Real Data: 16s rRNA

Yichen Han

2024-08-11

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]]$AUPRC
## NULL
```

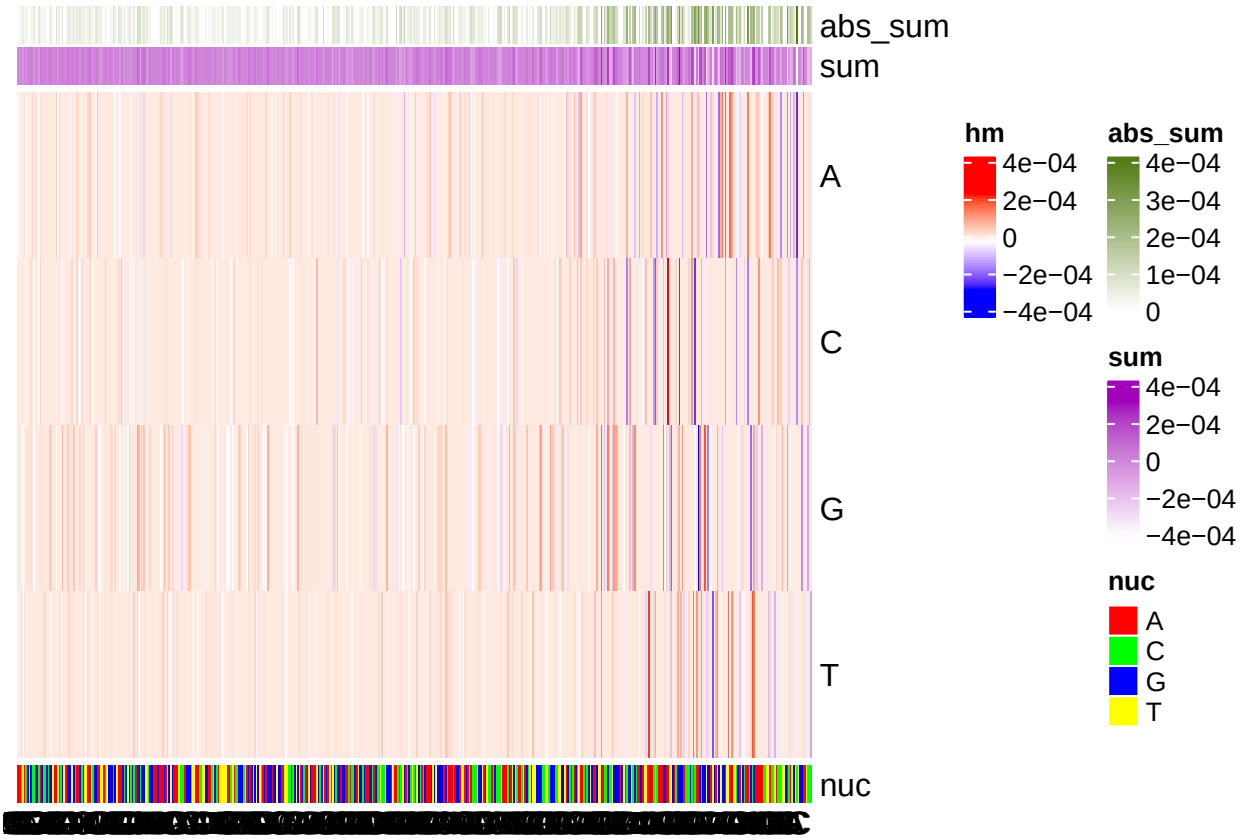
We see that the model has very high accuracy.

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.

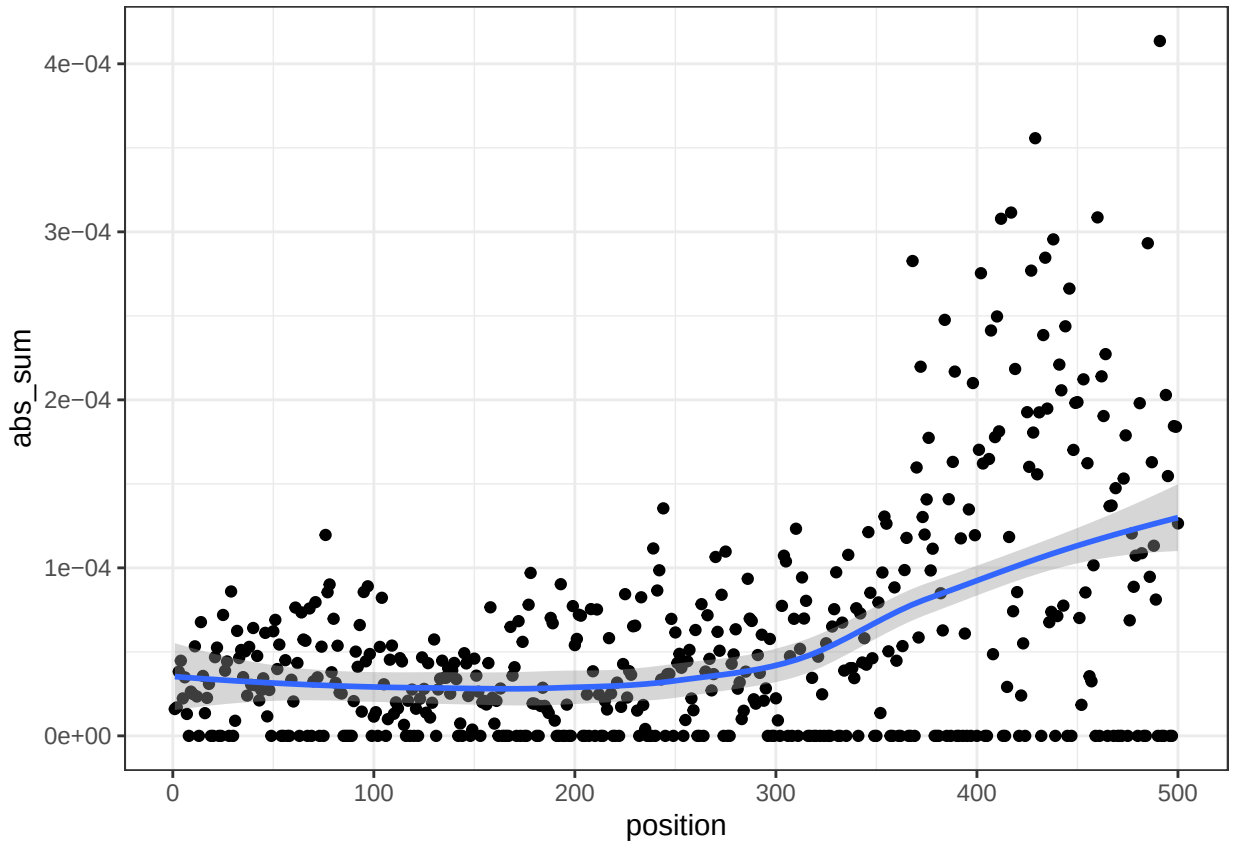
We then perform modified IG.

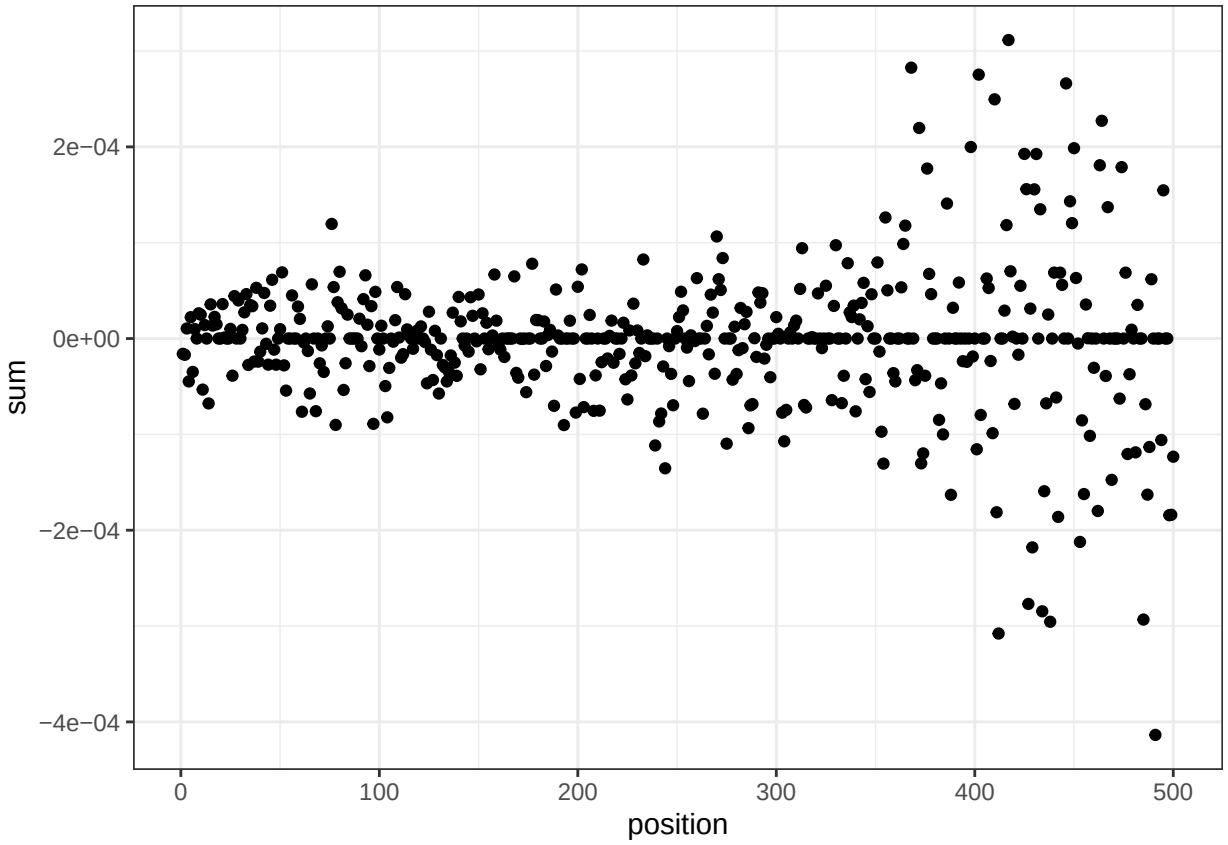
```
##           [,1]        [,2]
## [1,] 0.9987847 0.001215239
```

```
## [[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:

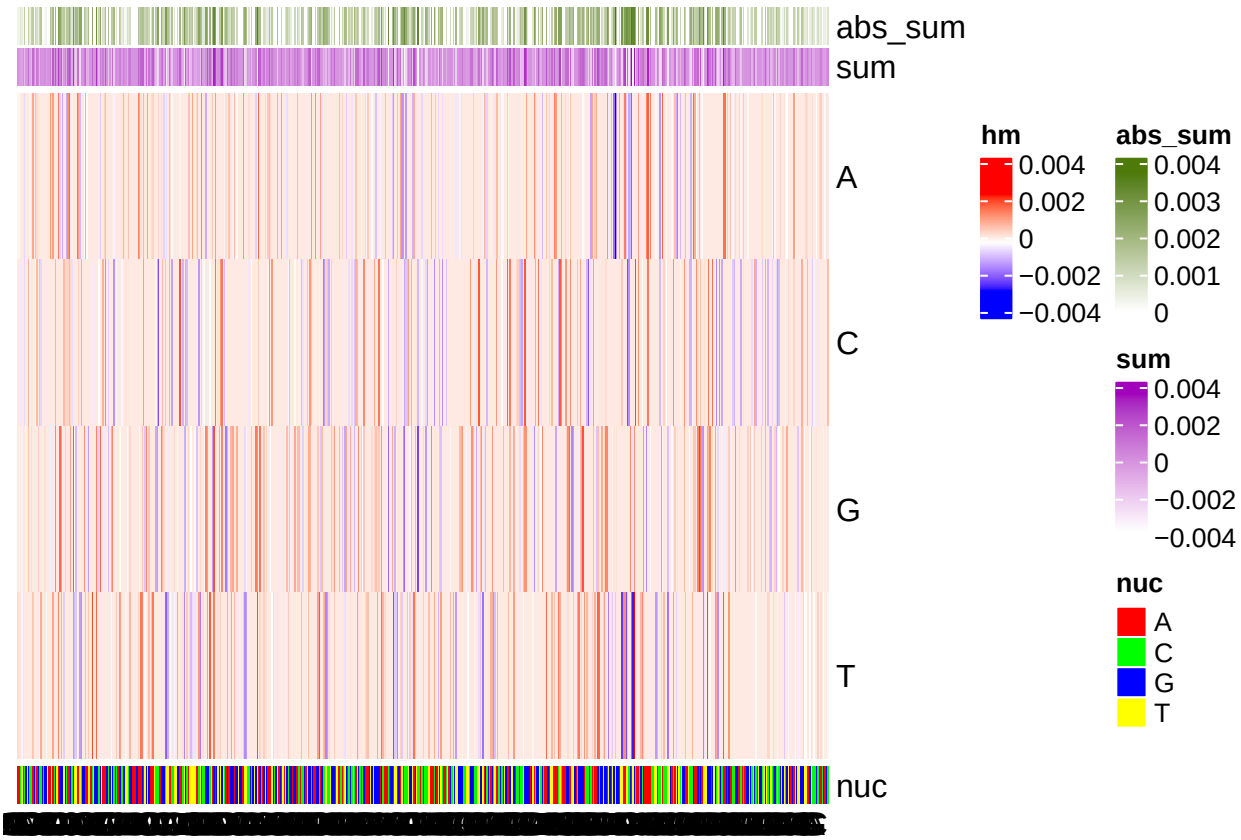
```
## Using checkpoint checkpoints/16S_vs_bacteria_full_2/Ep.008-val_loss0.13-val_acc0.991.hdf5

## [[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
```

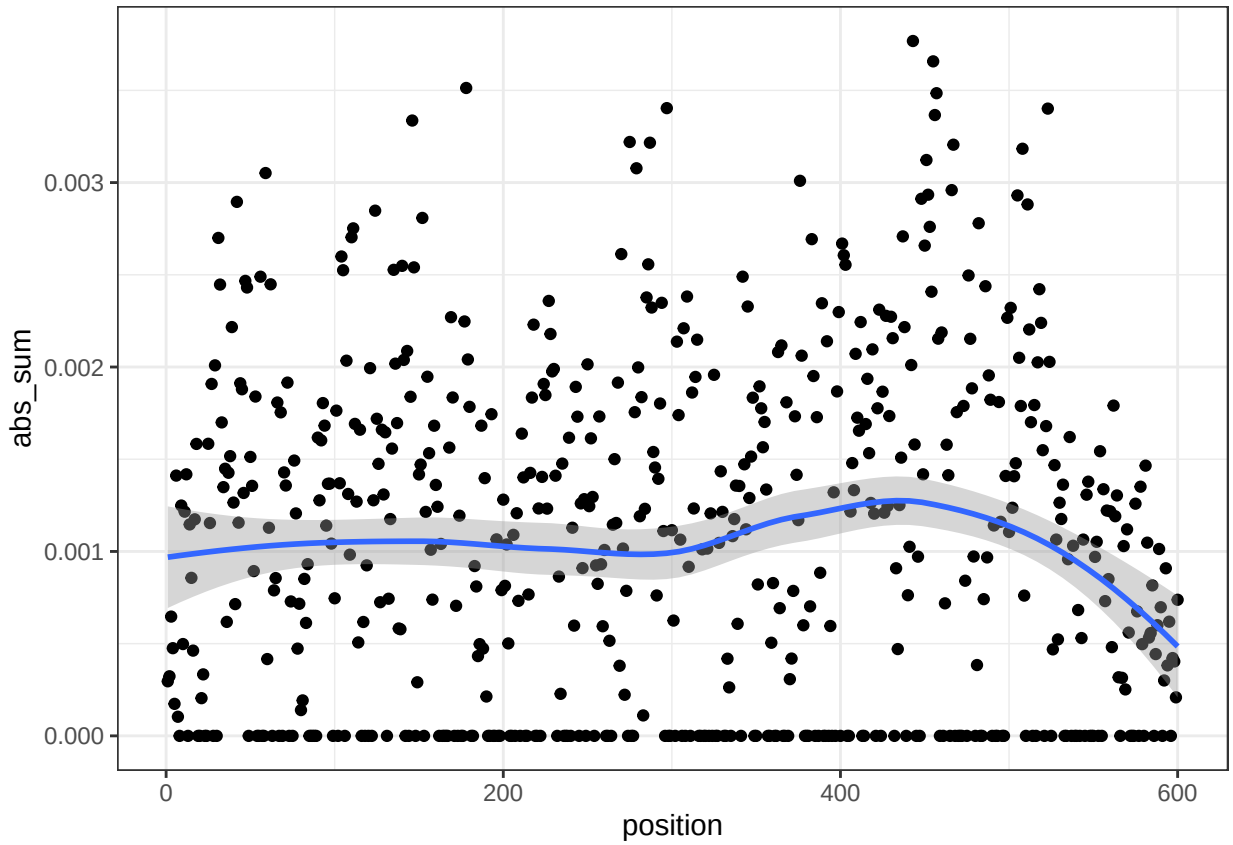
It still has a very high accuracy. We evaluate it again using modified IG.

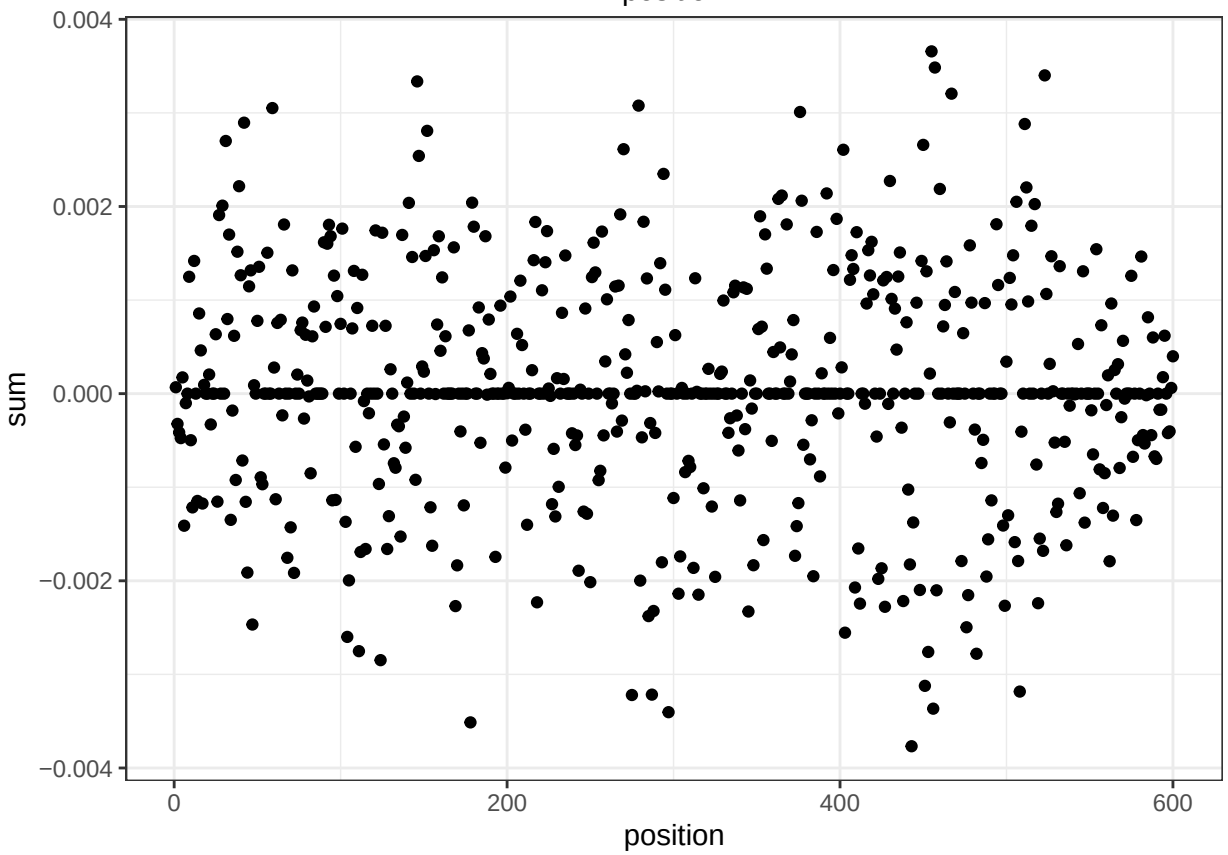
```
##          [,1]       [,2]
## [1,] 0.906086 0.09391395
```

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




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



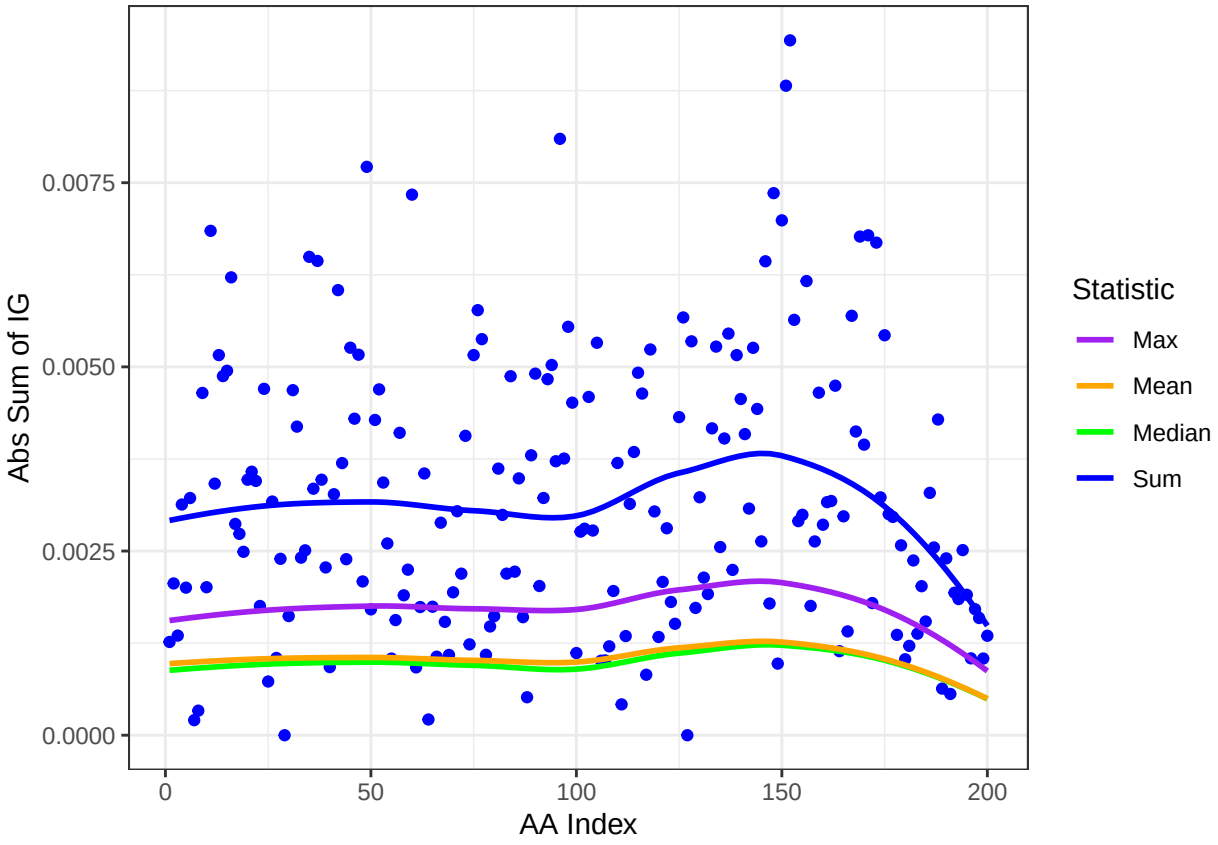


This time, the importance is scattered across positions in a more complex way, which reflects the nature of

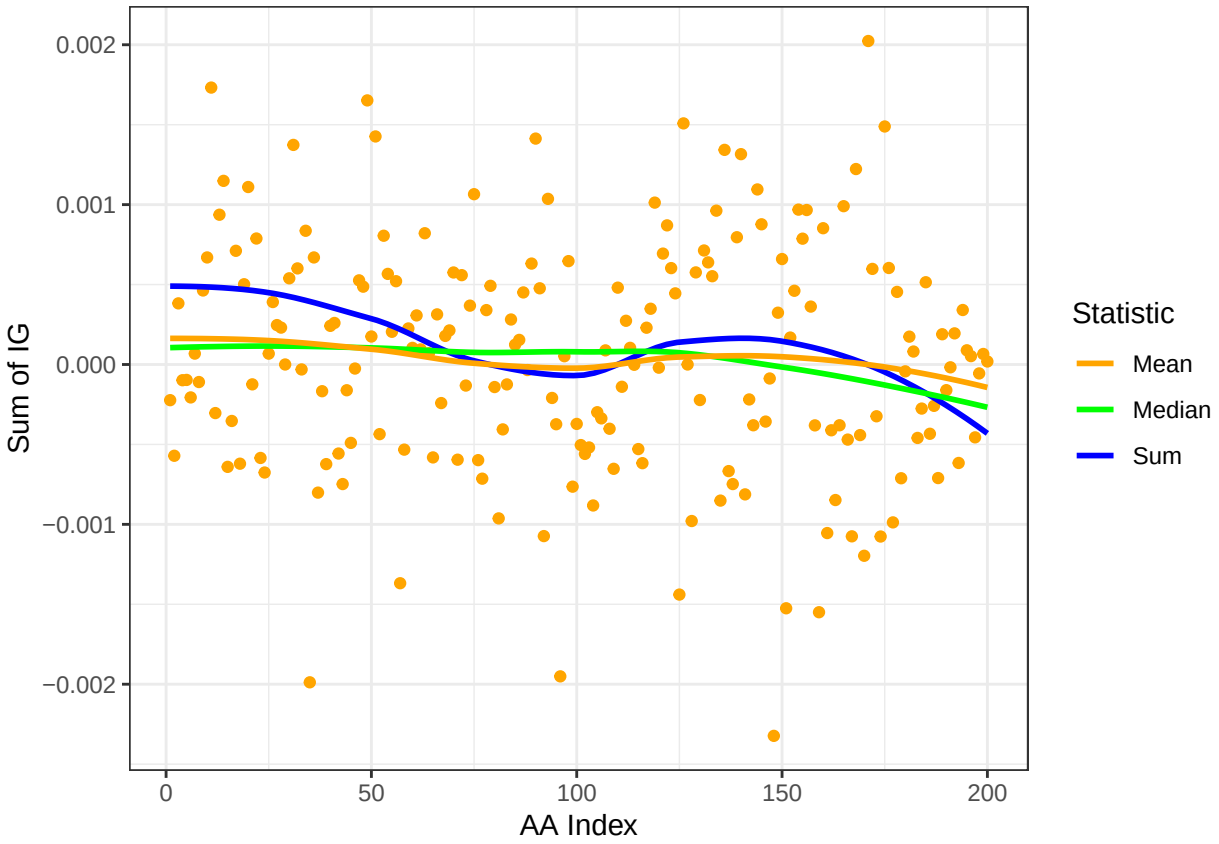
genetic data.

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

```
## `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'
```

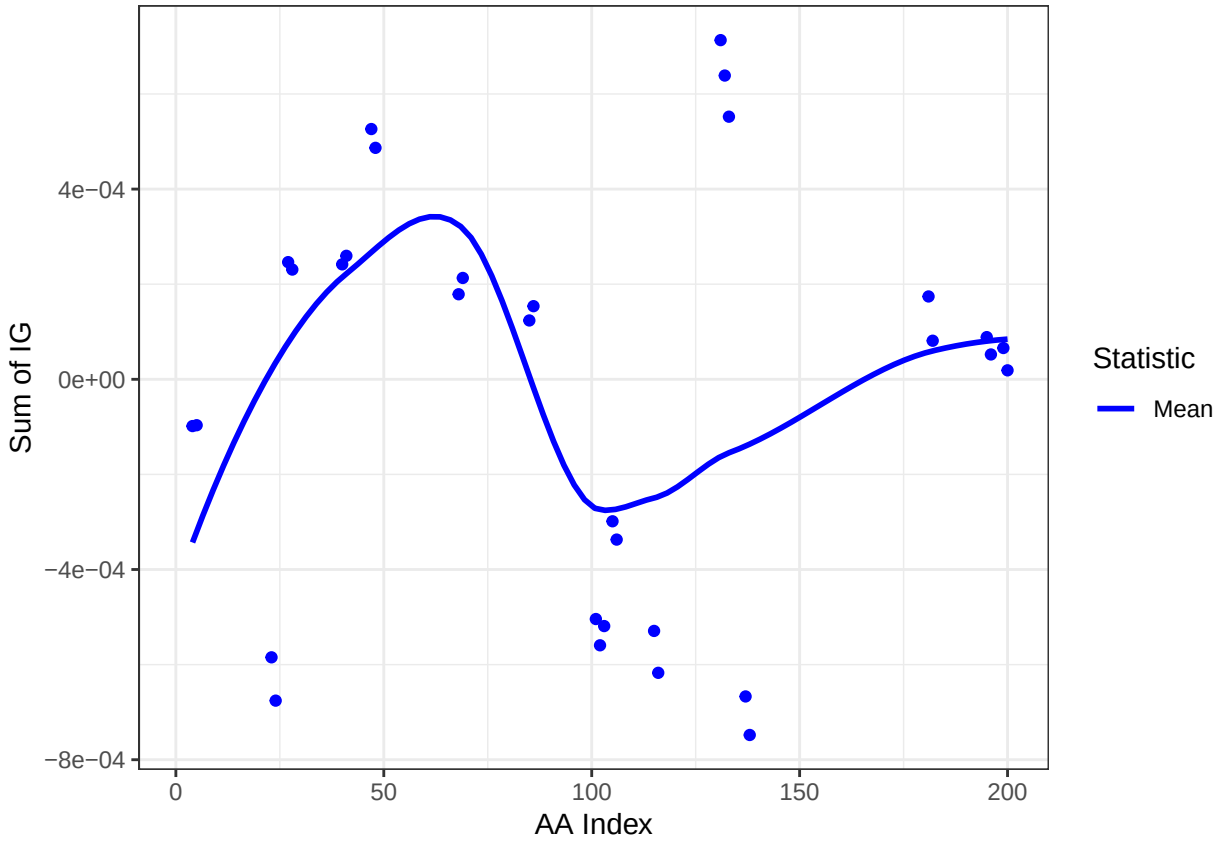


Before and after information compression, there are some obvious clustering on the graph. How important / interesting they are is unknown to us, but we extract them here for possible use.

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

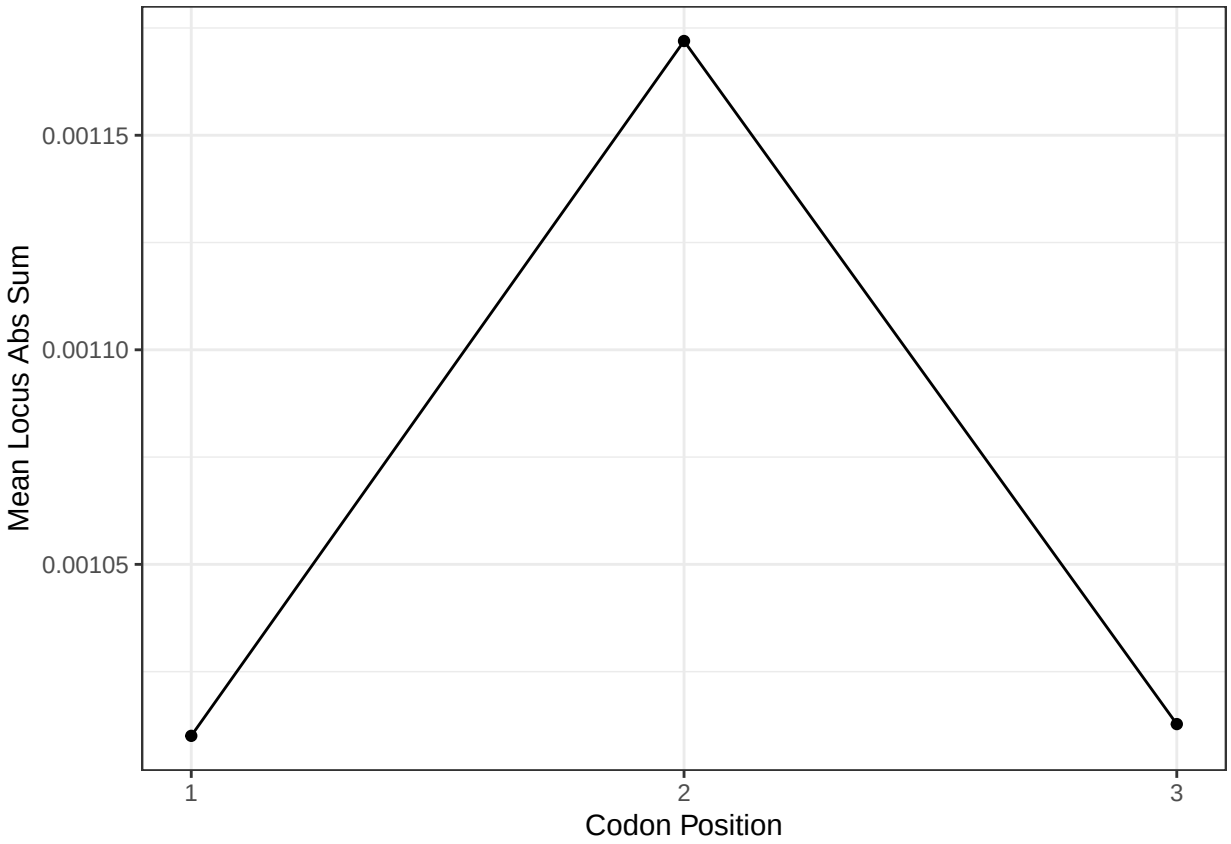
Table 1: Possible clustering of Importance score

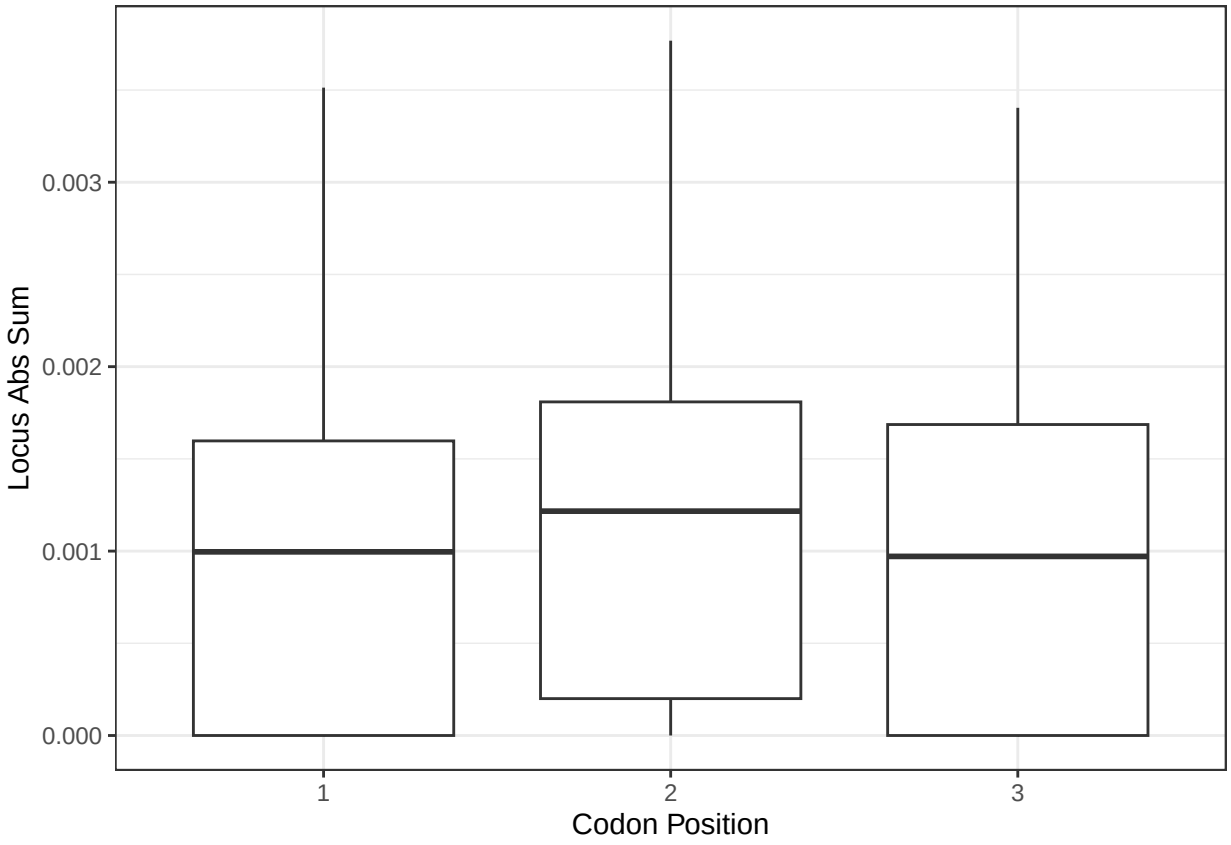
cluster	score	seq	aa
3	-0.0001950	CCAGCA	PA
21	-0.0012604	GCGTGC	AC
24	0.0004771	GGTTCG	GS
36	0.0005014	TGGAAC	WN
42	0.0010130	CGGGCT	RA
62	0.0003919	GGAGGA	GG
78	0.0002775	GCACGA	AR
93	-0.0015825	GCCCTAAAC	ALN
95	-0.0006357	GTCAAC	VN
104	-0.0011464	GGTAGC	GS
119	0.0019041	AAGATTAAA	KIK
123	-0.0014149	AATTGA	N*
166	0.0002553	GTGCTG	VL
179	0.0001409	TTAAGT	LS
182	0.0000846	ACGAGC	TS



We further studied wobleness. 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 wobleness is not always detected.

Here is an example of existing wobbleness.


Mean Locus Abs Sum
0.00115
0.00110
0.00105
1
2
3
Codon Position

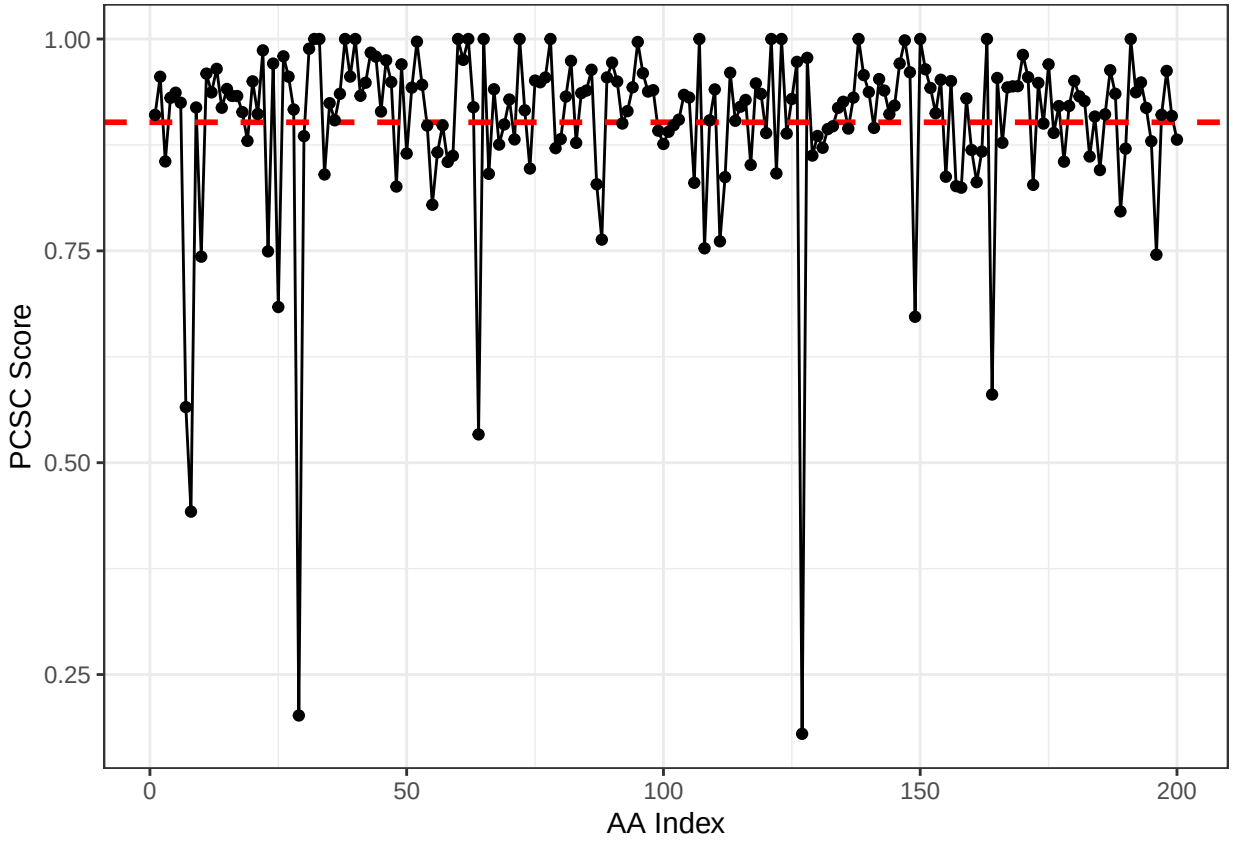

Locus Abs Sum
0.003
0.002
0.001
0.000
1
2
3
Codon Position

```
##
## 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
```

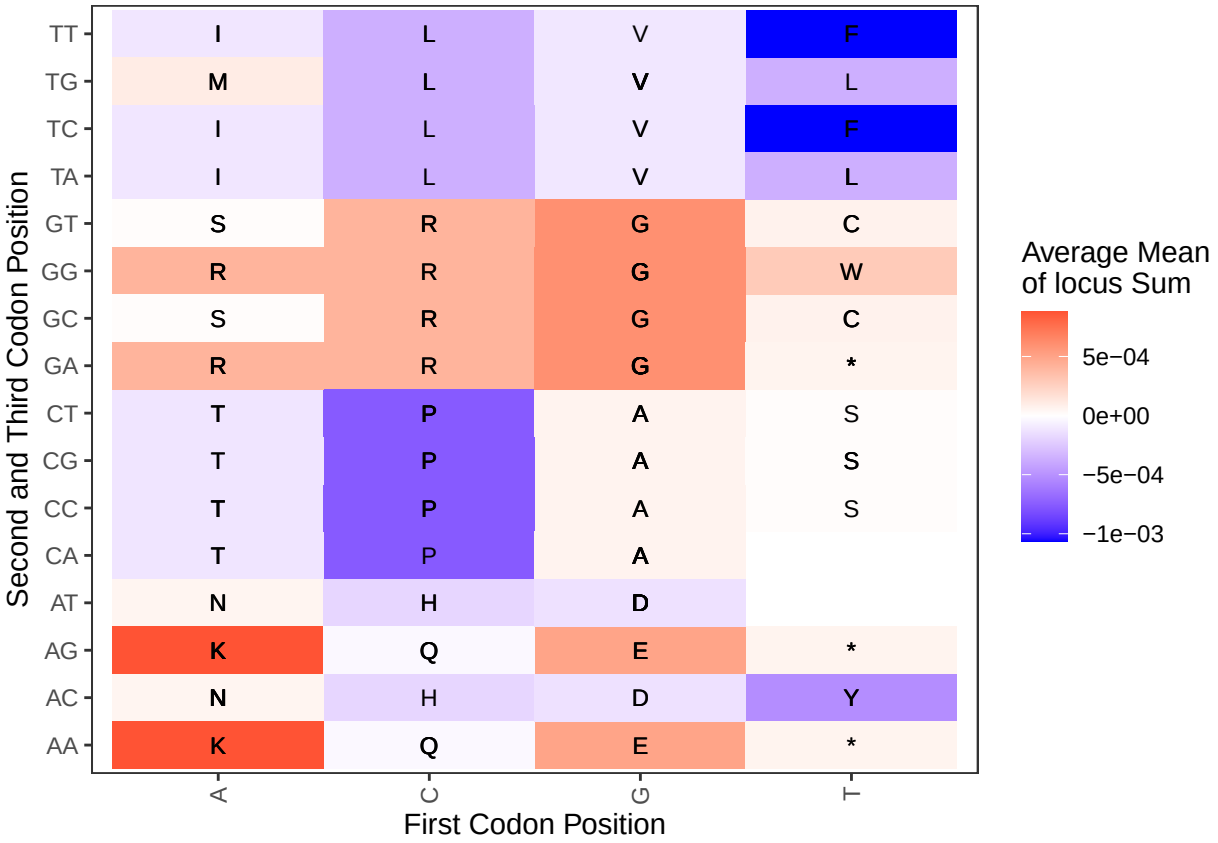
Consistency

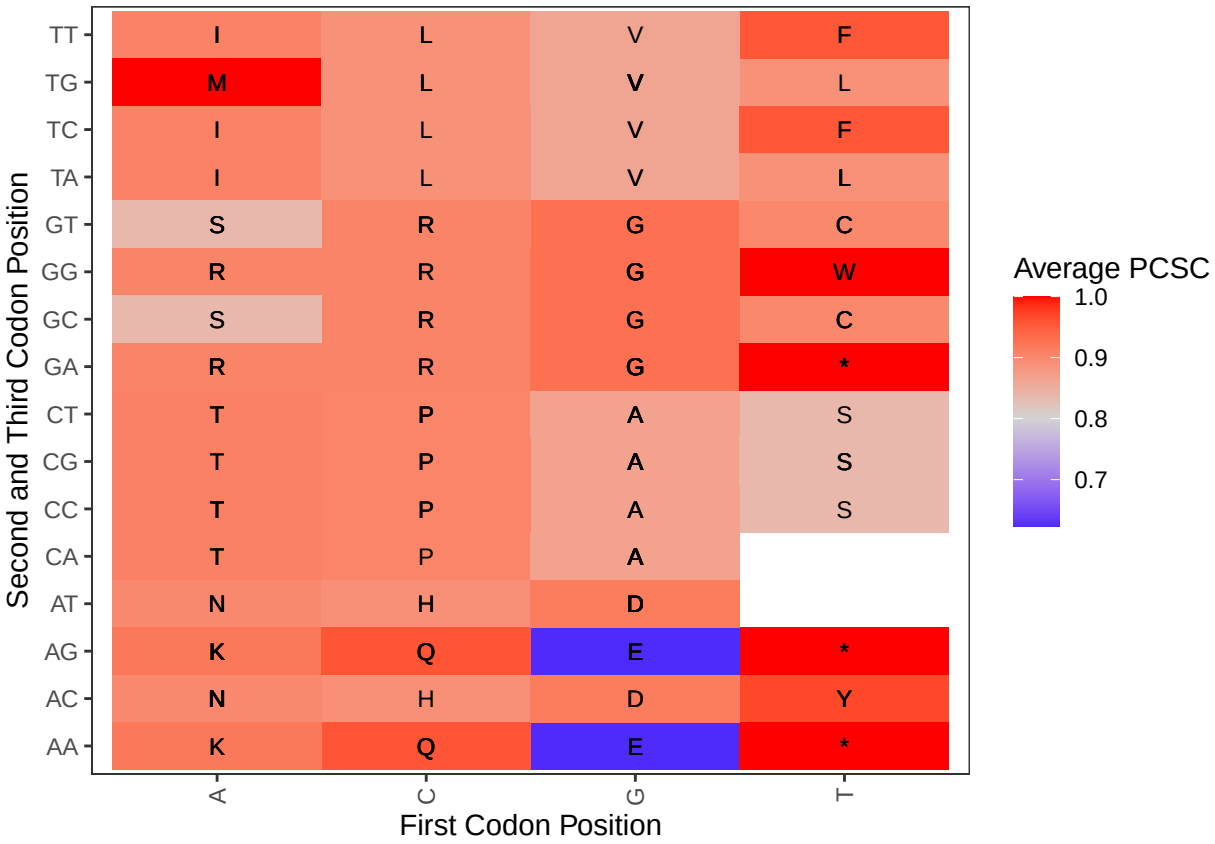
We then calculated PCSC score on our data. With an average score at 0.9, we conclude that the model was able to capture synonymous codons and treats them consistently.

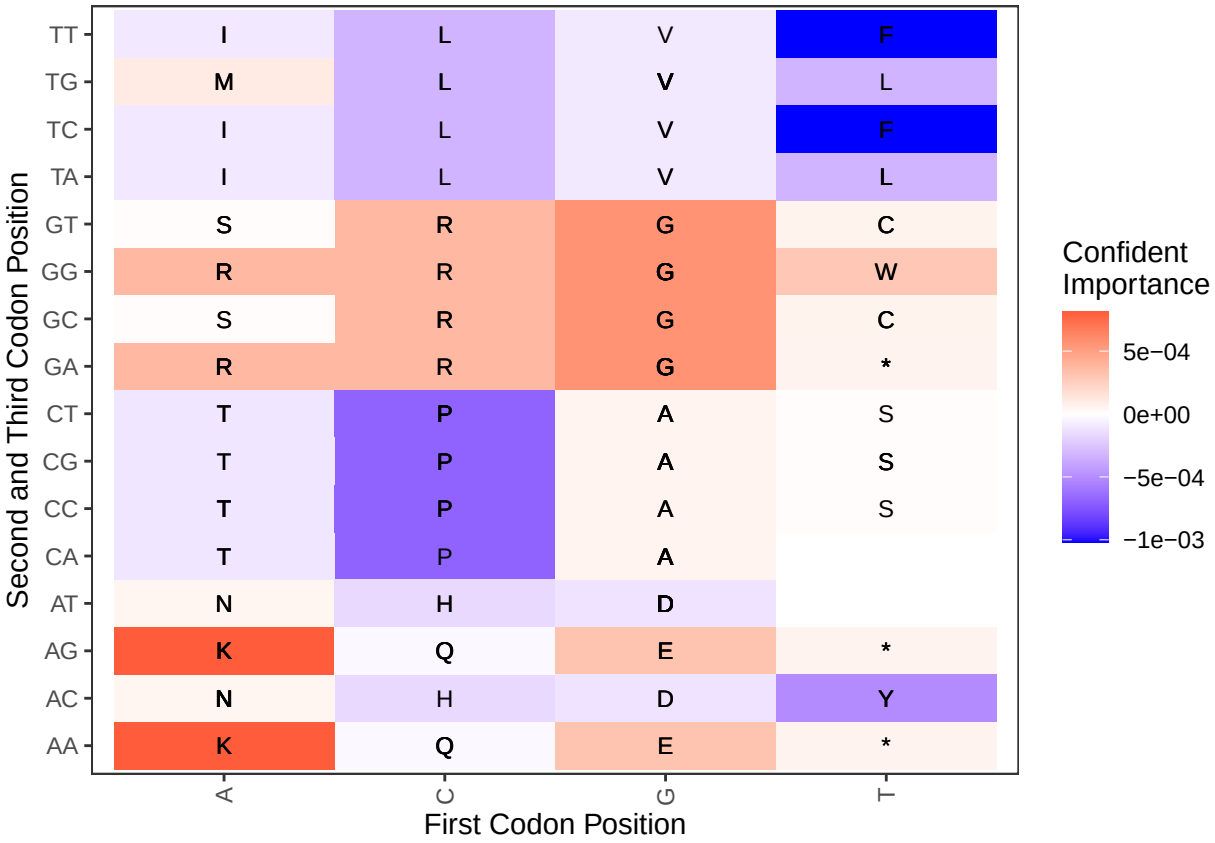
```
## Loaded result_df from existing CSV file.
```



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.



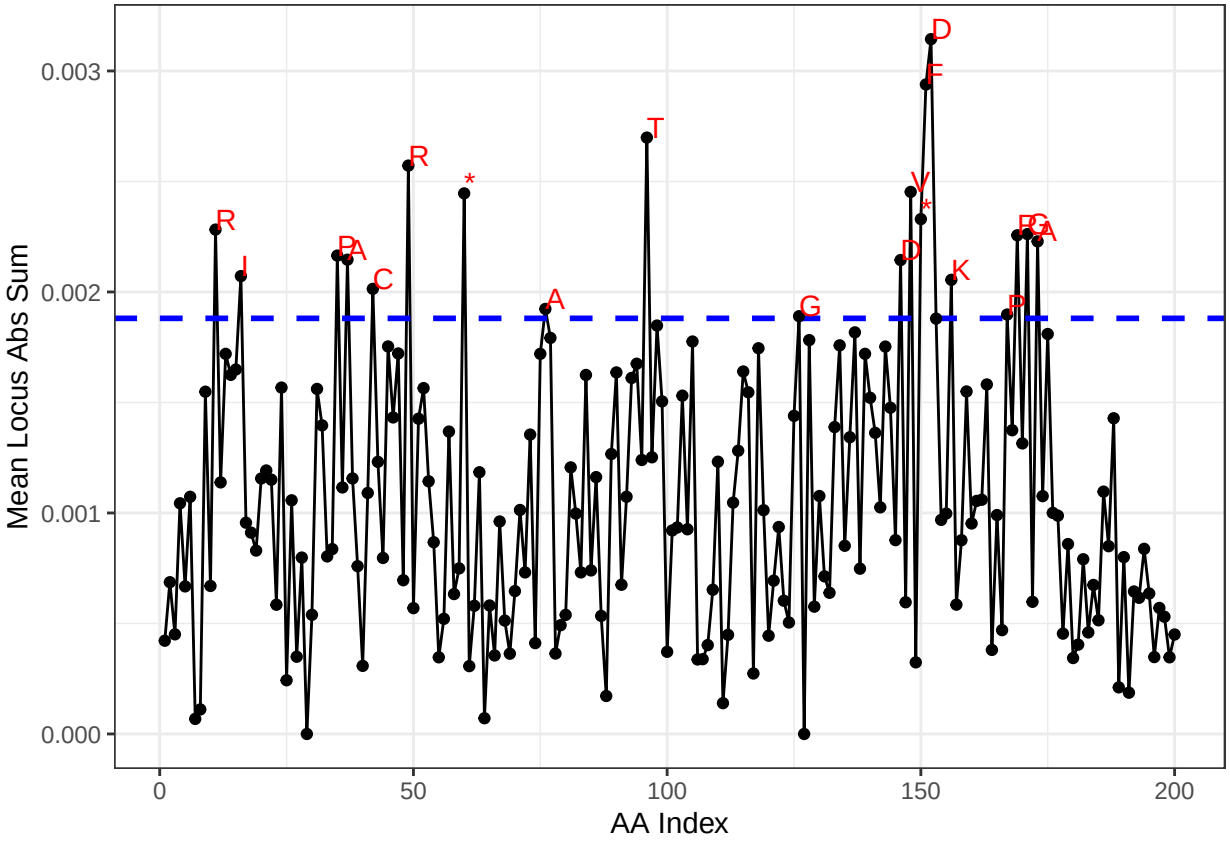

Second and Third Codon Position
TT I L V F
TG M L V L
TC I L V F
TA I L V L
GT S R G C
GG R R G W
GC S R G C
GA R R G *
CT T P A S
CG T P A S
CC T P A S
CA T P A
AT N H D
AG K Q E *
AC N H D Y
AA K Q E *
A C G T
First Codon Position
Average PCSC
1.0
0.9
0.8
0.7


Second and Third Codon Position
TT I L V F
TG M L V L
TC I L V F
TA I L V L
GT S R G C
GG R R G W
GC S R G C
GA R R G *
CT T P A S
CG T P A S
CC T P A S
CA T P A
AT N H D
AG K Q E *
AC N H D Y
AA K Q E *
A C G T
First Codon Position
Confident Importance
5e−04
0e+00
−5e−04
−1e−03

Feature Selection

We first visualize the average absolute feature importance by position based on our instance.

Points above the 90%-quantile are annotated with their corresponding amino acid.



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.

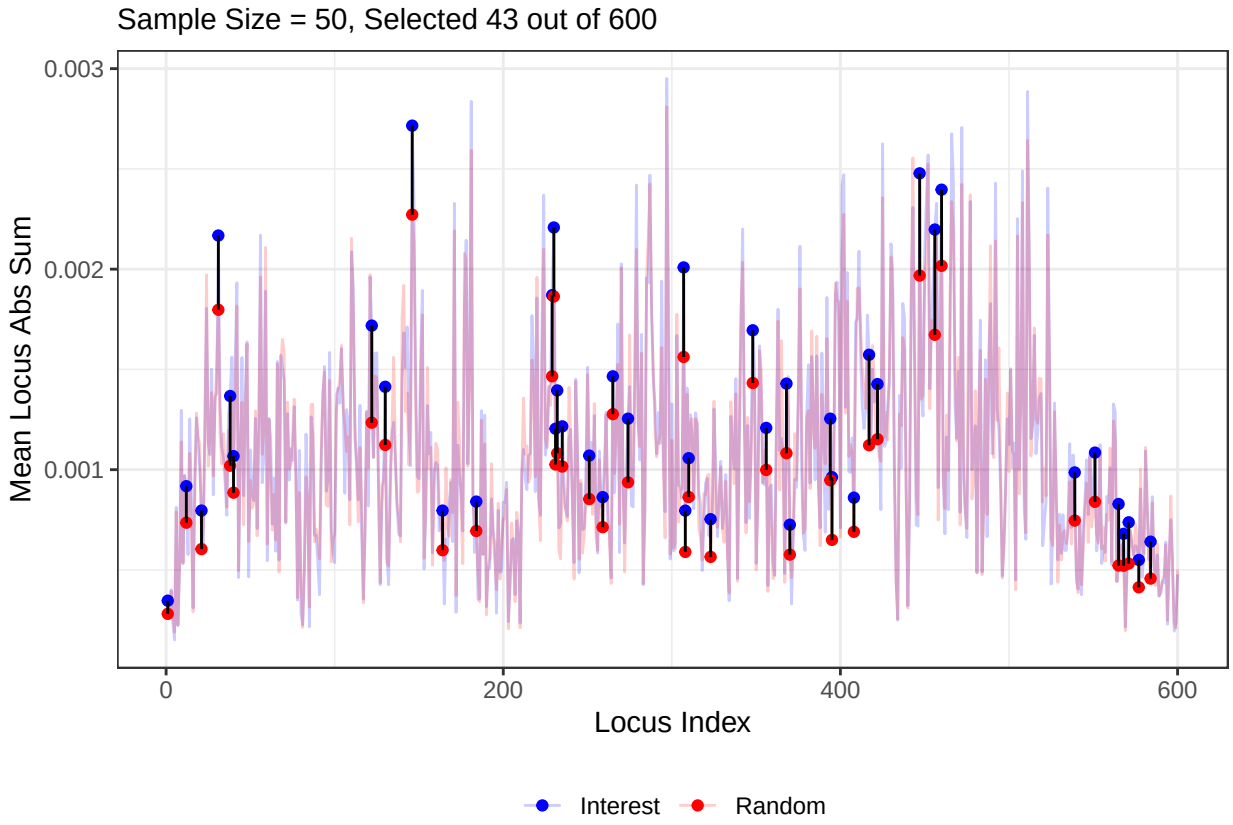
We performed t-test row-wise. Since each row was tested only once, no multiple testing is present.

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 consitution 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.

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

```
## 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   *
```

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

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

Discussion

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. The capturing of wobbleness seems to be an interesting topic. It is not clear how it can be used in practice, but it is a good indicator of the model's ability to capture the codon structure. If it can be proved to be universally present in correctly trained models, that would be very ideal.

3. 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.