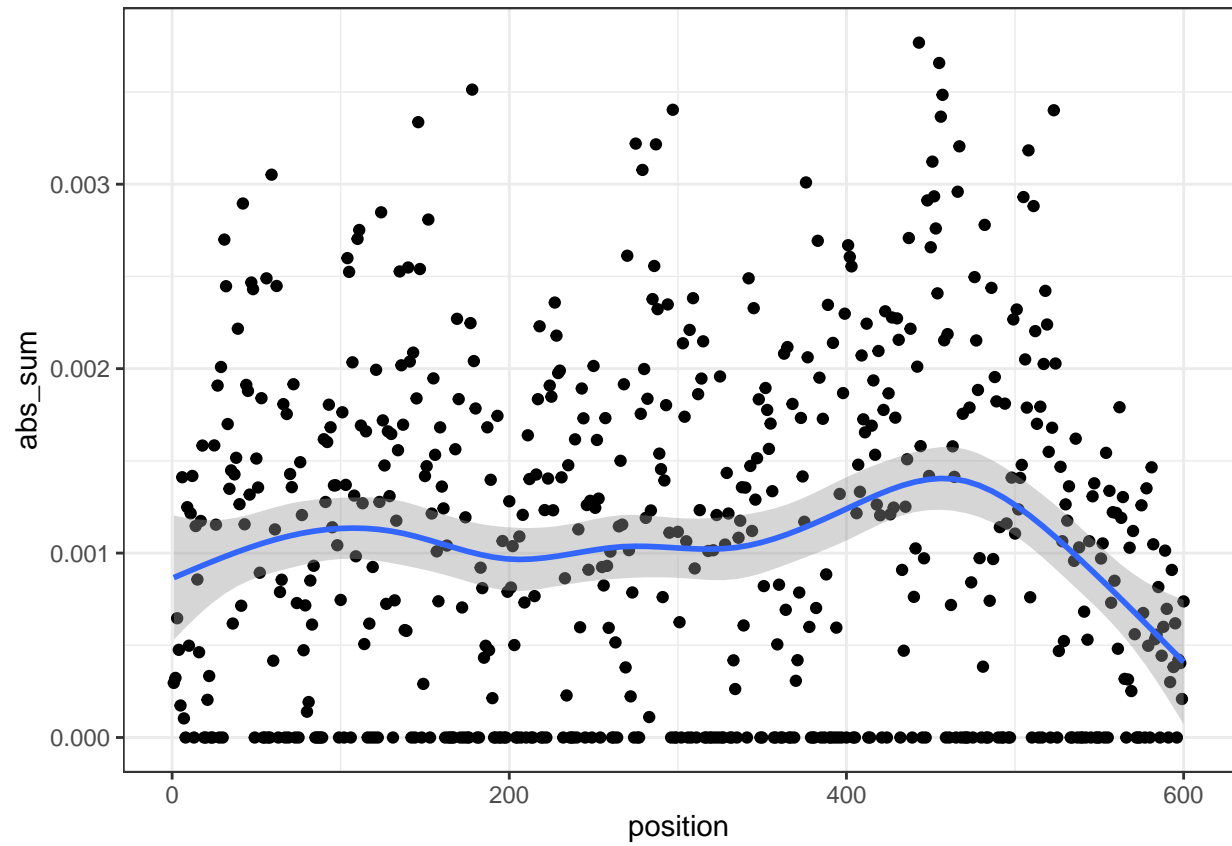
Baseline: GCF_002895085.1_ASM289508v1_genomic.fasta 499-1098 bp.

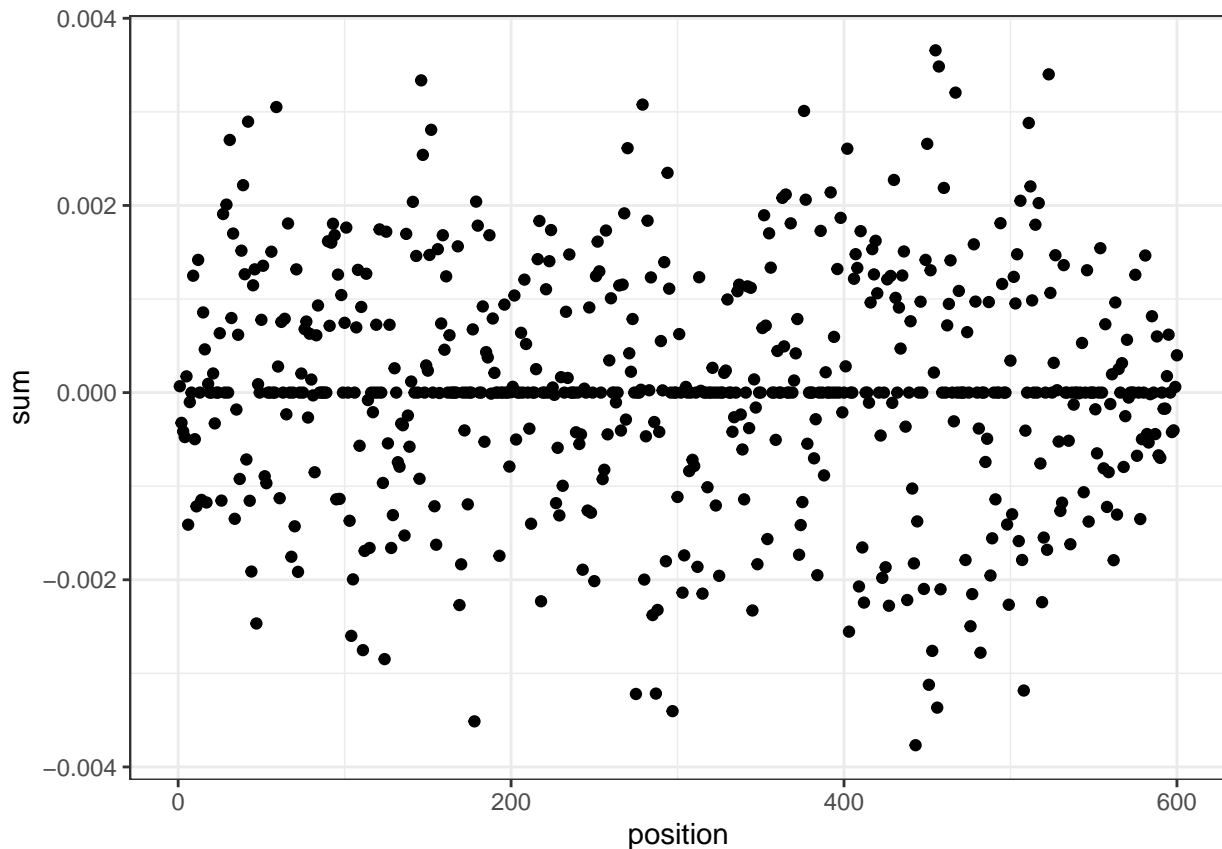
```
##           [,1]      [,2]
## [1,] 0.906086 0.09391395
```

```
## [[1]]
```



```
## 'geom_smooth()' using formula = 'y ~ s(x, bs = "cs")'
```





This time, the importance is scattered across positions in a more complex way, which reflects the nature of genetic data.

We also visualize the direct sum, since some points persistently have negative gradients. The implication of these is still unknown.

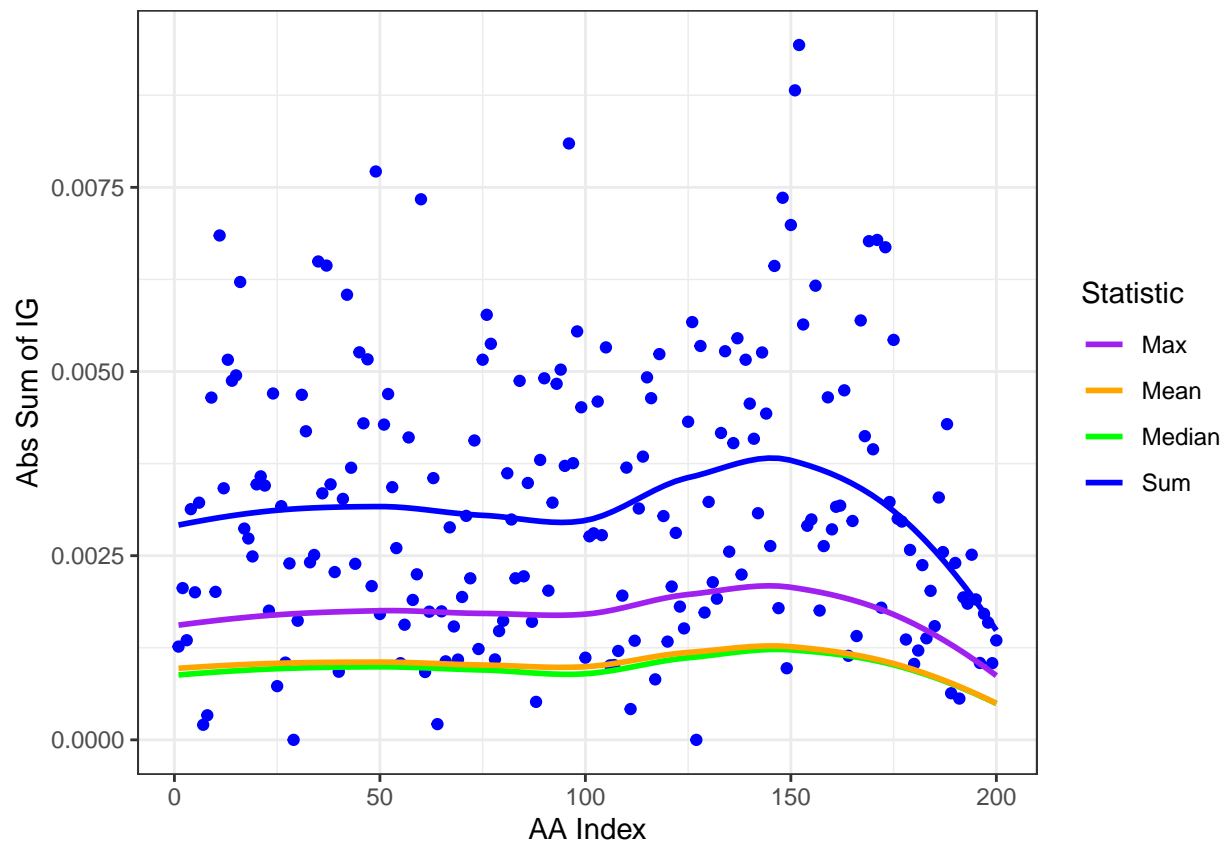
We may notice a peak at the second half of the absolute sum curve. We speculate this is due to the loess method applied, and the effect is more random than artifact. The totally unpatterned sum plot justifies this.

This can also be the effect of a LSTM model. Can be further studied.

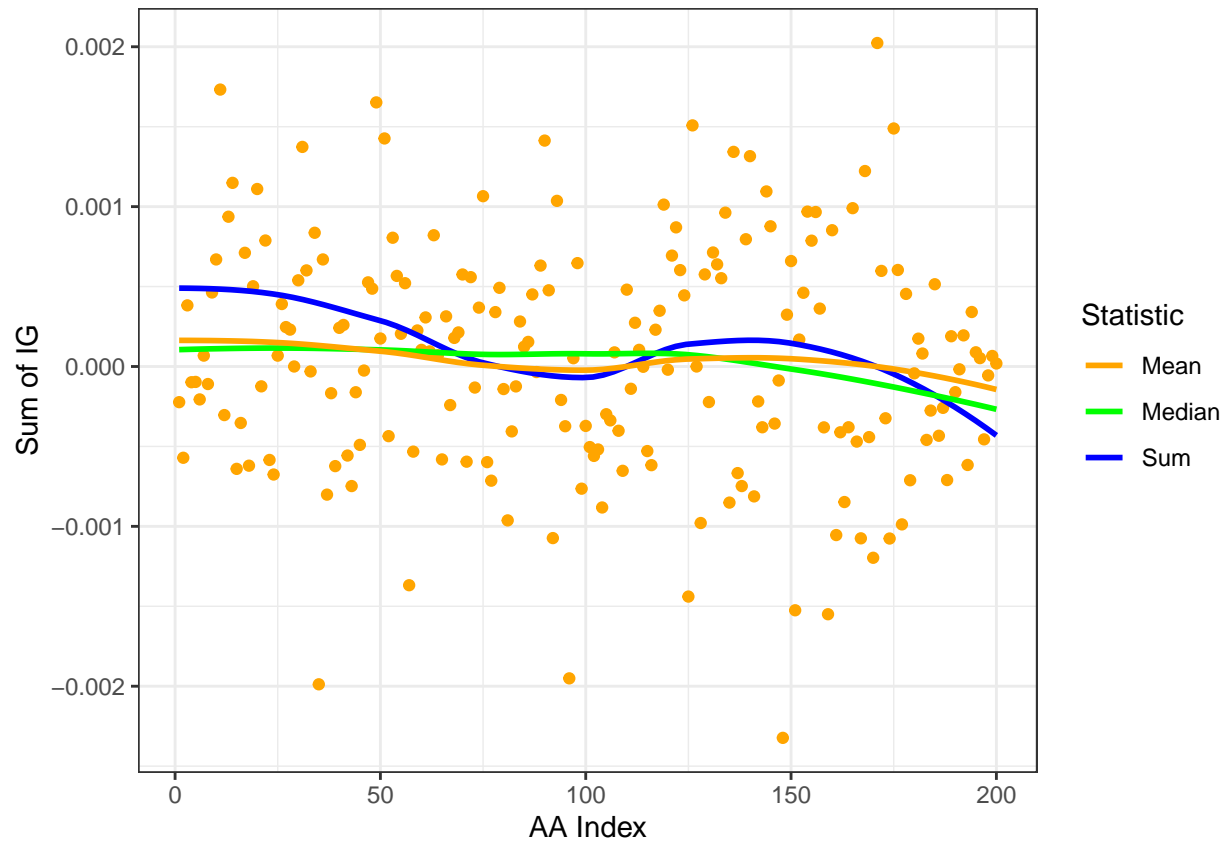
If the peak is not random and the loess is correct, moving the targeted sequence should also change the peak. As a mere check we move our instance 250 bp to the right, that is 748 to 1348 bp. The smooth curve looks the same as now, for simplicity we do not show it again.

The same information compression technique is shown below.

```
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
```

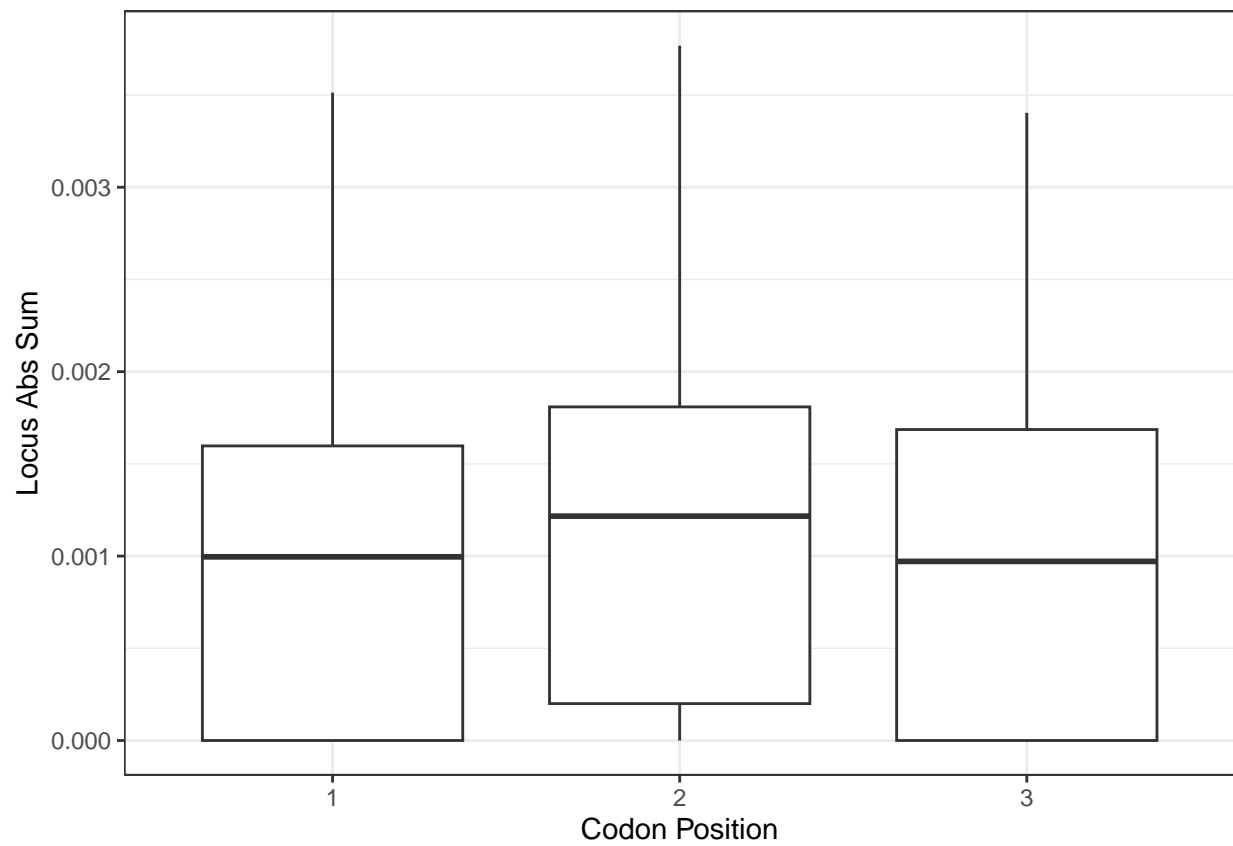
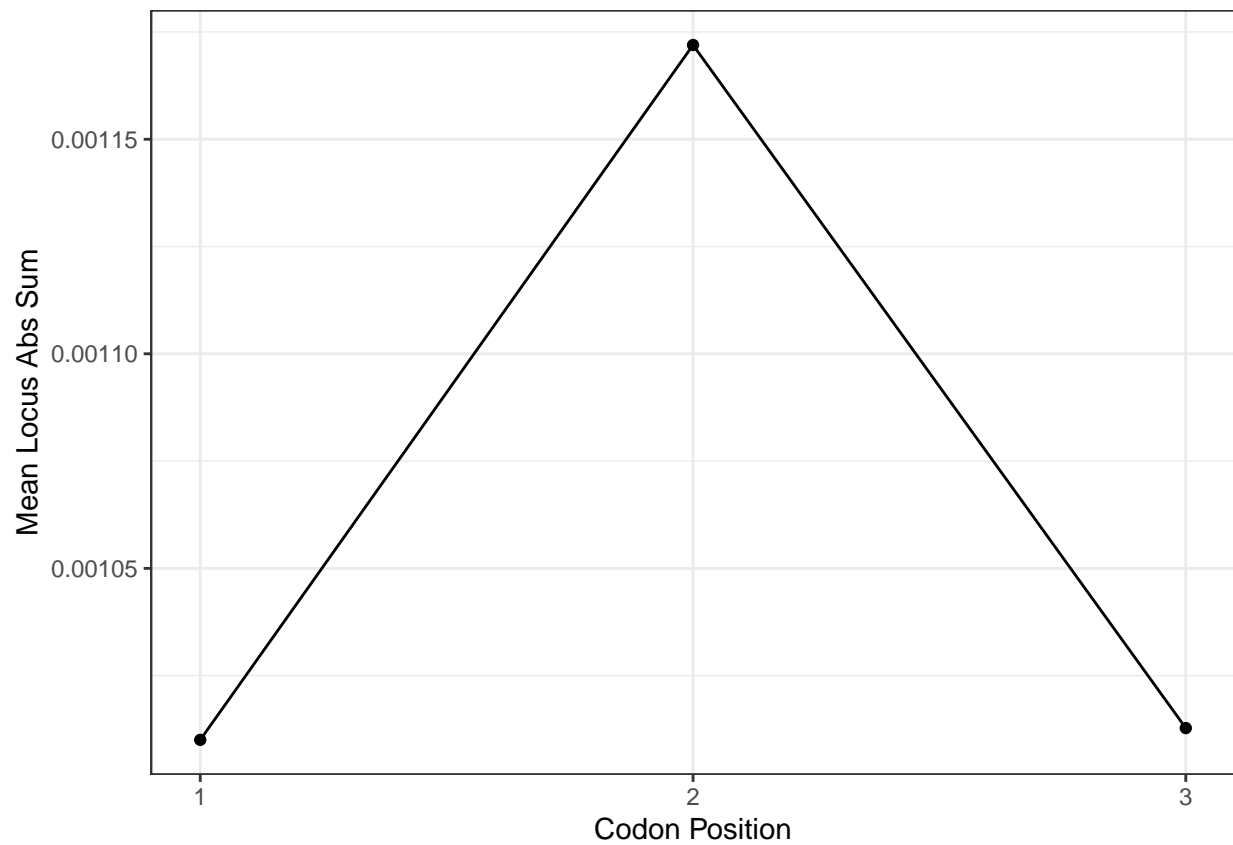


```
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
## 'geom_smooth()' using method = 'loess' and formula = 'y ~ x'
```

We further studied wobbliness (see Durrant & Bhatt). In this case, more randomness is introduced by the fact that we only calculated IG once. In RSDexport, the average importance almost consistently follow the 2-1-3 structure. Here, we tested some other instances and baselines and wobbliness is not always detected.

Here is an example of existing wobbliness.

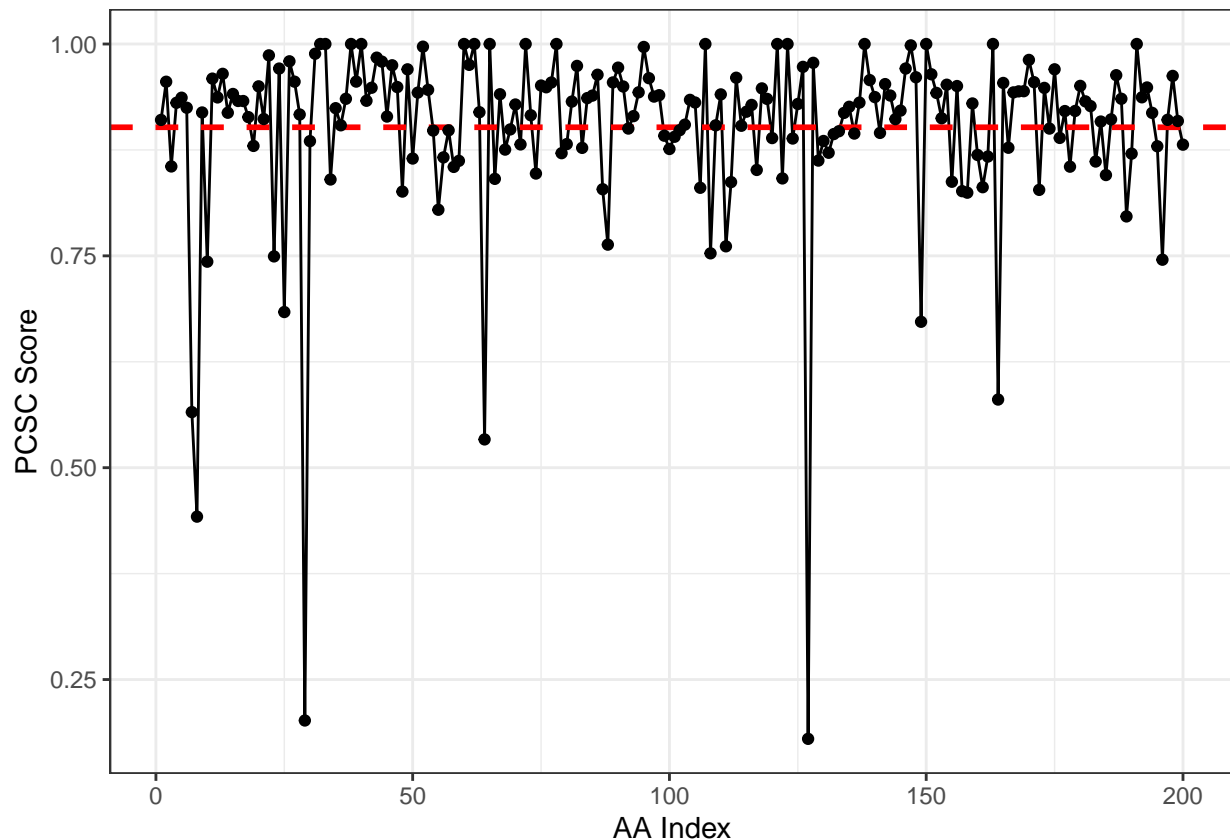


```
##
## Call:
## lm(formula = abs_sum ~ factor(position), data = codon_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.172e-03 -1.010e-03  9.090e-06  6.392e-04  2.596e-03
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.010e-03  6.453e-05  15.652  <2e-16 ***
## factor(position)2  1.619e-04  9.126e-05   1.774   0.0765 .
## factor(position)3  2.739e-06  9.126e-05   0.030   0.9761
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0009126 on 597 degrees of freedom
## Multiple R-squared:  0.006866, Adjusted R-squared:  0.003539
## F-statistic: 2.064 on 2 and 597 DF, p-value: 0.1279
```

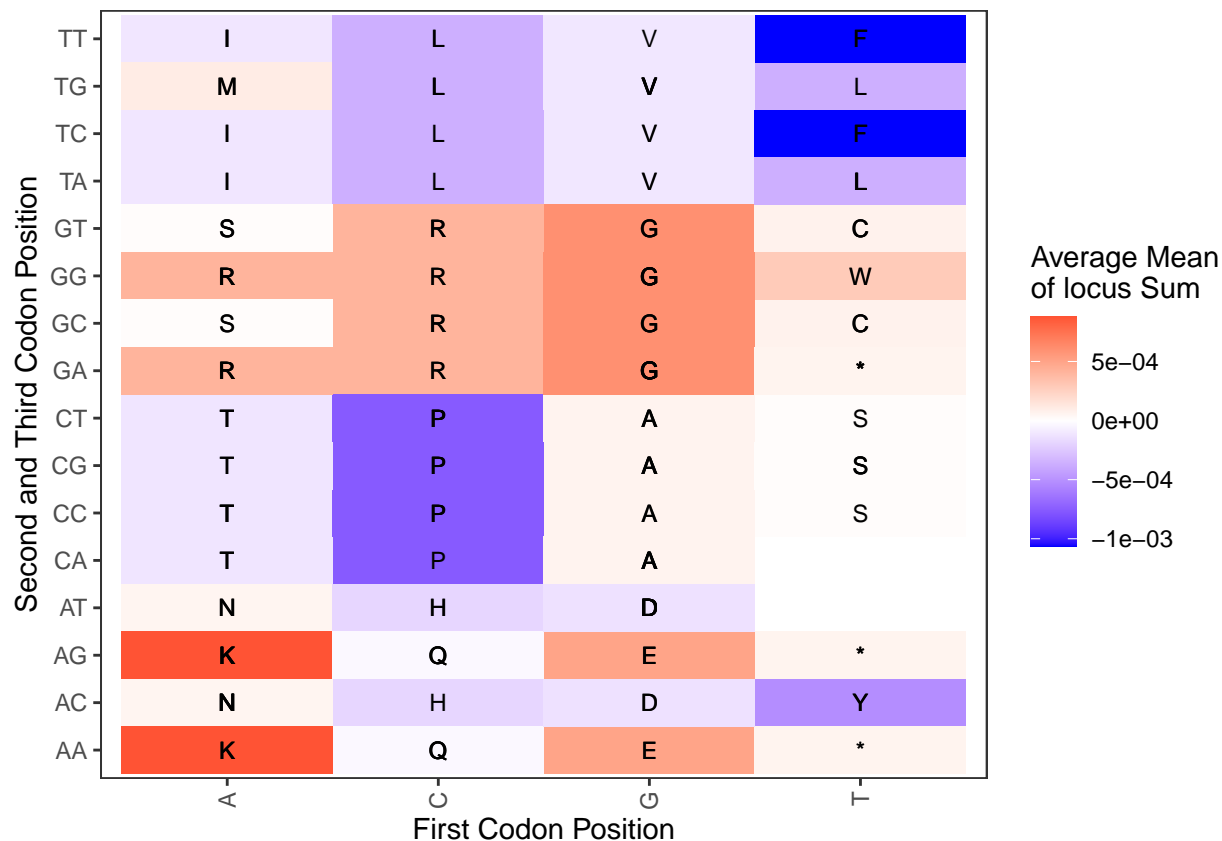
Consistency

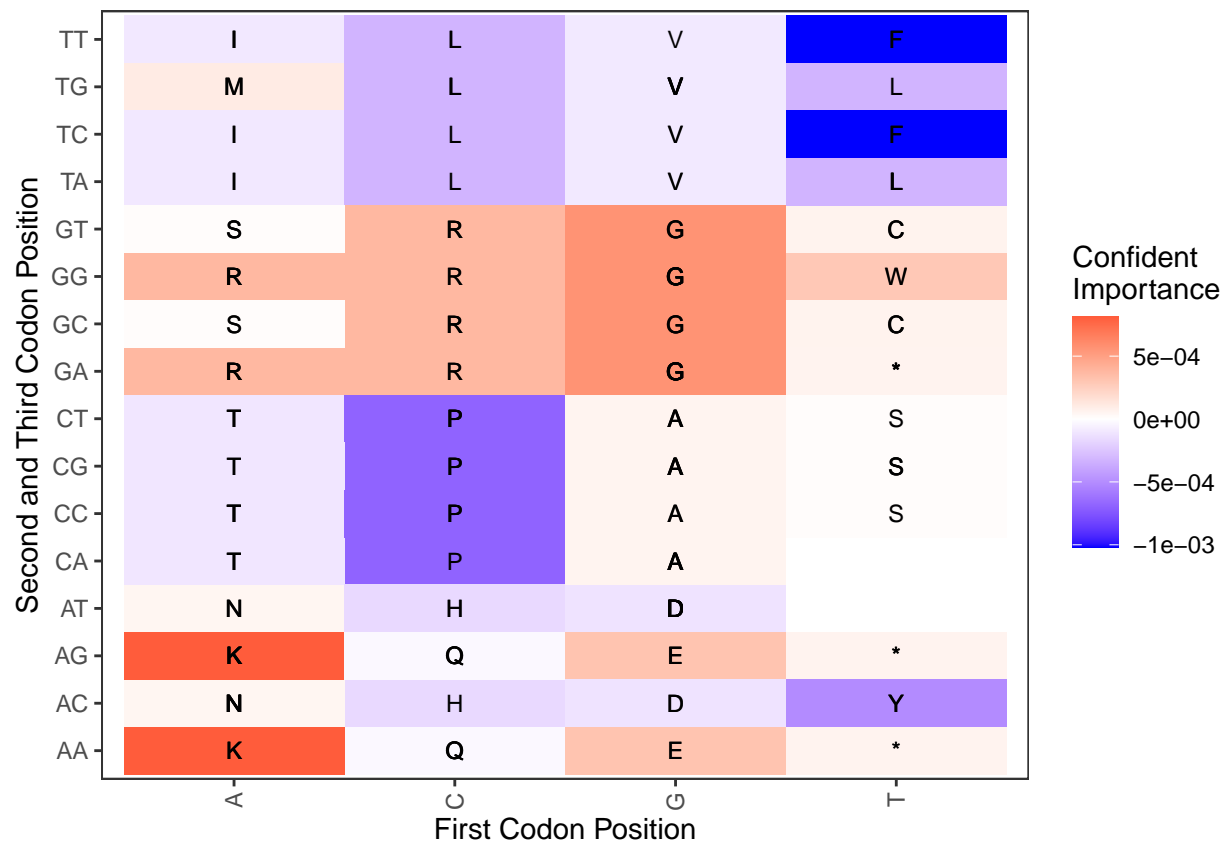
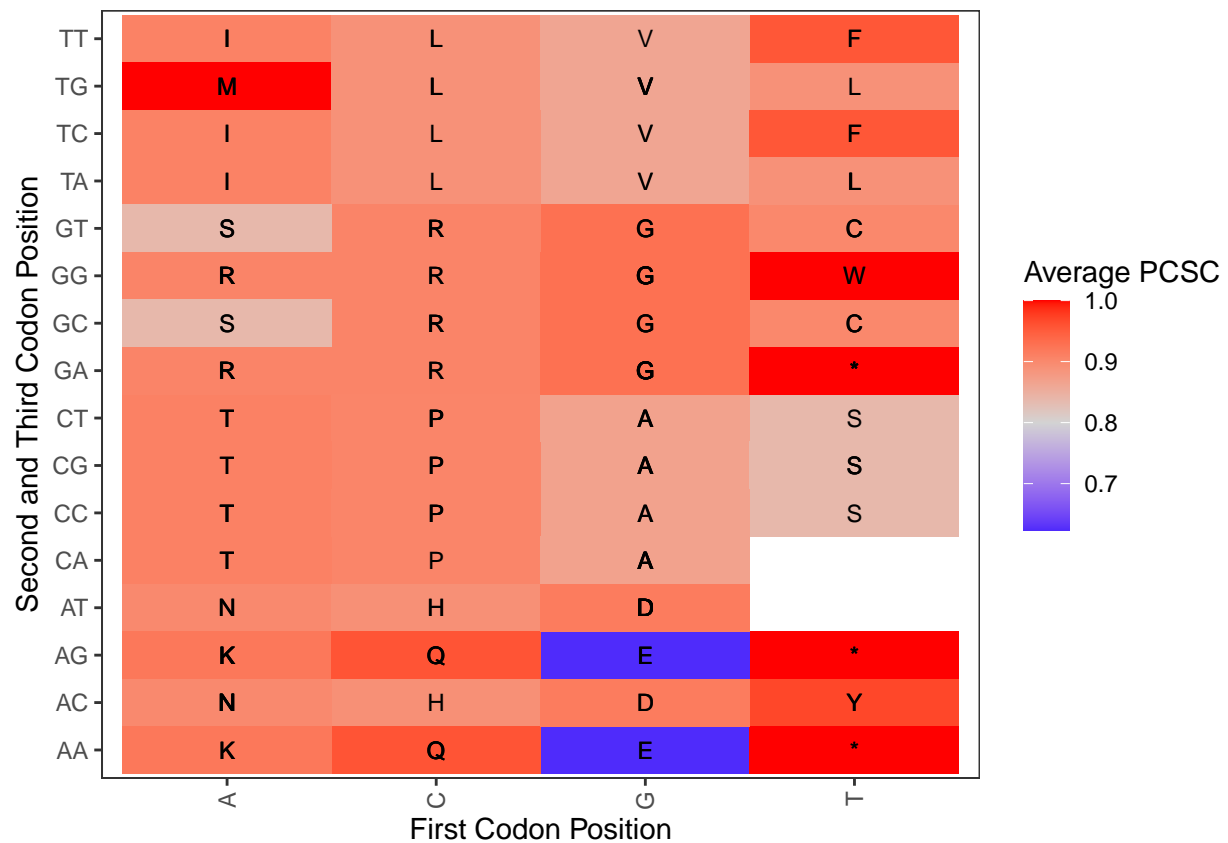
We then calculated PCSC score on our data. With an average score at 0.9, we conclude that the model was able to capture synonymous codons and treats them consistently.

```
## Loaded result_df from existing CSV file.
```



We then generated heatmaps as in RSDexport. The first heatmap visualizes average sum of gradients, the second average consistency based on AA, and the third the “confident importance”, which is average sum of gradients multiplied by PCSC score.





Feature Selection

We then implemented the feature selection algorithm as by the enhanced IG paper. We do this based on locus level, and sample 50 events for both interest and random group.

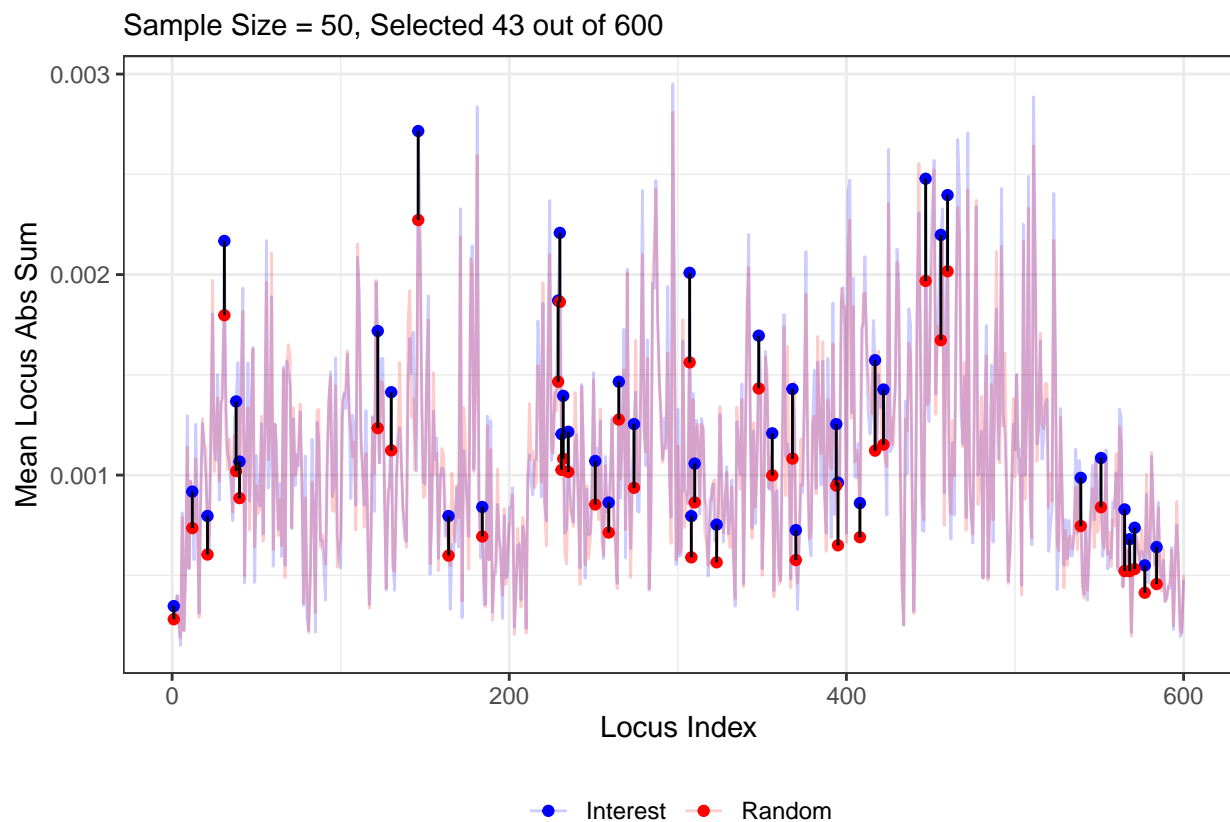
We performed one-sided t-test row-wise (only select positions with absolute sum of gradients larger than the random group). Since each row was tested only once, no multiple testing is present.

In the end, 43 out of 600 loci are identified as having significantly higher absolute sum of gradients than the random group.

We visualize these pairs, and notice that high importance points are not necessarily helpful in distinguishing instances, since the two curves are very similar in shape.

```
## Loaded interest_df from existing CSV file.
```

```
## Loaded random_df from existing CSV file.
```



We further explore the constitution of those important features. We only extracted the indices, and we compare them with one of the instances and extract the AA position of those points.

At the moment, no noticeable difference in GC content or pattern in AA is detected.

We noticed some consecutive pairs – loci that are next to each other and are all significant in test. They are extracted for possible study.

```
## [1] "ACGT content of selected features:"
```

```
## selected_trip  
## A C G T  
## 35 30 37 27
```

```

## [1] "Table of corresponding amino acids of selected features:"

## selected_key
## * A D F G H I K L N P Q R S T V W
## 3 2 4 1 2 1 2 1 2 5 3 2 4 5 2 3 1

##      position      row_mean  random_mean      p_value significance AApos trip key
## 1         38 0.0013673686 0.0010197466 0.003189344          *    13  CAA  Q
## 2         40 0.0010674413 0.0008851177 0.007368128          *    14  GCG  A
## 3        229 0.0018704395 0.0014656836 0.005221331          *    77  TCC  S
## 4        230 0.0022079518 0.0018635523 0.042234082          *    77  TCC  S
## 5        231 0.0012042120 0.0010258784 0.038524771          *    77  TCC  S
## 6        232 0.0013952660 0.0010816359 0.015697543          *    78  TGG  W
## 7        235 0.0012161013 0.0010156220 0.039433041          *    79  GAT  D
## 8        307 0.0020086681 0.0015621075 0.028586254          *   103  AAC  N
## 9        308 0.0007958138 0.0005896349 0.012281281          *   103  AAC  N
## 10       310 0.0010573849 0.0008636355 0.028824827          *   104  GAT  D
## 11       368 0.0014292921 0.0010820546 0.010784528          *   123  TGA  *
## 12       370 0.0007253654 0.0005763165 0.035002534          *   124  CCG  P
## 13       568 0.0006802126 0.0005208035 0.010131517          *   190  TCG  S
## 14       571 0.0007374910 0.0005313569 0.011778711          *   191  TGA  *

## [1] "ACGT content of selected, consecutive features:"

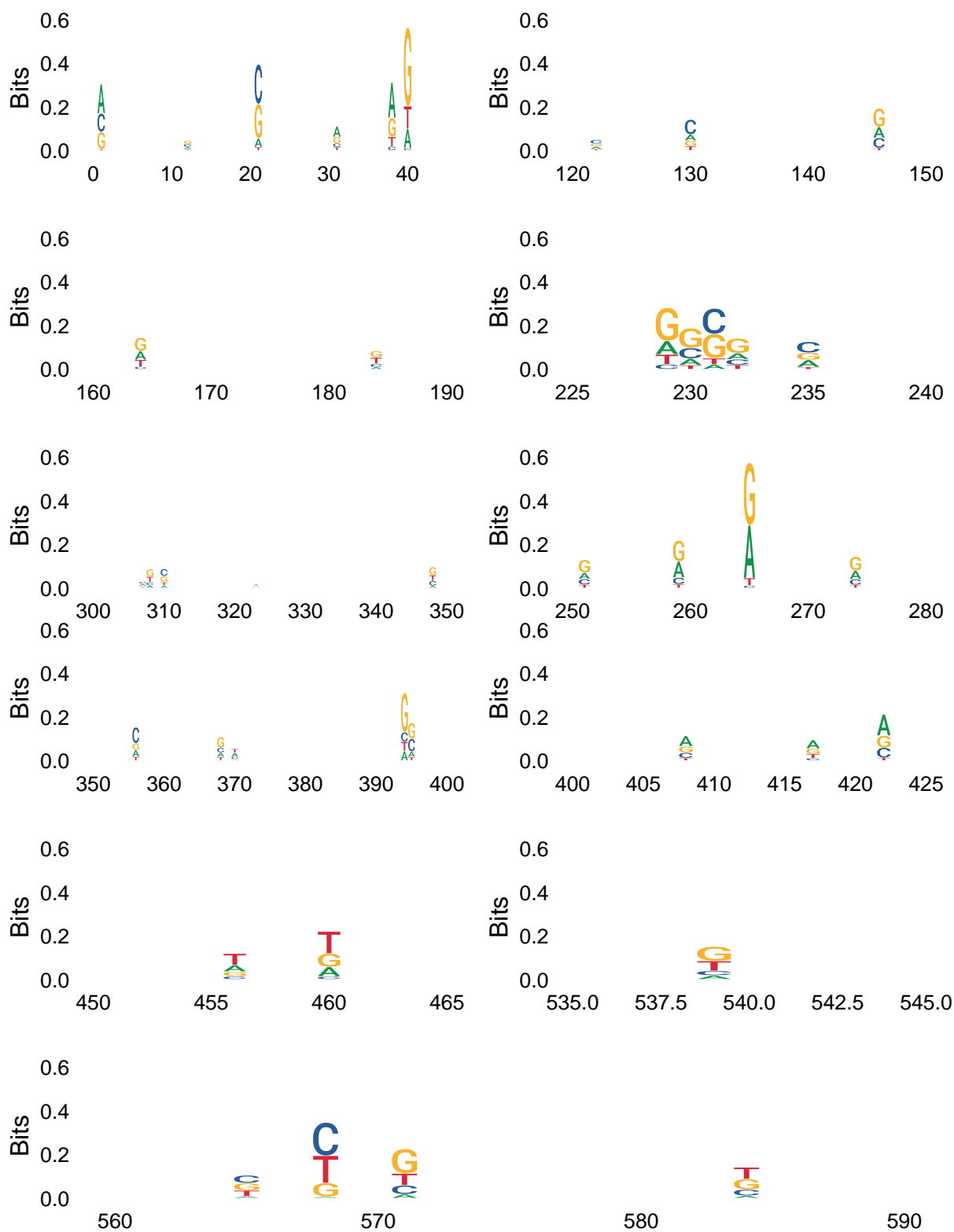
## consecutive_trip
##  A  C  G  T
## 10 13 10  9

## [1] "Table of corresponding amino acids of selected, consecutive features:"

##
## * A D N P Q S W
## 2 1 2 2 1 1 4 1

```

We have visualized the selected features in a **sequence logo plot**. Keep in mind that here, G and A are present with relatively high confidence.



Adversarial

We also tried adversarial instances. We replace the selected spots of the instance all with C, and the confidence for 16S rRNA drops from **0.906 to 0.176!** Also replacing them with T reduces the confidence to 0.20. Replacing with A and G is not effective.

We compare them with an exact same number of randomly selected positions and substitute them accordingly too.

We also tried replacing them with 0.25, and the confidence drops only to 0.7. There is also only 0.04 difference of confidence between the selected features and randomly sampled features.

```
## [1] "Original Prediction: "
```

```
##           [,1]      [,2]
## [1,] 0.906086 0.09391395
```

```
## [1] "Prediction after substituting important features with C: "
```

```
##           [,1]      [,2]
## [1,] 0.1767416 0.8232583
```

```
## [1] "Prediction after substituting same amount of random features with C: "
```

```
## [1] 0.2139295
```

```
## [1] "Prediction after substituting important features with A: "
```

```
##           [,1]      [,2]
## [1,] 0.8840113 0.1159886
```

```
## [1] "Prediction after substituting same amount of random features with A: "
```

```
## [1] 0.9280069
```

```
## [1] "Prediction after substituting important features with 0.25:"
```

```
##           [,1]      [,2]
## [1,] 0.7500898 0.2499101
```

```
## [1] "Prediction after substituting same amount of random features with 0.25:"
```

```
## [1] 0.7787492
```

We would like to study the drastic drop of prediction confidence when more C and T are introduced.

Input Reconstruction

We tried reconstructing the input sequence using a non-informative baseline, where A,C,G,T each has 0.1 for each position. We attempted 0.25, but the algorithm did not converge. Also choosing a bacteria sub-sequence leads to a non-converging result. This seems to be interesting.

This is done by iteratively doing:

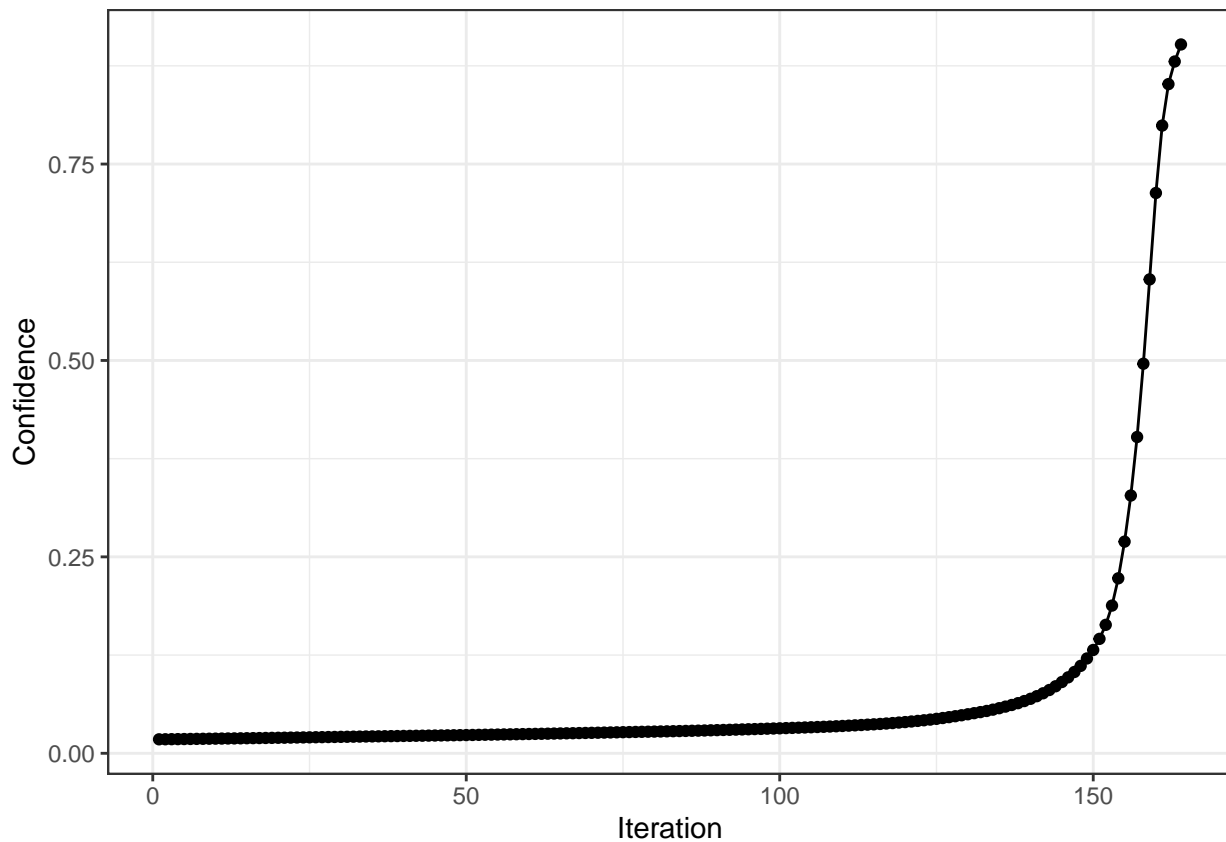
$$X^{(t+1)} = X^{(t)} + \epsilon \cdot IG(X^{(t)})$$

We set $\epsilon = 2$.

We plot the learning curve and the sequence logo (customed using final IG as y-axis) for a segment of the sequence.

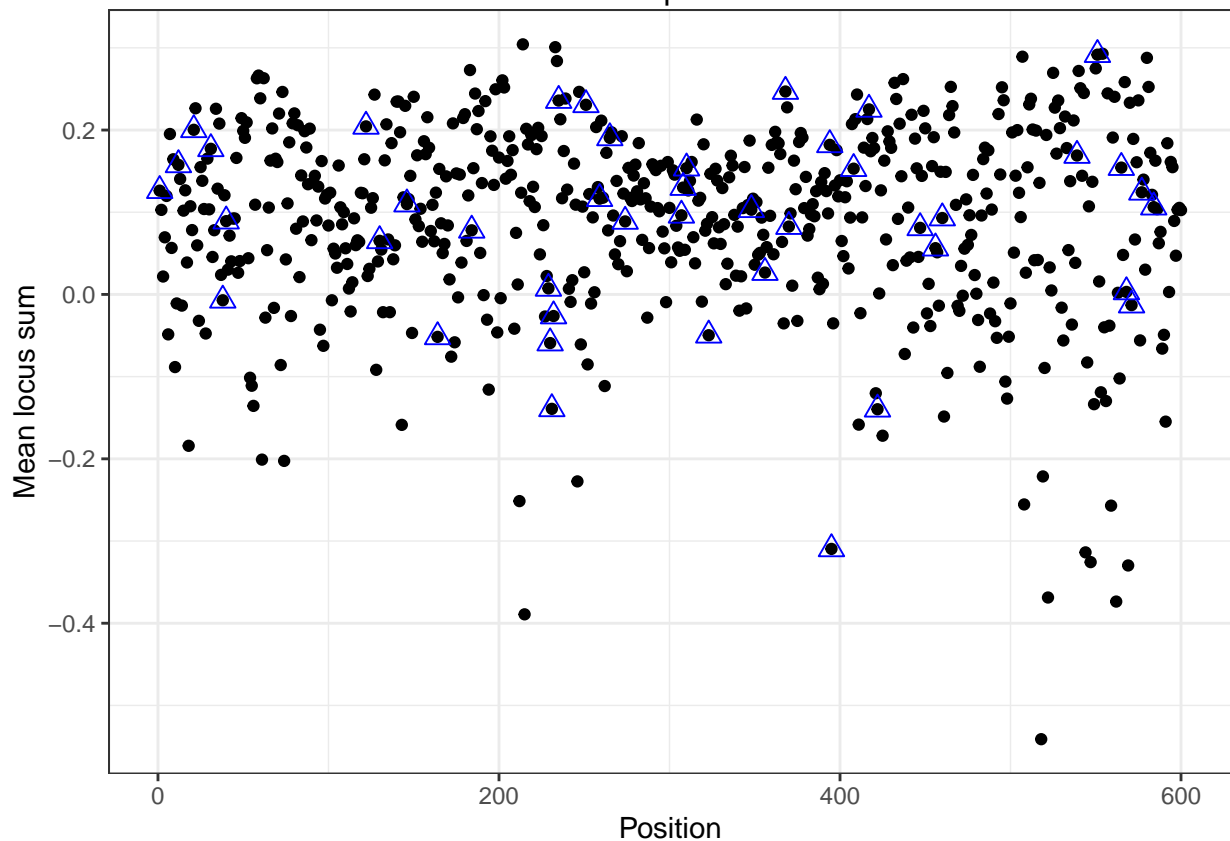
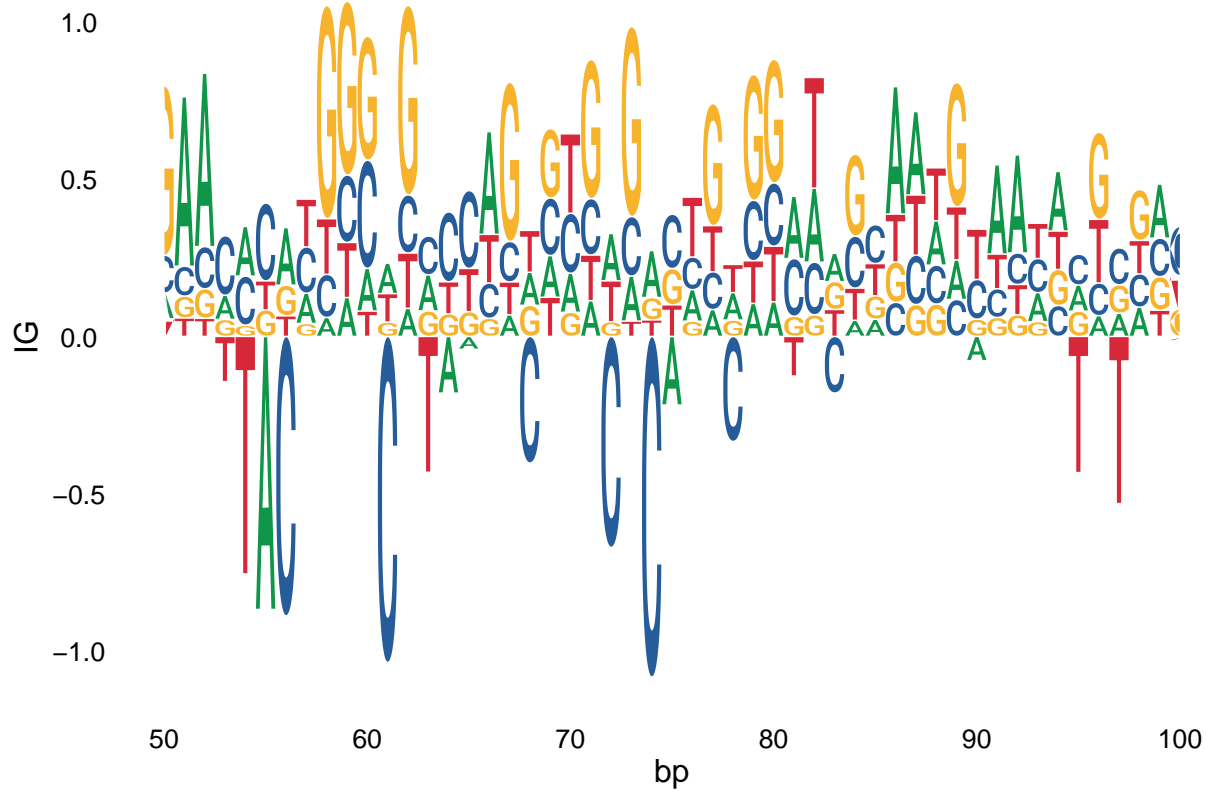
Interpretation: the achieved representation is the least required to be predicted as 16S rRNA gene with a confidence of 90%.

We also plotted the mean of each position with the previously selected important spots marked blue. The value here does not seem to correspond to the identified importance of the position.



```
## Scale for x is already present.
```

```
## Adding another scale for x, which will replace the existing scale.
```



Pay extra attention to the reconstructed sequence logo plot. In the negative region, C and T often have considerably negative weights, meaning they should not appear on this position if the prediction is 16S. This

could explain why the adversarial scenario where more Cs are included caused dramatic drop in confidence.

We also table the matching positions of the reconstructed sequence comparing the original one. Compared to selected features, we see the matching positions seem not to be totally away from the selected ones. Could this be of interest?

```
## [1] "Matched Reconstructed sequence and original sequence, AA position: "
```



```
## AAT ATA GGA GGG GGA AAT AGG AAC ATA GGT GGA GAT AAT GGC TTA CTG GTC AAT GAT AGT
##   1   9  17  20  27  45  53  63  66  68  69  79  80  89  94  97  98 103 104 116
## AGC GGG GGC AAA AAC GGA TAA GAC GAC GGA GGG AGA GGT GTA GGG
## 122 126 129 133 137 140 150 152 162 165 171 176 184 193 194
```



```
## [1] "Selected features, AA position: "
```



```
## [1]   1   4   7  11  13  14  41  44  49  55  62  77  77  77  78  79  84  87  89
## [20]  92 103 103 104 108 116 119 123 124 132 132 136 139 141 149 152 154 180 184
## [39] 189 190 191 193 195
```

Discussion

Main Point: IML in Genomics: What comes after non-informative local interpretations? The Interpretation of Interpretations

1. Due to lack of clear patterns in most real-world cases, it seems IG alone, even if correctly calculated, can be very much not informative. Instead of keep improving its theory minimally, it seems more reasonable to come up with methods tailored to genomics that can best utilize those data and provide interpretable or meaningful results. For example, the PCSC score could be a good measurement in explaining the reliability of the model in capturing synonymous codons.
2. How to work with the selected features? Are they biologically significant? Are they useful in distinguishing instances? (for which we need to adjust the model) These are questions that need to be answered in the future.
3. I personally find the convergence of feature reconstruction fascinating. For some reasonable targets, e.g. a bacterial sequence, it is not possible to transform it to a possible 16S simply using IG outputs.
4. The choice of baseline can be further studied. Choosing a real sequence from the other category can include too much artifact than truth.