Real Data: 16s rRNA

Yichen Han

2024-08-10

```
# input data are divided into train, validation sets
path_16S_train <- file.path("16s/train")
path_16S_validation <- file.path("16s/validation")
path_bacteria_train <- file.path("bacteria/train")
path_bacteria_validation <- file.path("bacteria/validation")

checkpoint_path <- file.path("checkpoints")
tensorboard.log <- file.path("tensorboard")

dir_path <- file.path("outputs")
#unlink(paste0(checkpoint_path, "/16S_vs_bacteria_checkpoints/*"))
#unlink(paste0(checkpoint_path, "/lm_16S_target_middle_lstm/*"))
#if (!dir.exists(checkpoint_path)) dir.create(checkpoint_path)
#if (!dir.exists(tensorboard.log)) dir.create(tensorboard.log)
#if (!dir.exists(dir_path)) dir.create(dir_path)
```

```
# load model checkpoint with best validation accuracy
run_name <- "16S_vs_bacteria_checkpoints"
model <- load_cp(paste0(checkpoint_path, "/", run_name),
                 cp_filter = "last_ep")
```

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

```
# evaluate model
path_input <- c(path_16S_validation, path_bacteria_validation)
pred_df_path <- tempfile(fileext = ".rds")
eval_model <- evaluate_model(path_input = path_input,
  model = model,
  batch_size = 250,
  step = 500,
  vocabulary_label = c("16s", "bacteria"),
  number_batches = 10,
  mode = "label_folder",
  verbose = FALSE,
  auc = TRUE,
  proportion_per_seq = 0.98,
  max_samples = 500,
  path_pred_list = pred_df_path)

eval_model
```

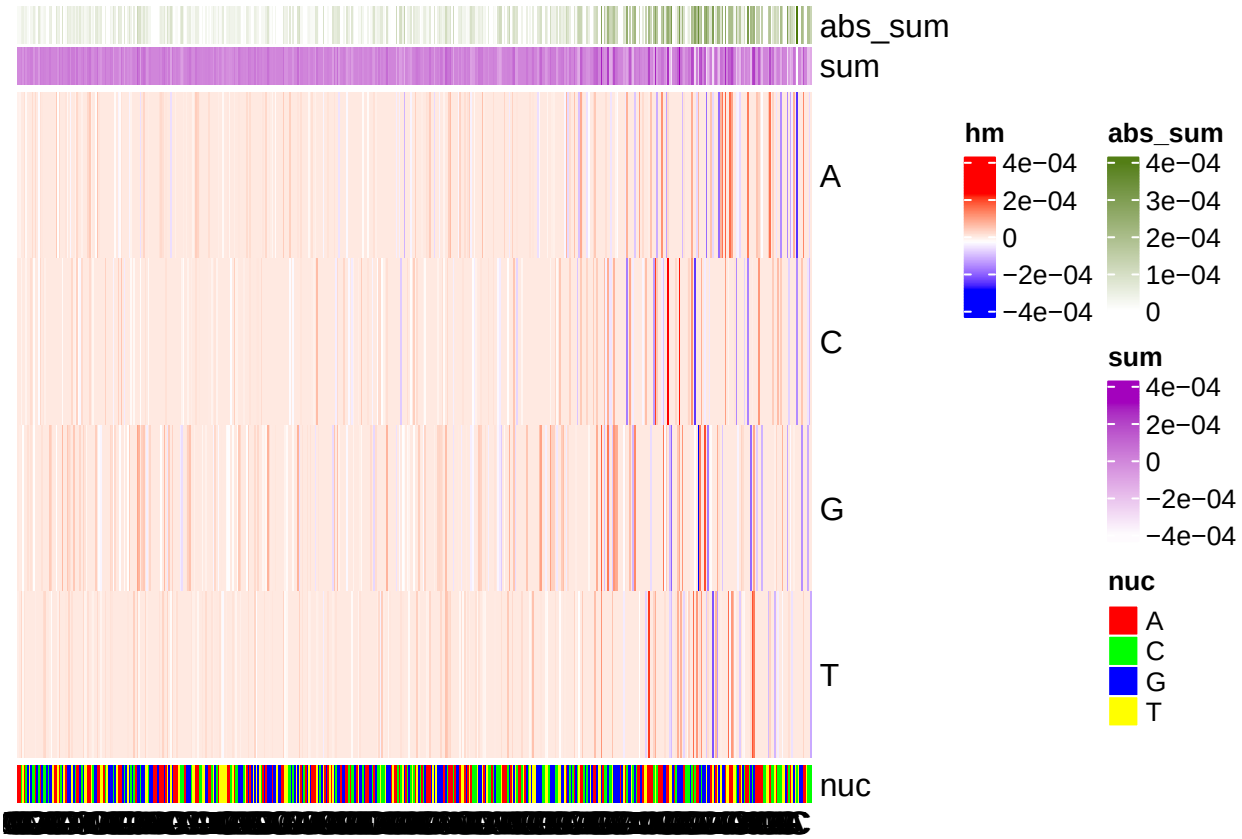
```
## [[1]]
## [[1]]$confusion_matrix
##           Truth
## Prediction  16s bacteria
##   16s      1250        1
##   bacteria    0     1249
##
## [[1]]$accuracy
## [1] 0.9996
##
## [[1]]$categorical_crossentropy_loss
## [1] 0.004219269
##
## [[1]]$AUC
## [1] 1
##
## [[1]]$AUPRC
## NULL
```

```
# instance is a randomly chosen fasta file from 16s validation
instance <- microseq::readFasta('16s/validation/GCF_001986655.1_ASM198665v1_genomic.16s.fasta.fasta')$S
instance_sub <- substr(instance, 499, 998)
onehot_instance <-  seq_encoding_label(char_sequence = instance_sub,
                                       maxlen = 500,
                                       start_ind = 1,
                                       vocabulary = c("A", "C", "G", "T"))
baseline <- microseq::readFasta('bacteria/validation/GCF_002895085.1_ASM289508v1_genomic.fasta')$Sequen
baseline_sub <- substr(baseline, 499, 998)
onehot_baseline <- seq_encoding_label(char_sequence = baseline_sub,
                                       maxlen = 500,
                                       start_ind = 1,
                                       vocabulary = c("A", "C", "G", "T"))
pred <- predict(model, onehot_instance, verbose = 0)
pred
```

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

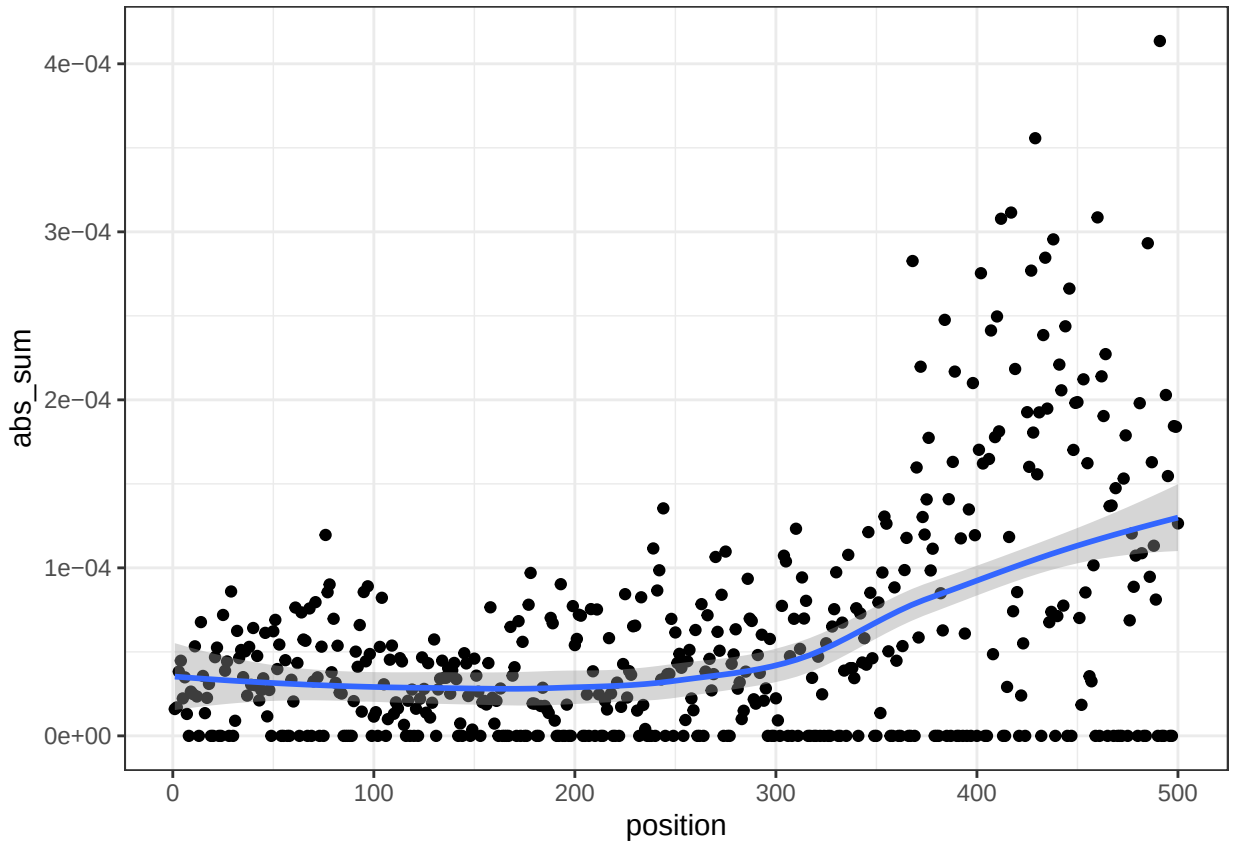
```
igs <- ig_modified(
  input_seq = onehot_instance,
  baseline_type = "modify",
  baseline_onehot = onehot_baseline,
  target_class_idx = 1,
  model = model,
  num_baseline_repeats = 1)
heatmaps_integrated_grad(integrated_grads = igs,
                         input_seq = onehot_instance)
```

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

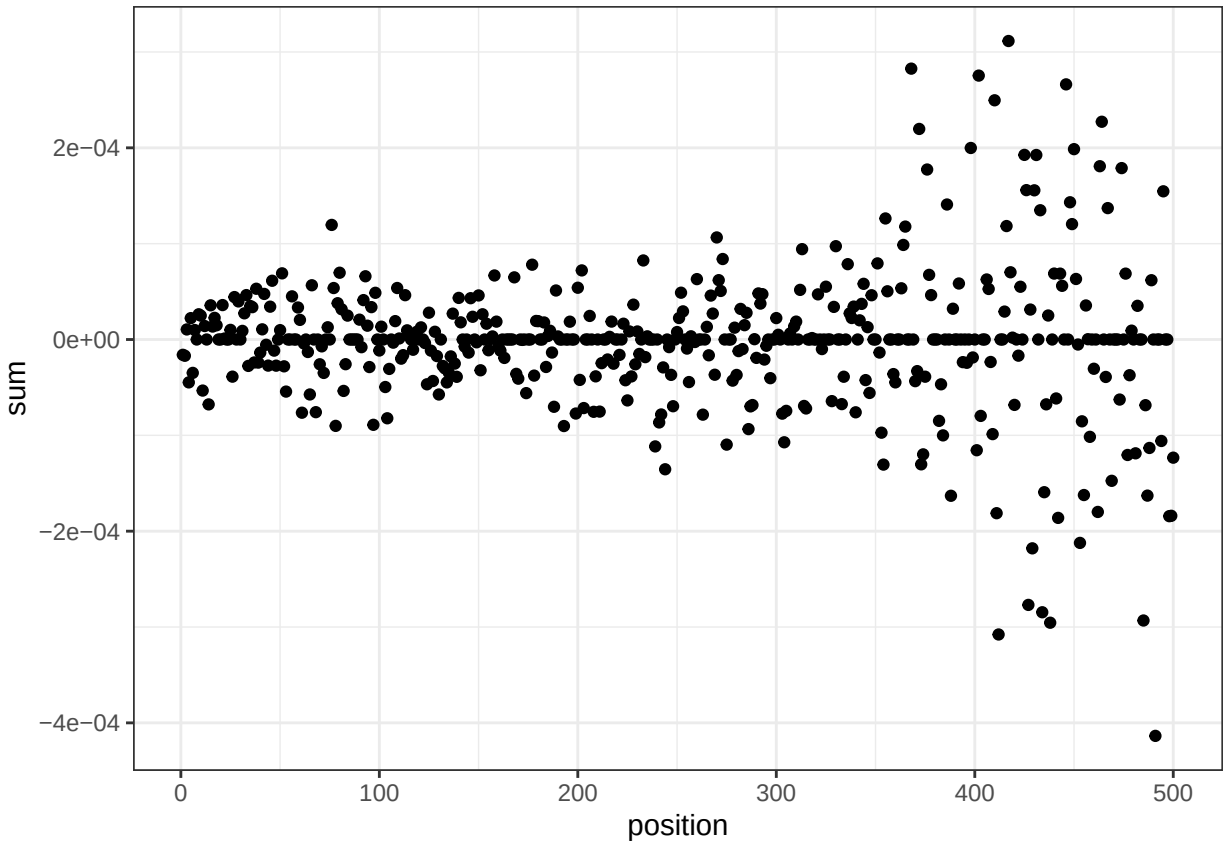



```
sum <- rowSums(as.array(igs))
abs_sum <- rowSums(abs(as.array(igs)))
df <- data.frame(abs_sum = abs_sum, sum=sum, position = 1 : 500)

ggplot(df, aes(x = position, y = abs_sum)) + geom_point() + geom_smooth() + theme_bw()
```

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




```
ggplot(df, aes(x = position, y = sum)) + geom_point() + theme_bw()
```



The emphasis on the tail is recurrent whatever instance, baseline, or sequence starting index. We speculate the model to be not biologically trustworthy. We thus retrain a model.

```
#model <- create_model_lstm_cnn(
#  maxlen = 600,
#  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)

#run_name <- "16S_vs_bacteria_full_1"

#hist <- train_model(train_type = "label_folder",
#  model = model,
#  path = c(path_16S_train, path_bacteria_train), # path to training files
#  path_val = c(path_16S_validation, path_bacteria_validation), # path to validation files
#  vocabulary_label = c("16s", "bacteria"),
#  path_checkpoint = checkpoint_path,
#  train_val_ratio = 0.2,
#  run_name = run_name,
#  batch_size = 256, # number of samples to process in parallel
#  steps_per_epoch = 25, # 1 epoch = 25 batches
#  epochs = 8,
#  step = c(6, 30),
```

```
#  proportion_per_seq = c(0.95, 0.05)) # randomly sample 95% from 16S files and
                                       # 5% from bacteria file
#plot(hist)
```

```
# load model checkpoint with best validation accuracy
run_name <- "16S_vs_bacteria_full_2"
model <- load_cp(paste0(checkpoint_path, "/", run_name),
                 cp_filter = "last_ep")
```

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

```
# evaluate model
path_input <- c(path_16S_validation, path_bacteria_validation)
pred_df_path <- tempfile(fileext = ".rds")
eval_model <- evaluate_model(path_input = path_input,
  model = model,
  batch_size = 250,
  step = 500,
  vocabulary_label = c("16s", "bacteria"),
  number_batches = 10,
  mode = "label_folder",
  verbose = FALSE,
  auc = TRUE,
  proportion_per_seq = 0.98,
  max_samples = 500,
  path_pred_list = pred_df_path)

eval_model
```

```
## [[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
```

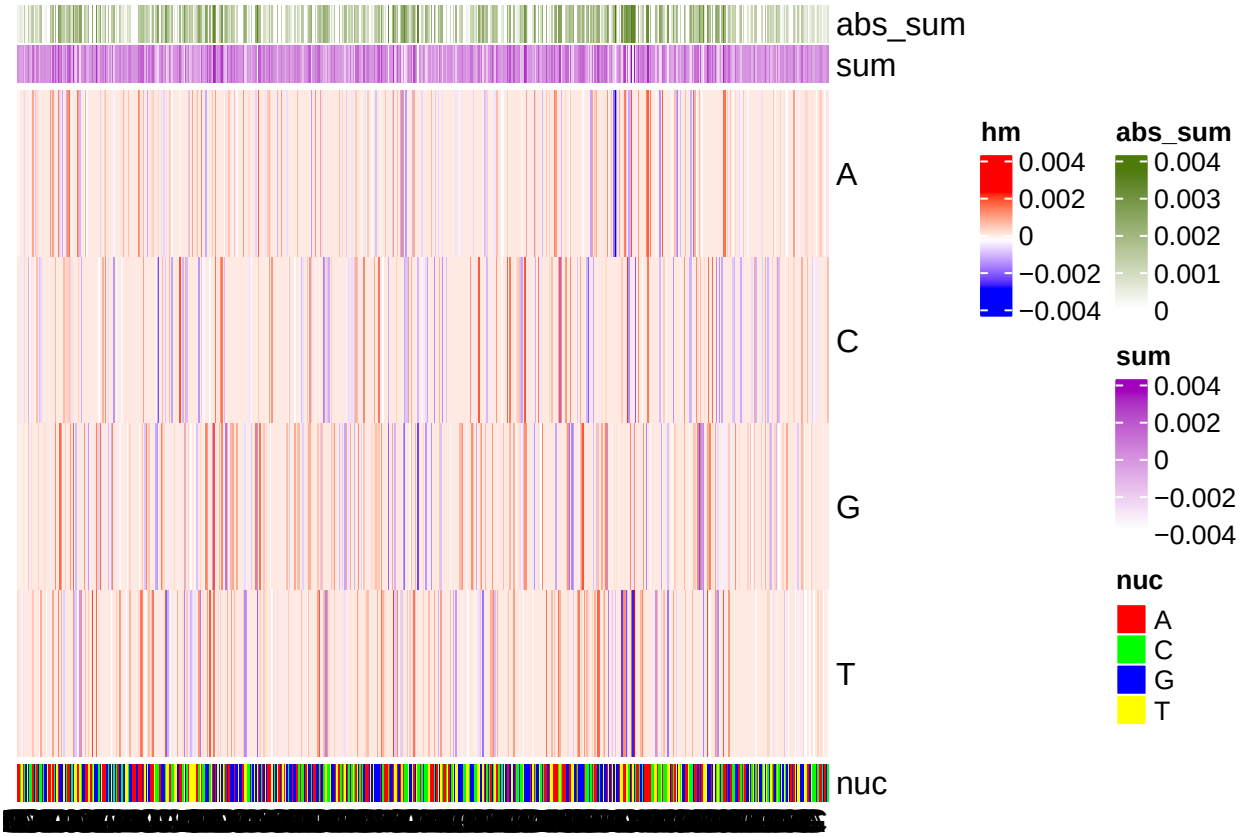
```
instance <- microseq::readFasta('16s/validation/GCF_001986655.1_ASM198665v1_genomic.16s.fasta.fasta')$S
instance_sub <- substr(instance, 499, 1098)
onehot_instance <-  seq_encoding_label(char_sequence = instance_sub,
                                       maxlen = 600,
```

```
                                          start_ind = 1,
                                          vocabulary = c("A", "C", "G", "T"))
baseline <- microseq::readFasta('bacteria/validation/GCF_002895085.1_ASM289508v1_genomic.fasta')$Sequen
baseline_sub <- substr(baseline, 499, 1098)
onehot_baseline <- seq_encoding_label(char_sequence = baseline_sub,
                                          maxlen = 600,
                                          start_ind = 1,
                                          vocabulary = c("A", "C", "G", "T"))
pred <- predict(model, onehot_instance, verbose = 0)
pred
```

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

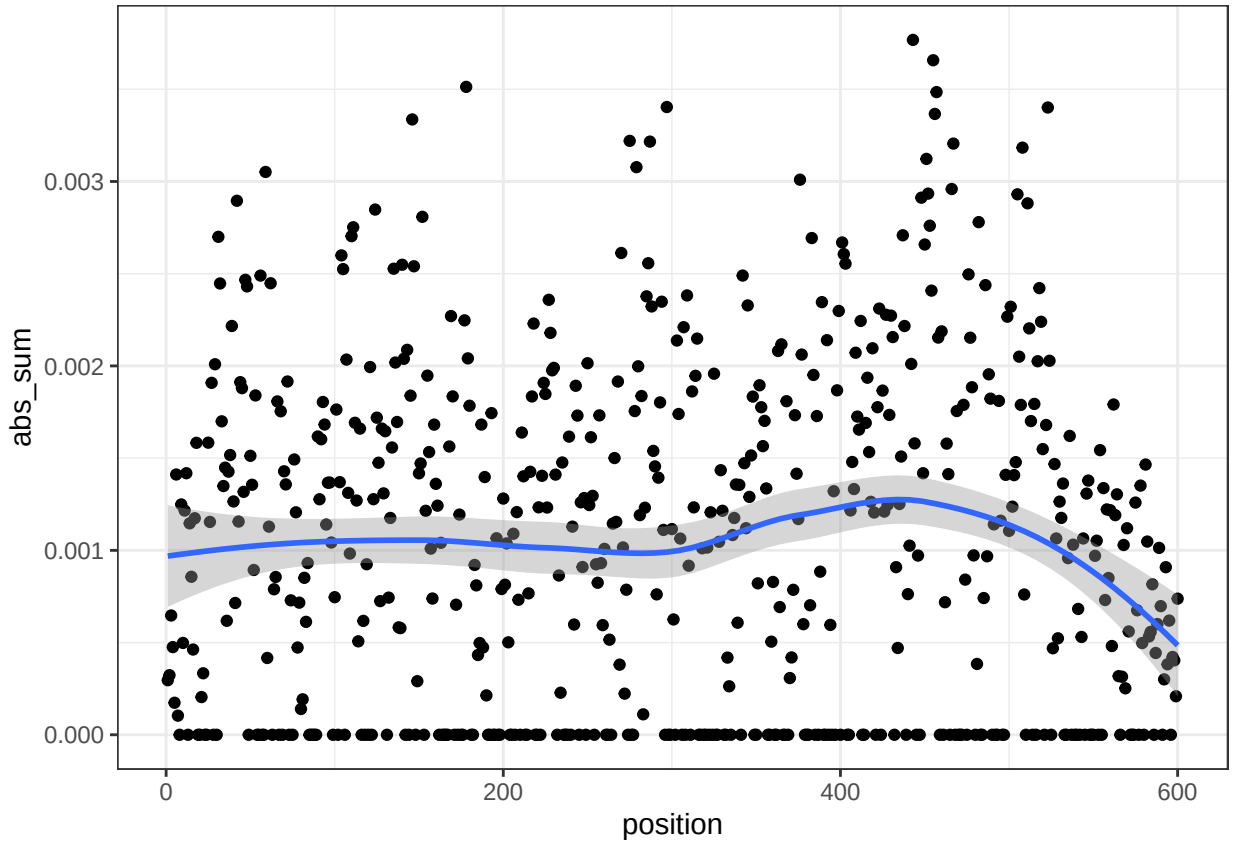
```
igs <- ig_modified(
  input_seq = onehot_instance,
  baseline_type = "modify",
  baseline_onehot = onehot_baseline,
  target_class_idx = 1,
  model = model,
  num_baseline_repeats = 1)
heatmaps_integrated_grad(integrated_grads = igs,
                         input_seq = onehot_instance)
```

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

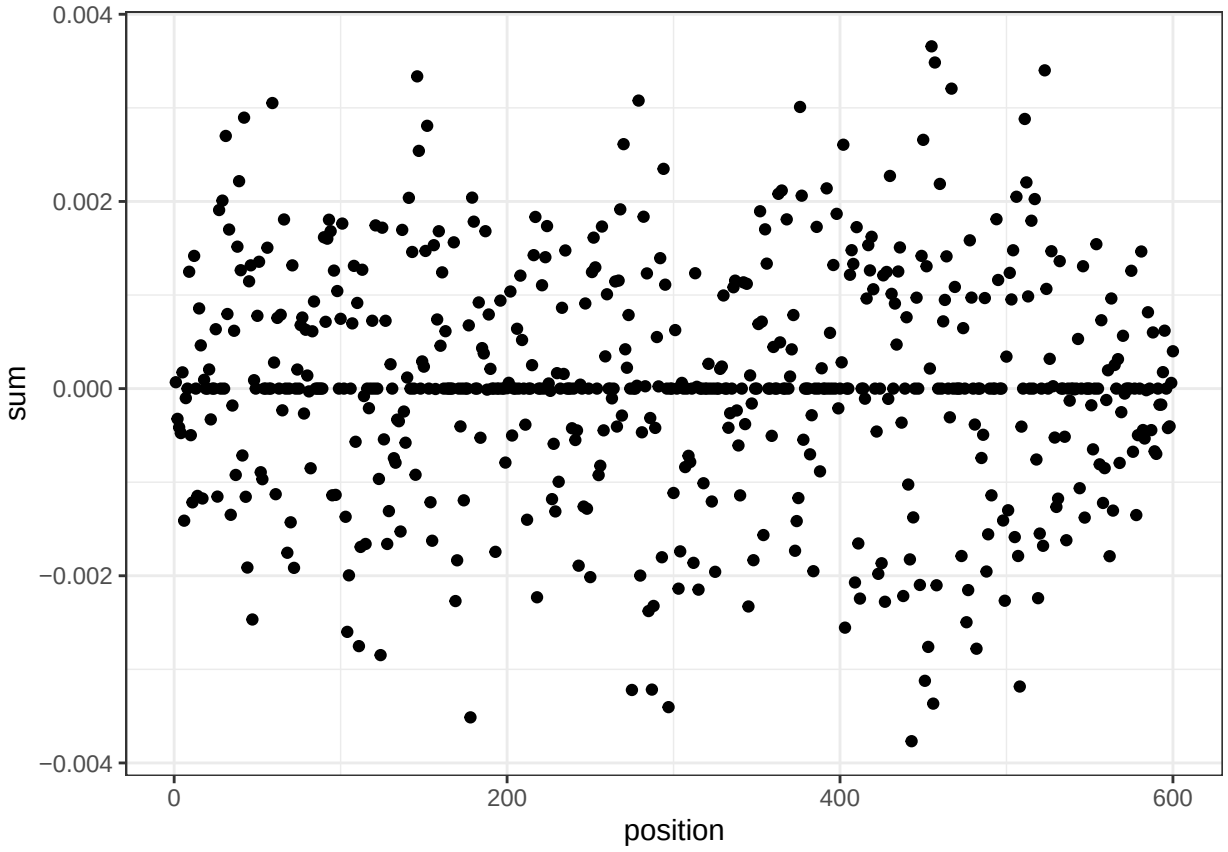


```r
sum <- rowSums(as.array(igs))
abs_sum <- rowSums(abs(as.array(igs)))
df <- data.frame(abs_sum = abs_sum, sum=sum, position = 1 : 600)

ggplot(df, aes(x = position, y = abs_sum)) + geom_point() + geom_smooth() + theme_bw()
```

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



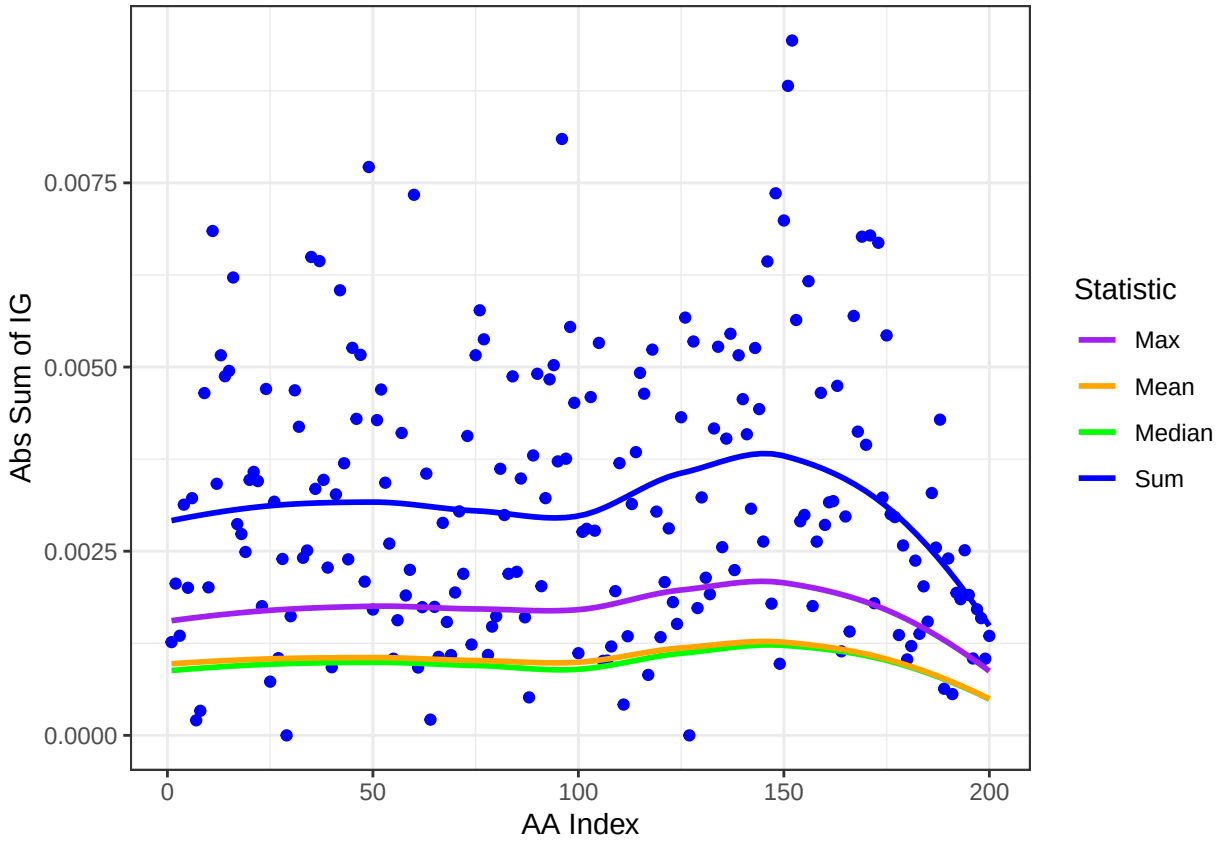

```r
ggplot(df, aes(x = position, y = sum)) + geom_point() + theme_bw()
```




```
df_mod <- df %>%
  mutate(group = rep(1:(nrow(df) / 3), each = 3)) %>%
  group_by(group) %>%
  summarise(
    abs_sum_sum = sum(abs_sum),
    abs_sum_median = median(abs_sum),
    abs_sum_mean = mean(abs_sum),
    abs_sum_max = max(abs_sum),
    sum_sum = sum(sum),
    sum_median = median(sum),
    sum_mean = mean(sum)
  )

ggplot(df_mod, aes(x = group)) +
  # draw points of max
  geom_point(aes(x = group, y = abs_sum_sum), color = "blue") +
  geom_smooth(aes(y = abs_sum_sum, color = "Sum"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = abs_sum_median, color = "Median"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = abs_sum_mean, color = "Mean"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = abs_sum_max, color = "Max"), method = "auto", se = FALSE) +
  scale_color_manual(values = c("Sum" = "blue", "Median" = "green", "Mean" = "orange", "Max"="purple")) +
  theme_bw() +
  labs(y = "Abs Sum of IG", color = "Statistic", x = "AA Index")
```

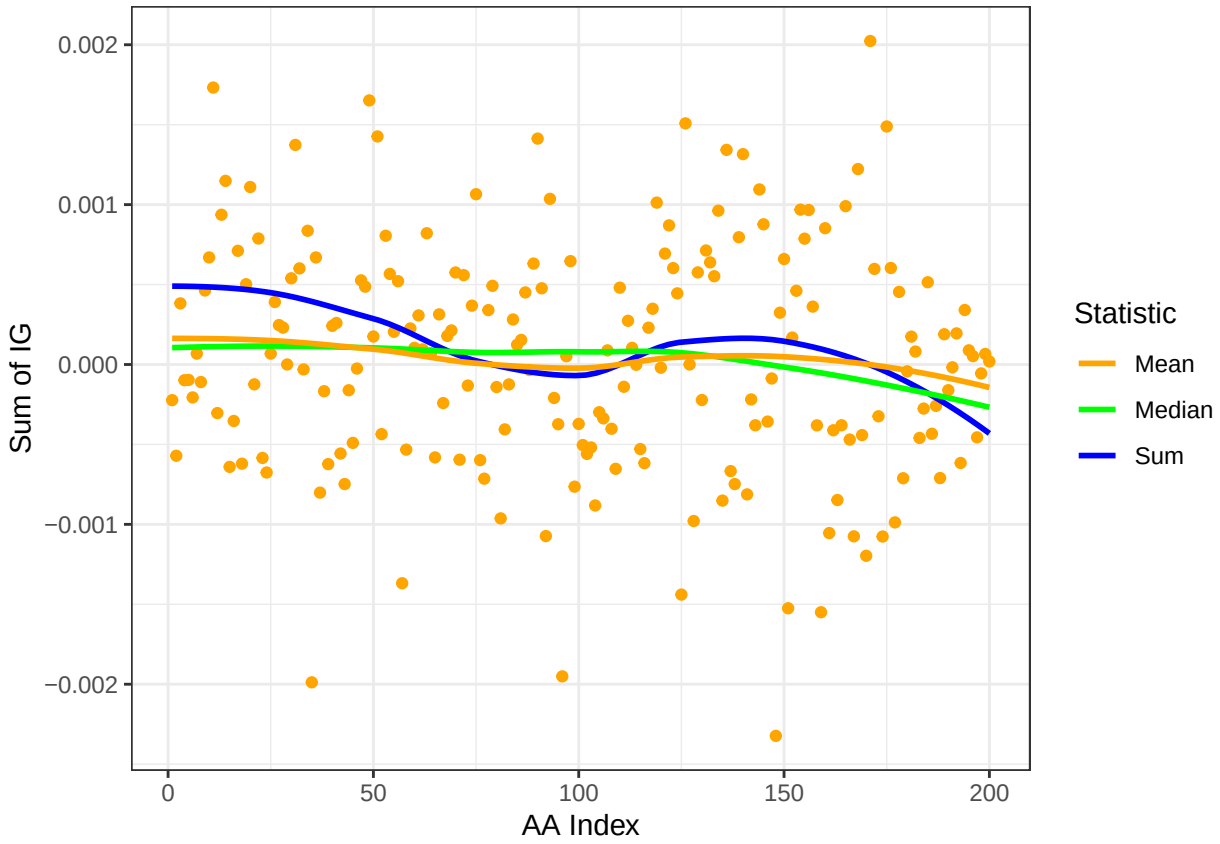
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

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



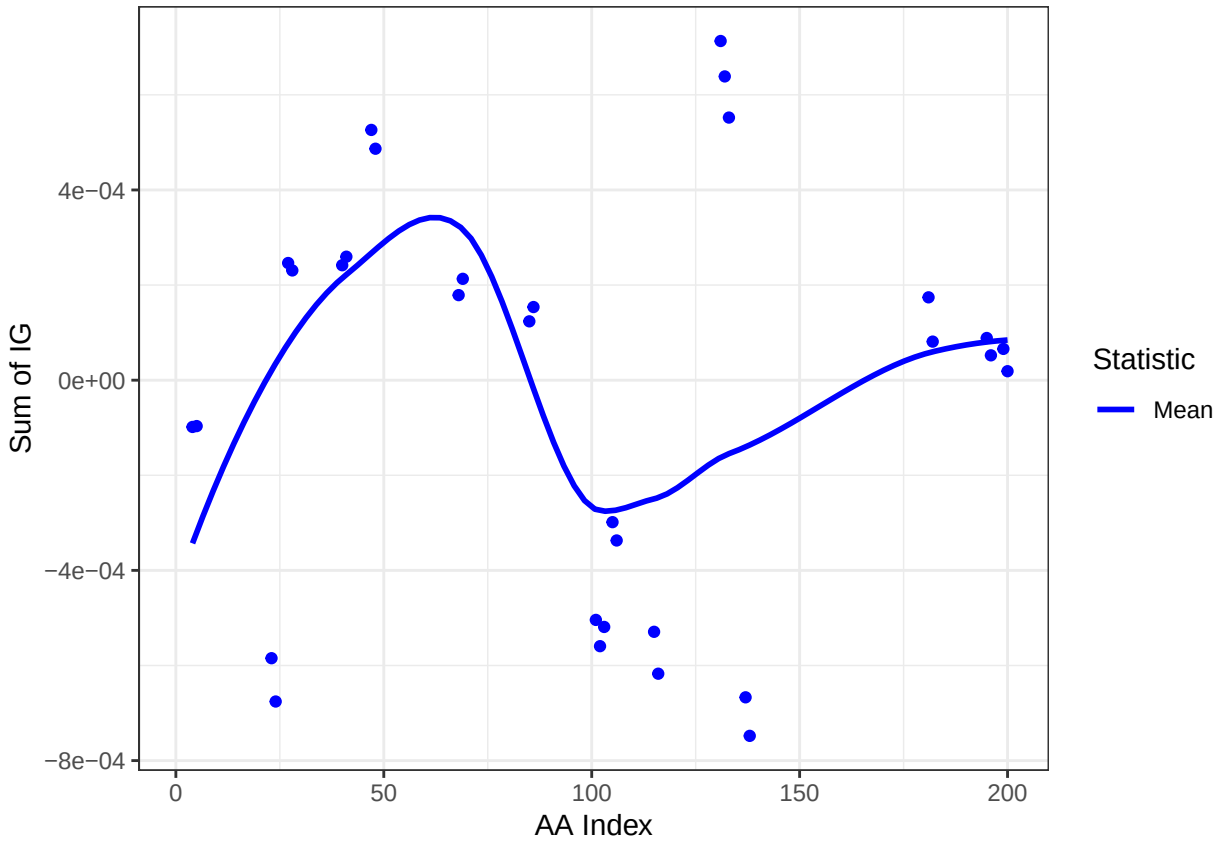

```
ggplot(df_mod, aes(x = group)) +
  # draw points of max
  geom_point(aes(x = group, y = sum_mean), color = "orange") +
  geom_smooth(aes(y = sum_sum, color = "Sum"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = sum_median, color = "Median"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = sum_mean, color = "Mean"), method = "auto", se = FALSE) +
  scale_color_manual(values = c("Sum" = "blue", "Median" = "green", "Mean" = "orange")) +
  theme_bw() +
  labs(y = "Sum of IG", color = "Statistic", x = "AA Index")
```

```
## `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'
```




```
df_sorted <- df_mod %>% arrange(group)
df_sorted <- df_sorted %>%
  mutate(diff_y = abs(sum_mean - lag(sum_mean, default = first(sum_mean))))
threshold <- 0.0001
df_sorted <- df_sorted %>%
  mutate(cluster = cumsum(diff_y >= threshold))  # Increment cluster ID when difference exceeds thresho

# Step 5: Filter clusters with more than one point (true clusters)
df_clusters <- df_sorted %>%
  group_by(cluster) %>%
  filter(n() > 1) %>%  # Keep only clusters with more than one point
  ungroup()

# plot
ggplot(df_clusters, aes(x = group)) +
  # draw points of max
  geom_point(aes(x = group, y = sum_mean), color = "blue") +
  geom_smooth(aes(y = sum_mean, color = "Mean"), method = "auto", se = FALSE) +
  scale_color_manual(values = c("Mean" = "blue", "Abs Mean" = "orange")) +
  theme_bw() +
  labs(y = "Sum of IG", color = "Statistic", x = "AA Index") +
  # set x range to 1:600
  xlim(1, 200)
```

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




```
char <- strsplit(instance_sub, "")[[1]]
trip_instance <- sapply(seq(1, length(char), by = 3), function(i) {
  paste(char[i:min(i+2, length(char))], collapse = "")
})
keyed_instance <- triplets_keying(trip_instance)
df_key <- data.frame(
  trip = trip_instance,
  key = keyed_instance,
  group = 1:200
)
# left join df_key to df_clusters by group
df_clusters <- left_join(df_clusters, df_key, by = "group")
df_keyed <- left_join(df_mod, df_key, by = "group")
seq_cluster <- df_clusters %>%
  group_by(cluster) %>%
  summarise(
    score = sum(sum_mean),
    seq = paste(trip, collapse = ""),
    aa = paste(key, collapse = "")
  )
kableExtra::kable(seq_cluster, format = "latex", booktabs = TRUE, caption = "Possible clustering of Impo
```

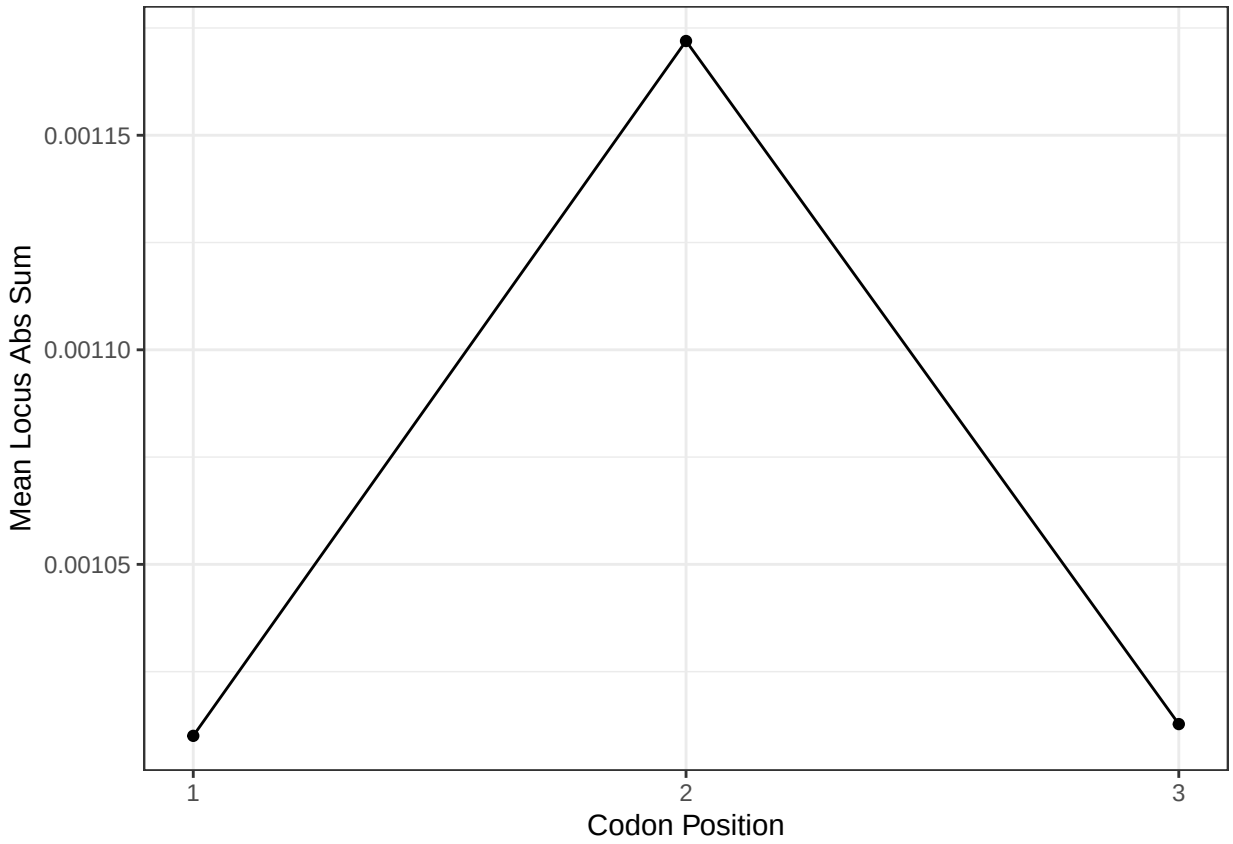
```
df_keyed <- df_keyed %>%
  mutate(cod1 = substr(trip, 1, 1),
         cod23 = substr(trip, 2, 3))
df_keyed <- df_keyed %>%
```

Table 1: Possible clustering of Importance score

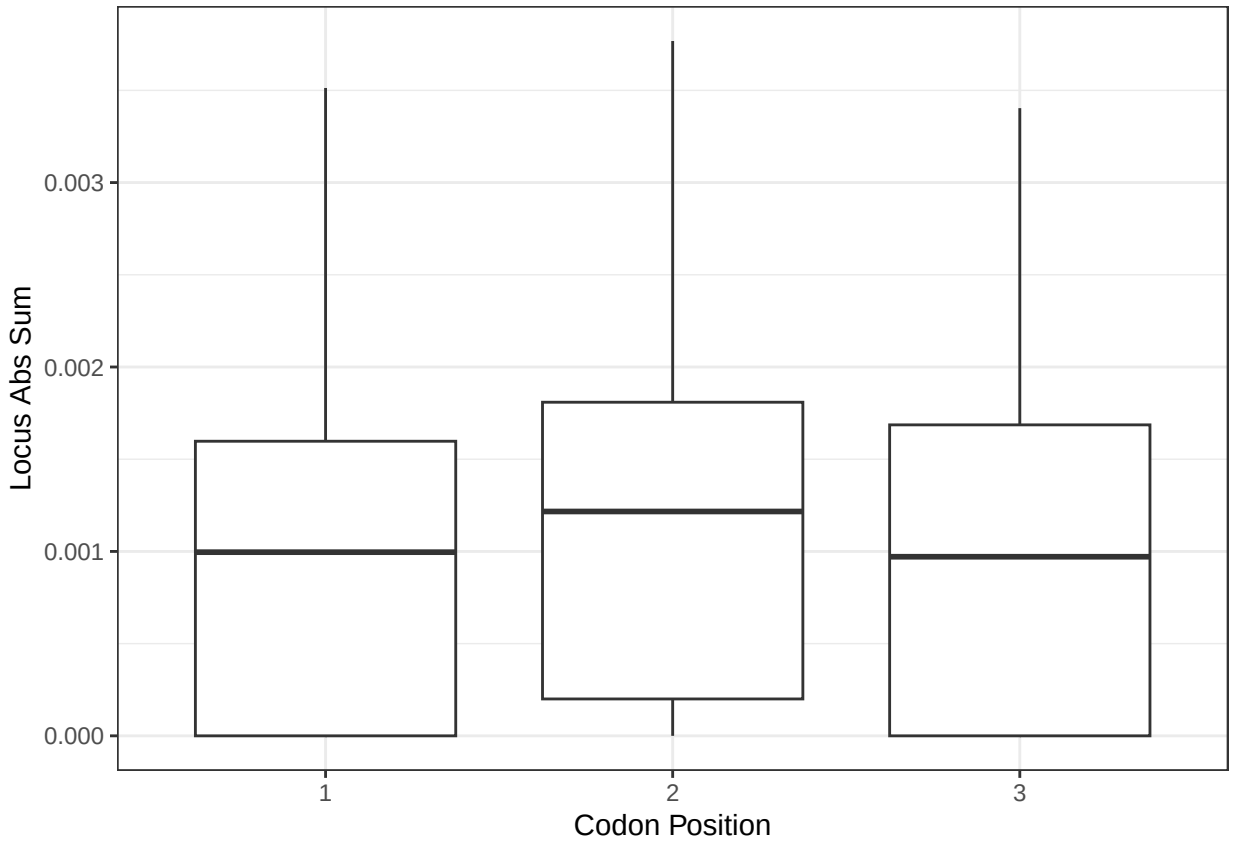
cluster	score	seq	aa
3	-0.0001950	CCAGCA	PA
21	-0.0012604	GCGTGC	AC
24	0.0004771	GGTTCG	GS
36	0.0005014	TGGAAC	WN
42	0.0010130	CGGGCT	RA
62	0.0003919	GGAGGA	GG
78	0.0002775	GCACGA	AR
93	-0.0015825	GCCCTAAAC	ALN
95	-0.0006357	GTCAAC	VN
104	-0.0011464	GGTAGC	GS
119	0.0019041	AAGATTAAA	KIK
123	-0.0014149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

```
  group_by(key) %>%
  mutate(value = mean(sum_mean))

hm1 <- ggplot(df_keyed, aes(x = cod1, y = cod23, fill = value)) +
  geom_tile() +
  geom_text(aes(label = key), color = "black", size = 3) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_bw() +
  labs(fill = "Average Mean\nof locus Sum",
       x = "First Codon Position",
       y = "Second and Third Codon Position") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        strip.text = element_text(face = "bold"),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank())
# hm1
```

```
mean_codon <- sapply(1:3, function(i) {
  mean(df$abs_sum[seq(i, 600, by = 3)])
})
# plot with line. x:1,2,3, y: mean_codon
ggplot(data = data.frame(x = 1:3, y = mean_codon), aes(x = x, y = y)) +
  geom_line() +
  geom_point() +
  theme_bw() +
  labs(x = "Codon Position", y = "Mean Locus Abs Sum") +
  # x tick only 1,2,3, integer
  scale_x_continuous(breaks = 1:3, minor_breaks = NULL, labels = c("1", "2", "3"))
```




```r
codon_data <- data.frame(
  position = rep(1:3, each = length(df$abs_sum) / 3),
  abs_sum = unlist(lapply(1:3, function(i) df$abs_sum[seq(i, 600, by = 3)]))
)

# Box plot for each codon position
ggplot(codon_data, aes(x = factor(position), y = abs_sum)) +
  geom_boxplot() +
  theme_bw() +
  labs(x = "Codon Position", y = "Locus Abs Sum") +
  scale_x_discrete(labels = c("1", "2", "3"))
```




```
lm_model <- lm(abs_sum ~ factor(position), data = codon_data)

# Summarize the linear model
summary(lm_model)
```

```
##
## 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
```

```r
# ok-mutate based on instance, 100 times, saved to pcss_df
# just use permute_sequence, nothing else
pcss_df <- data.frame(permuted = instance_sub)
for (i in 1:999) {
  permuted_instance <- permute_sequence(trip_instance, type = "ok", min.subs = 5,
                                        max.subs = 30, dict = codon.dict,
                                        spec.cond = FALSE, spec.region = NULL)
  permuted_instance <- paste(permuted_instance, collapse = "")
  pcss_df <- rbind(pcss_df, data.frame(permuted = I(list(permuted_instance))))
}
list_onehot <- lapply(pcss_df$permuted, function(x) {
  seq_encoding_label(char_sequence = x, maxlen = 600, start_ind = 1, vocabulary = c("A", "C", "G", "T"))
})
```

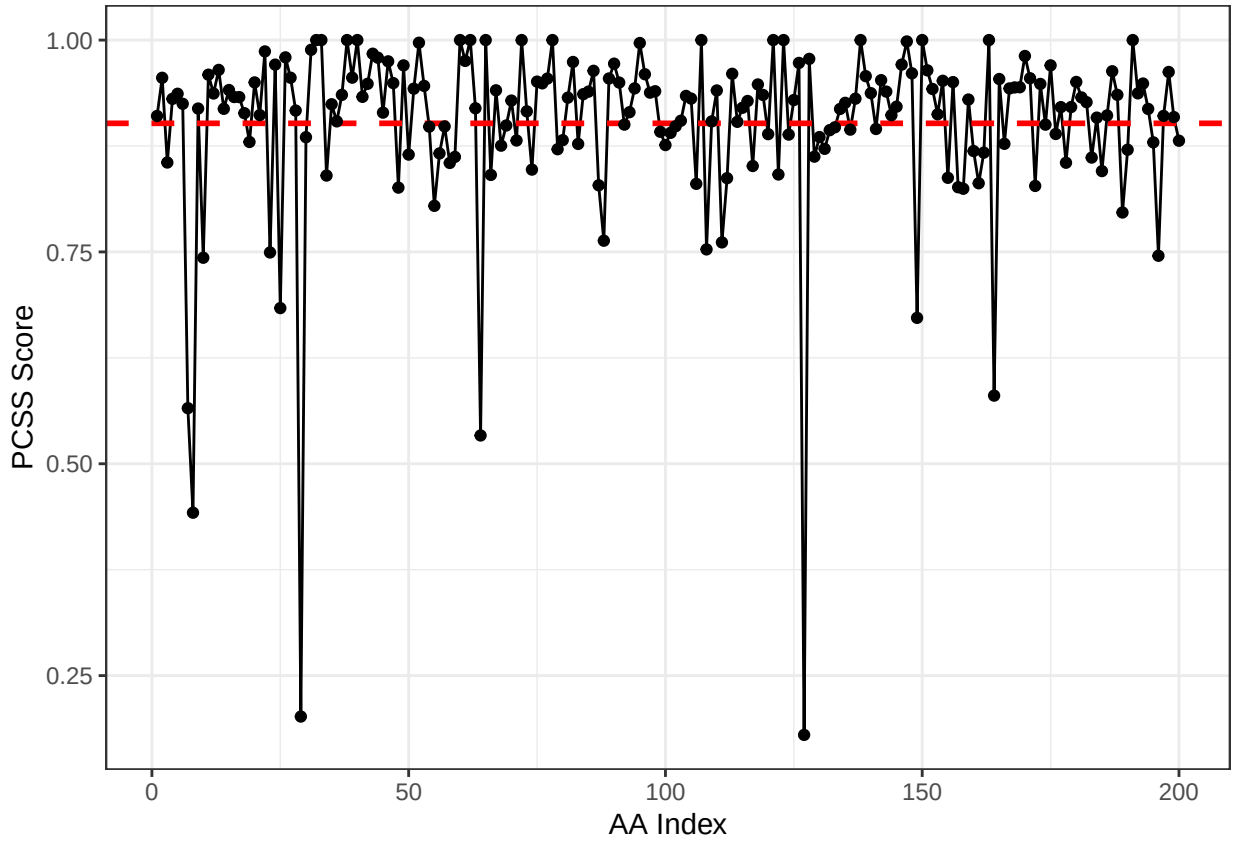
```r
csv_file_path <- "pcssdata_16s.csv"

# Check if the CSV file already exists
if (file.exists(csv_file_path)) {
  # If it exists, read the CSV into result_df
  result_df <- read.csv(csv_file_path)
  message("Loaded result_df from existing CSV file.")
} else {
# Initialize an empty dataframe with the position column
  result_df <- data.frame(position = 1:600)

# Loop through each one-hot encoded instance in the list
  for (i in seq_along(list_onehot)) {
    onehot_instance <- list_onehot[[i]]

  # Compute Integrated Gradients
    igw <- ig_modified(
      input_seq = onehot_instance,
      baseline_type = "modify",
      baseline_onehot = onehot_baseline,
      target_class_idx = 1,
      model = model,
      num_baseline_repeats = 1)

  # Compute the absolute sum of the IG scores
    abs_sum <- rowSums(abs(as.array(igw)))

  # Add the abs_sum as a new column in the result_df
    result_df[[paste0("abssum", i)]] <- abs_sum
  }
  write.csv(result_df, csv_file_path, row.names = FALSE)
}
```

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

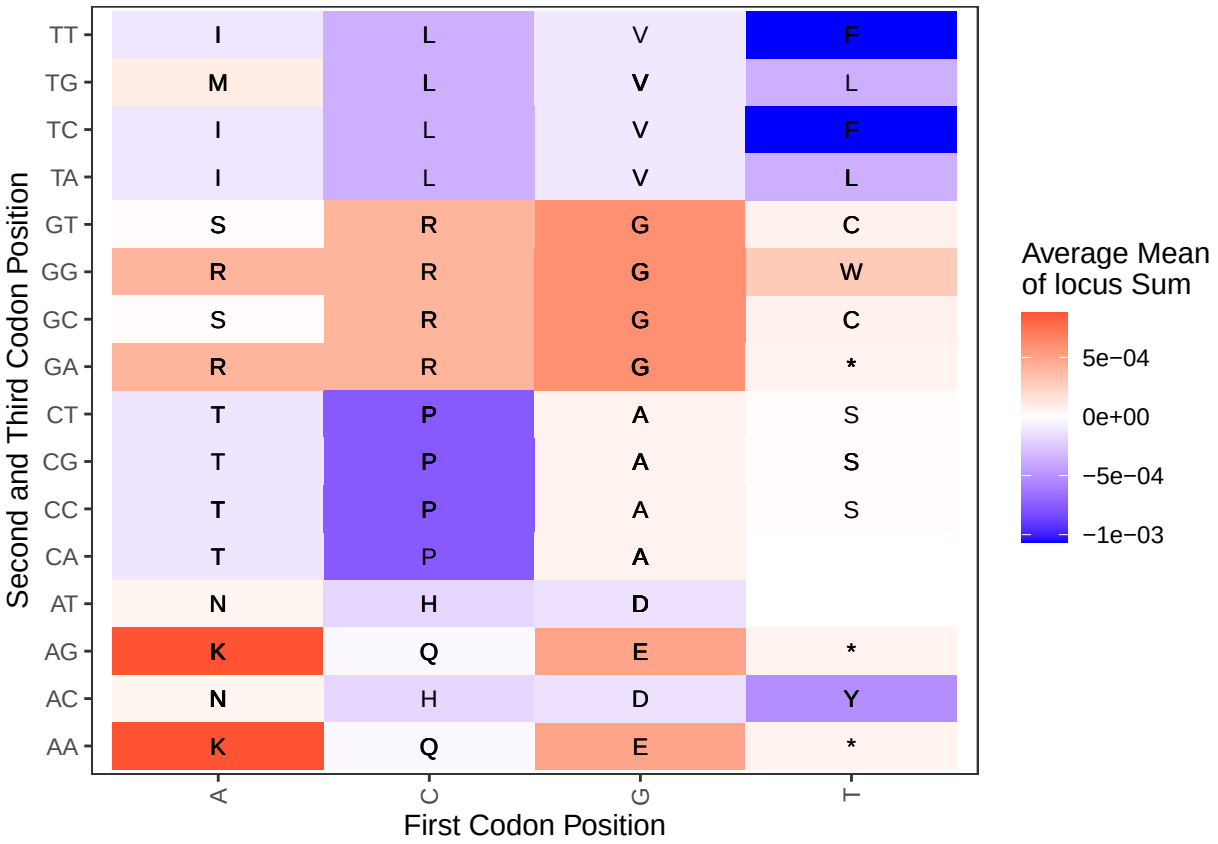
```r
calculate_mean_every_three_rows <- function(df) {
  # Calculate the number of groups (each group will consist of three rows)
  n_groups <- nrow(df) %/% 3
```

```
  # Initialize an empty list to store the means
  mean_list <- list()

  # Loop through each group and calculate the mean for each column
  for (i in 1:n_groups) {
    # Select the three rows corresponding to the current group
    rows <- df[((i-1) * 3 + 1):(i * 3), ]

    # Calculate the mean for each column in the current group
    mean_row <- colSums(rows)

    # Append the result to the list
    mean_list[[i]] <- mean_row
  }

  # Combine the results into a new dataframe
  mean_df <- do.call(rbind, mean_list)

  return(mean_df)
}

# Apply the function to result_df
result_df_mean <- calculate_mean_every_three_rows(result_df)

# Convert the result to a data frame with appropriate column names
result_df_mean <- as.data.frame(result_df_mean)
names(result_df_mean) <- names(result_df)
result_df_mean <- result_df_mean %>%
  dplyr::select(-position)

# Function to calculate the coefficient of variance (CV) for each row
calculate_cv_rowwise <- function(df) {
  # Apply the function to calculate CV (sd/mean) row-wise
  cv <- apply(df[-1], 1, function(row) {
    row_sd <- sd(row, na.rm=TRUE)
    row_mean <- mean(row, na.rm=TRUE)
    if (row_mean != 0) {
      return(row_sd / row_mean)
    } else {
      return(NA)  # Handle division by zero
    }
  })
  return(cv)
}
invtrans <- function(x) {
  return(1 / (1 + x))
}

# Apply the function to result_df_mean (excluding the first column "position")
rowwise_std <- calculate_cv_rowwise(result_df_mean)
pcss_final <- data.frame(position=1:length(rowwise_std), key = keyed_instance,
                         trip_instance, sig.cv=invtrans(rowwise_std))
```

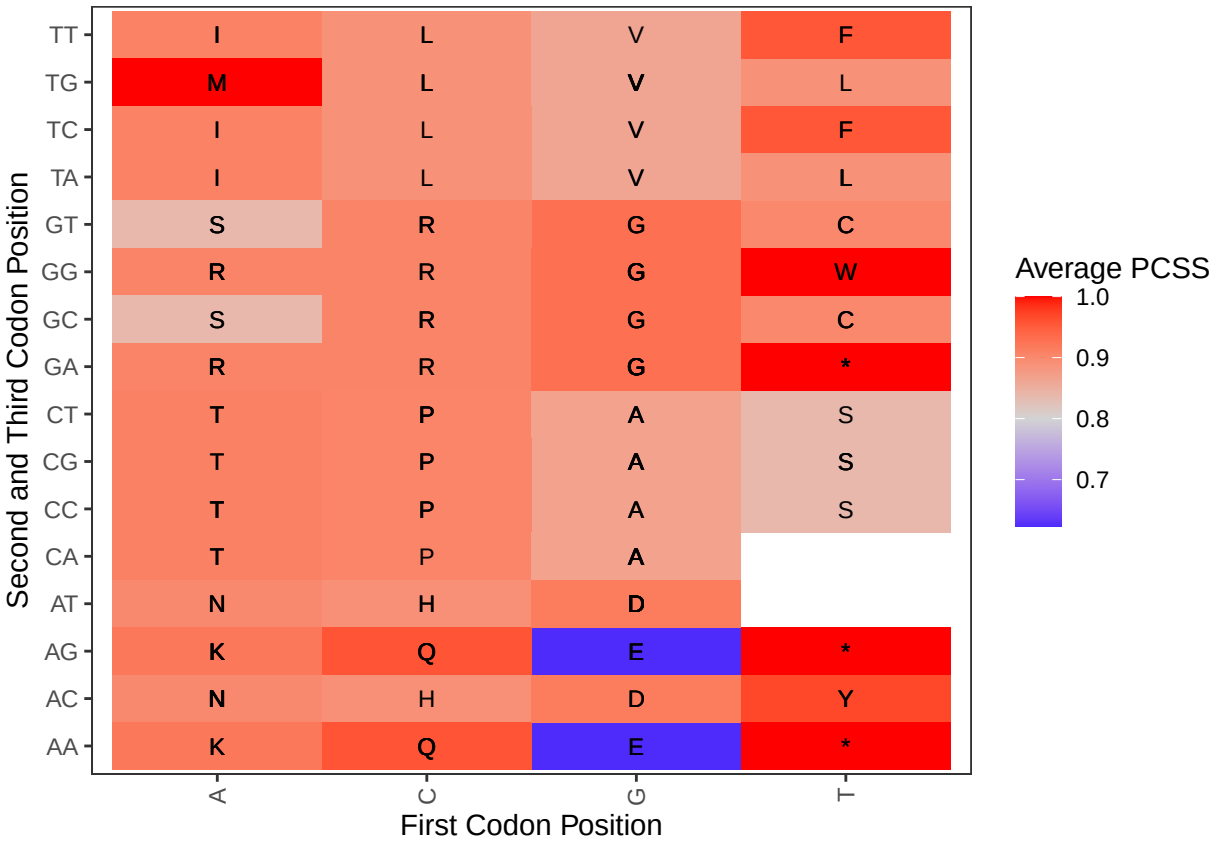
```r
# plot pcss_final, x:position, y:cv, draw a horizontal line for the mean(pcss_final$cv), and mark its y
ggplot(pcss_final, aes(x = position, y = sig.cv)) +
  geom_line() +
  geom_hline(yintercept = mean(pcss_final$sig.cv, na.rm=TRUE), linetype = "dashed", lwd=1, color = "red
  geom_point() +
  theme_bw() +
  labs(x = "AA Index", y = "PCSS Score")
```



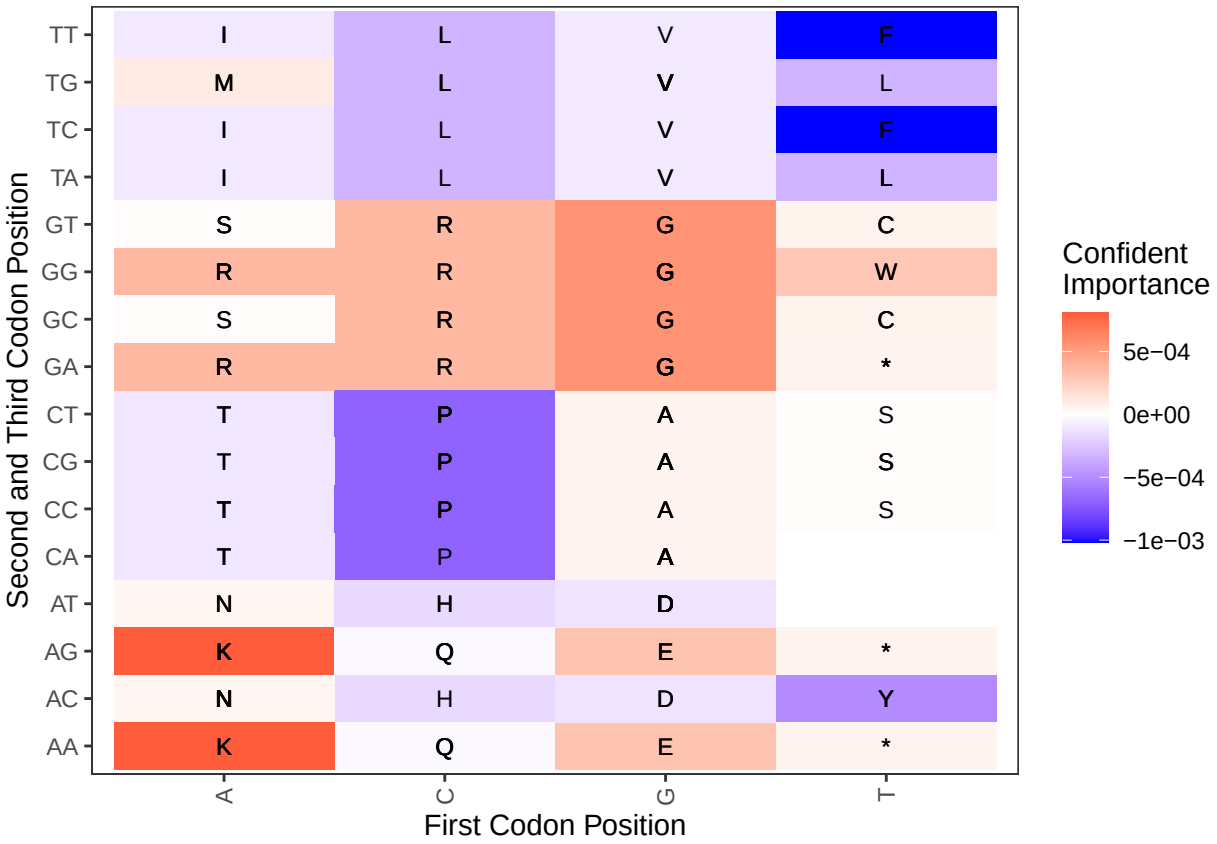

```r
pcss_final <- pcss_final %>%
  mutate(cod1 = substr(trip_instance, 1, 1),
         cod23 = substr(trip_instance, 2, 3))
pcss_final <- pcss_final %>%
  group_by(key) %>%
  mutate(value = mean(sig.cv))
hm1
```



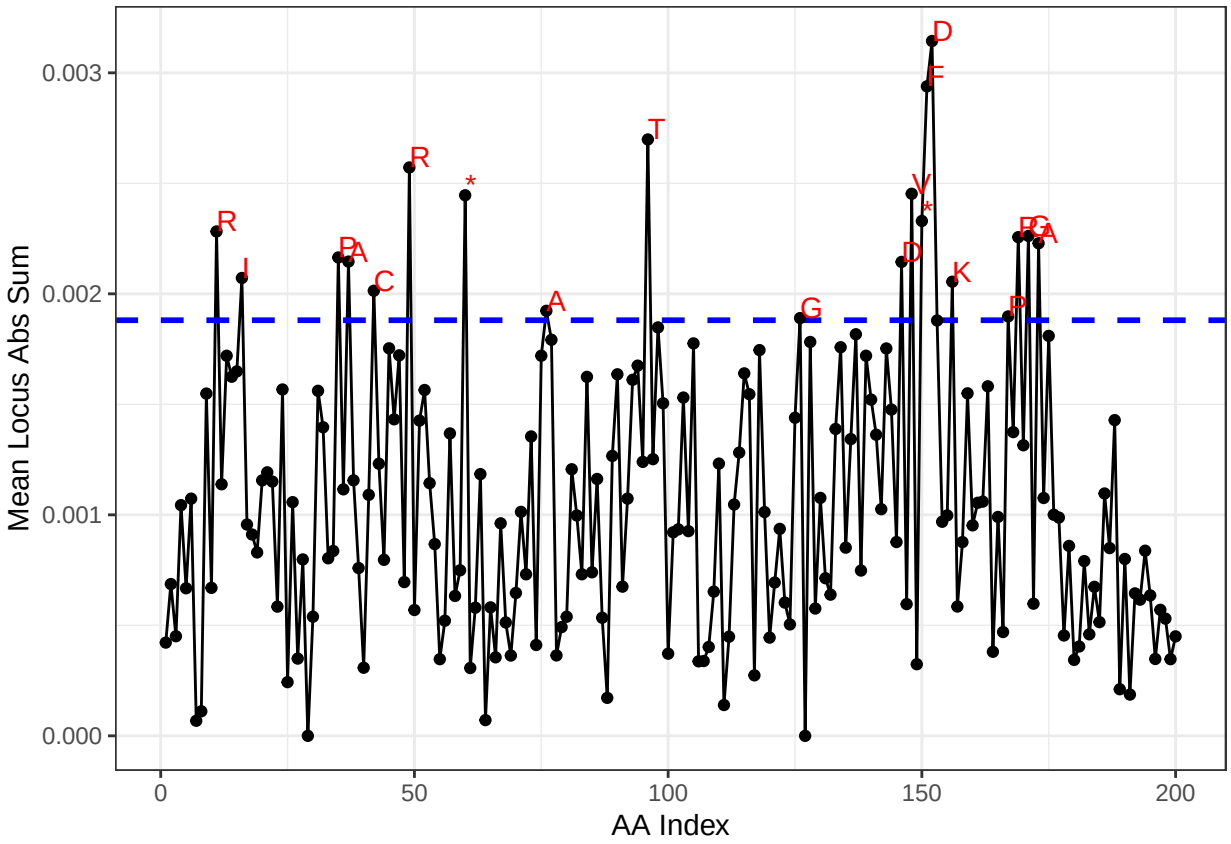

```
hm2 <- ggplot(pcss_final, aes(x = cod1, y = cod23, fill = value)) +
  geom_tile() +
  geom_text(aes(label = key), color = "black", size = 3) +
  scale_fill_gradient2(low = "blue", mid = "lightgrey", high = "red", midpoint = 0.8) +
  theme_bw() +
  labs(fill = "Average PCSS",
       x = "First Codon Position",
       y = "Second and Third Codon Position") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        strip.text = element_text(face = "bold"),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank())
hm2
```




```
combined <- left_join(pcss_final, df_keyed, by = c("position" = "group"))
combined <- combined %>%
  mutate(conf = value.x*value.y)

hm3 <- ggplot(combined, aes(x = cod1.x, y = cod23.x, fill = conf)) +
  geom_tile() +
  geom_text(aes(label = key.x), color = "black", size = 3) +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_bw() +
  labs(fill = "Confident\nImportance",
       x = "First Codon Position",
       y = "Second and Third Codon Position") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
        strip.text = element_text(face = "bold"),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank())
hm3
```




```r
quantile_90 <- quantile(df_keyed$abs_sum_mean, 0.90)
df_keyed_above_90 <- df_keyed %>% filter(abs_sum_mean > quantile_90)
ggplot(df_keyed, aes(x = group, y = abs_sum_mean)) +
  geom_line() +
  geom_hline(yintercept = quantile_90, linetype = "dashed", lwd=1, color = "blue") +
  geom_point() +
  geom_text(data = df_keyed_above_90,
            aes(label = key),
            vjust = -0.0005, hjust = 0,
            color = "red") +
  theme_bw() +
  labs(x = "AA Index", y = "Mean Locus Abs Sum")
```


Mean Locus Abs Sum
0.003
0.002
0.001
0.000
0
50
100
150
200
AA Index
R
I
P
A
C
R
*
A
T
G
D
F
V
*
D
K
P
R
G
A