IG Modification

Yichen Han

2024-08-10

```
# wild type baseline
rsd <- seqinr::read.fasta("rsd.FASTA", seqtype = "DNA")
rsd <- toupper(unlist(seqinr::getSequence(rsd)))
triplets <- tokenize_triplets(rsd)
keyed <- triplets_keying(triplets)
rsd_pasted <- paste(triplets, collapse = "")
onehot_baseline <-  seq_encoding_label(char_sequence = rsd_pasted,
                                       maxlen = 477,
                                       start_ind = 1,
                                       vocabulary = c("A", "C", "G", "T"))
```

```
# abnormal baseline
bs_abn <- permute_sequence(triplets, type="ok", min.subs=10, max.subs=30,
                           dict=codon.dict, spec.cond=FALSE, spec.region=NULL)
bs_abn <- permute_sequence(bs_abn, type="func", min.subs=10, max.subs=30,
                           dict=codon.dict, spec.cond=FALSE, spec.region=30:60)
bsabn_pasted <- paste(bs_abn, collapse = "")
chars <- strsplit(bsabn_pasted, "")[[1]]
# Create triplets
trip_abn <- sapply(seq(1, length(chars), by = 3), function(i) {
  paste(chars[i:min(i+2, length(chars))], collapse = "")
})
keyed_abn <- triplets_keying(trip_abn)
onehot_bsabn <-  seq_encoding_label(char_sequence = bsabn_pasted,
                                       maxlen = 477,
                                       start_ind = 1,
                                       vocabulary = c("A", "C", "G", "T"))
```

```
checkpoint_path <- file.path("checkpoints")
dir_path <- file.path("outputs")
run_name <- "rsd-permutation_39"
model <- load_cp(paste0(checkpoint_path, "/", run_name),
                 cp_filter = "last_ep")
```

```
## Using checkpoint checkpoints/rsd-permutation_39/Ep.010-val_loss0.18-val_acc0.946.hdf5
```

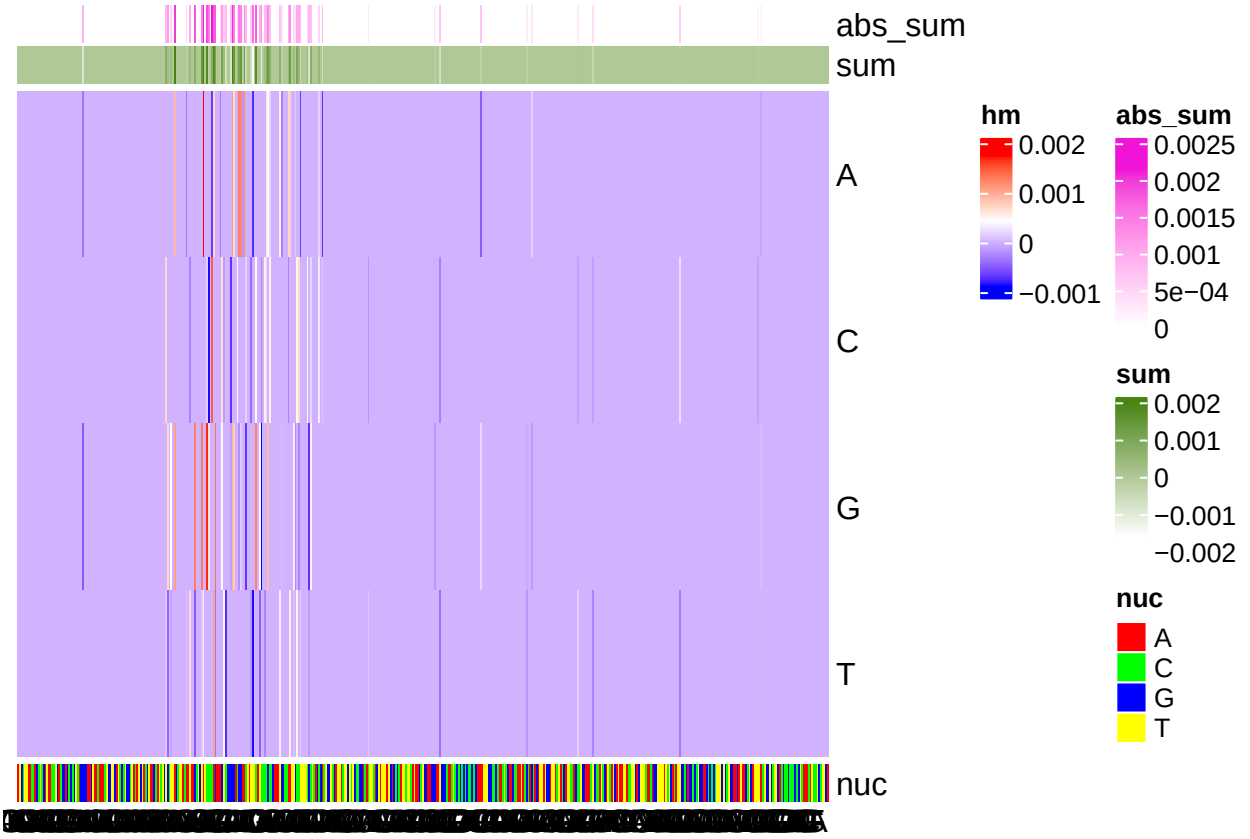
```
instance <- permute_sequence(triplets, type="ok", min.subs=10, max.subs=30,
                           dict=codon.dict, spec.cond=FALSE, spec.region=NULL)
instance <- permute_sequence(instance, type="func", min.subs=10, max.subs=30,
                           dict=codon.dict, spec.cond=TRUE, spec.region=30:60)
```

```
instance_pasted <- paste(instance, collapse = "")
onehot_instance <-  seq_encoding_label(char_sequence = instance_pasted,
                                       maxlen = 477,
                                       start_ind = 1,
                                       vocabulary = c("A", "C", "G", "T"))
pred <- predict(model, onehot_instance, verbose = 0)
pred
```

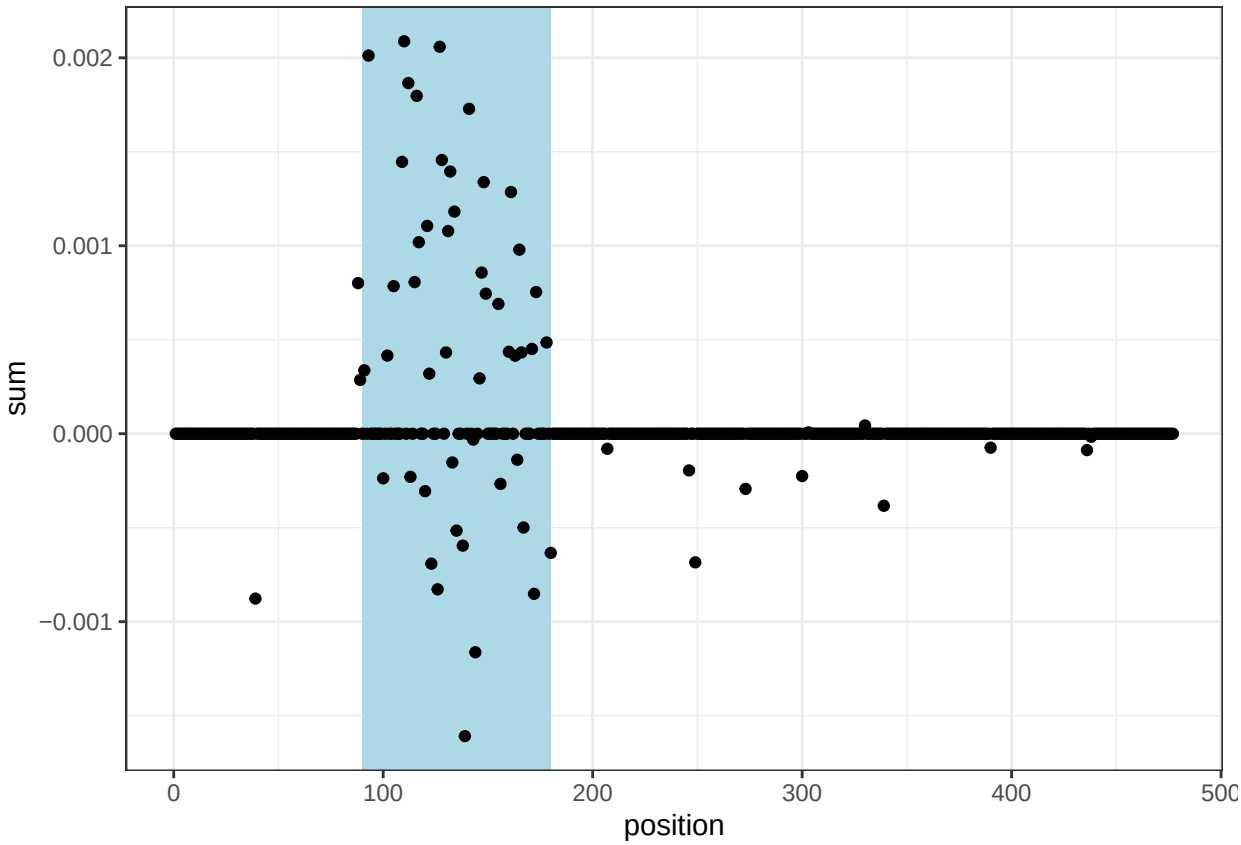
```
##              [,1]         [,2] [,3]
## [1,] 9.656643e-09 1.406469e-18    1
```

```
ig <- ig_modified(
  input_seq = onehot_instance,
  baseline_type = "modify",
  baseline_onehot = onehot_baseline,
  target_class_idx = 3,
  model = model,
  num_baseline_repeats = 1)

heatmaps_integrated_grad(integrated_grads = ig,
                         input_seq = onehot_instance)
```

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

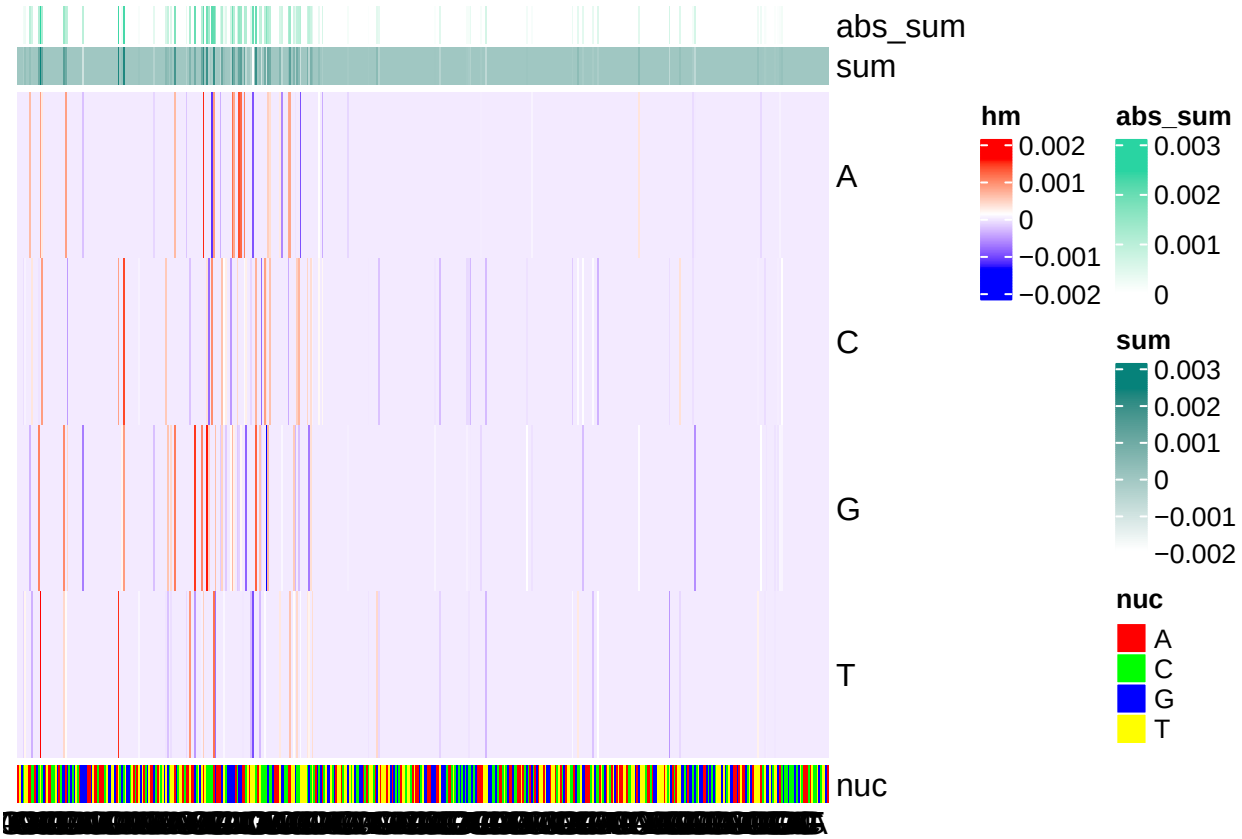


```r
sum <- rowSums(as.array(ig))
abs_sum <- rowSums(abs(as.array(ig)))
df <- data.frame(abs_sum = abs_sum, sum=sum, position = 1 : 477)

ggplot(df, aes(x = position, y = sum)) + geom_rect(aes(xmin = 90, xmax = 180, ymin = -Inf, ymax = Inf),
```

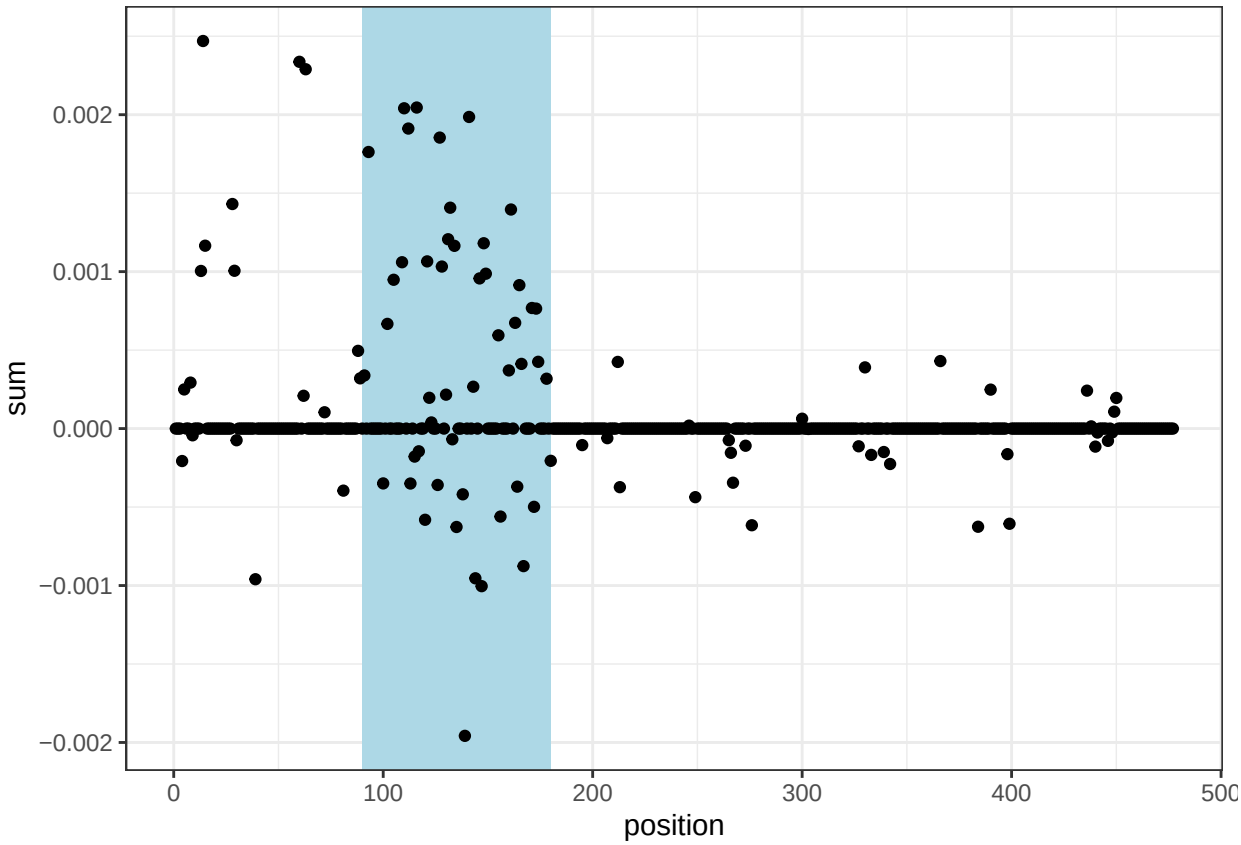



```r
ig2 <- ig_modified(
  input_seq = onehot_instance,
  baseline_type = "modify",
  baseline_onehot = onehot_bsabn,
  target_class_idx = 3,
  model = model,
  num_baseline_repeats = 1)

heatmaps_integrated_grad(integrated_grads = ig2,
                         input_seq = onehot_instance)
```

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

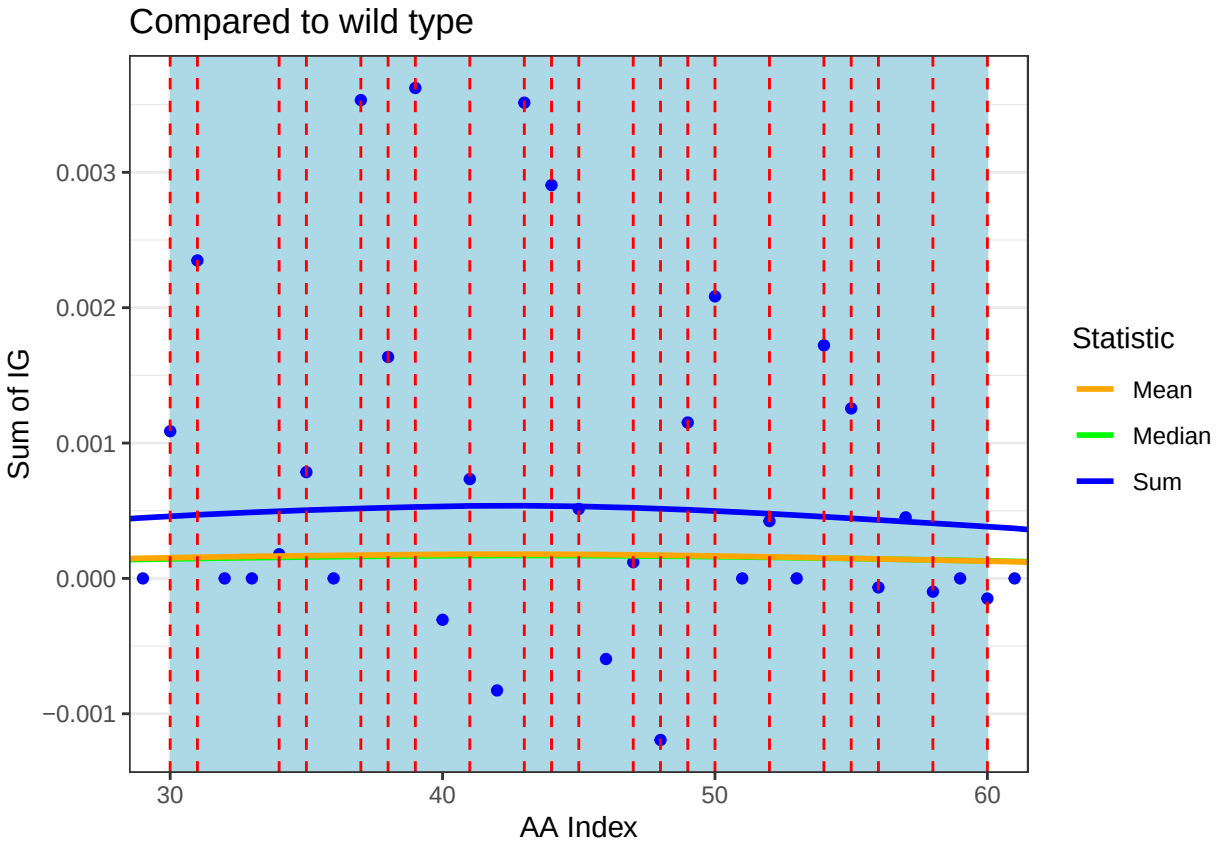



```
sum <- rowSums(as.array(ig2))
abs_sum <- rowSums(abs(as.array(ig2)))
df2 <- data.frame(abs_sum = abs_sum, sum=sum, position = 1 : 477)

ggplot(df2, aes(x = position, y = sum)) + geom_rect(aes(xmin = 90, xmax = 180, ymin = -Inf, ymax = Inf)
```

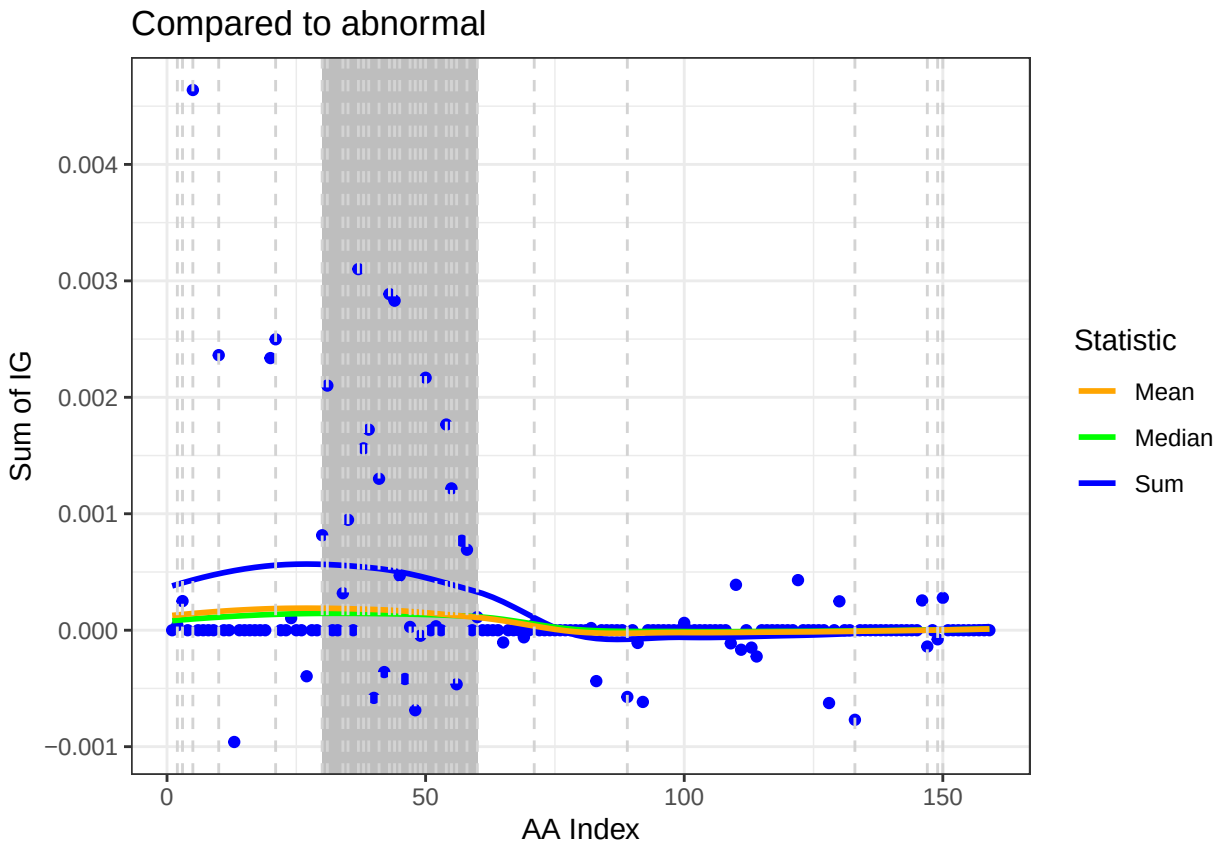



```
df_mod <- df %>%
  mutate(group = rep(1:(nrow(df) / 3), each = 3)) %>%
  group_by(group) %>%
  summarise(
    abs_sum_sum = sum(abs_sum),
    abs_sum_median = median(abs_sum),
    abs_sum_mean = mean(abs_sum),
    abs_sum_max = max(abs_sum),
    sum_sum = sum(sum),
    sum_median = median(sum),
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

df2_mod <- df2 %>%
  mutate(group = rep(1:(nrow(df2) / 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)
  )
```

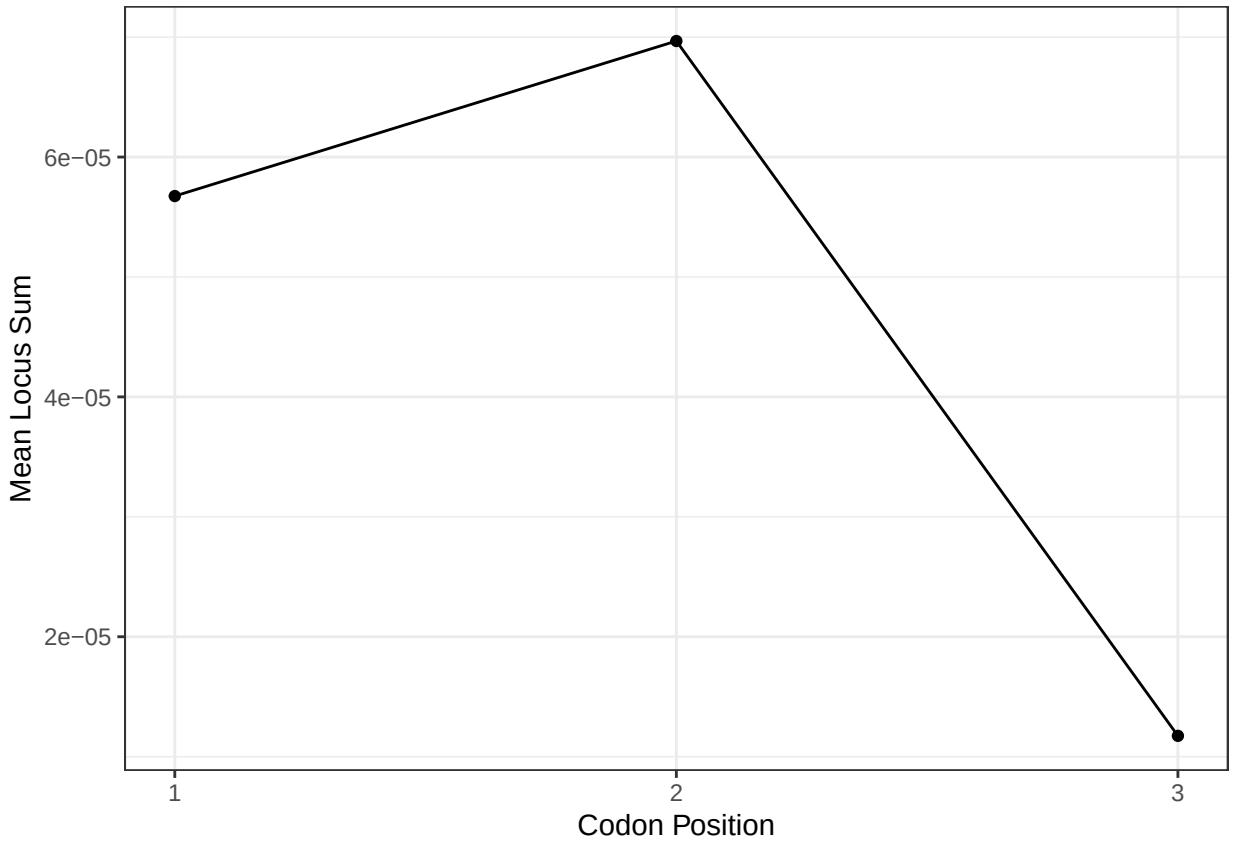
```r
# Split instance (a string) into triplets (every third character)
chars <- strsplit(instance_pasted, "")[[1]]

# Create triplets
trip_instance <- sapply(seq(1, length(chars), by = 3), function(i) {
  paste(chars[i:min(i+2, length(chars))], collapse = "")
})

keyed_instance <- triplets_keying(trip_instance)

# Get the index of sequences at which trip_instance != triplets
index <- which(keyed_instance != keyed)

index2 <- which(keyed_instance != keyed_abn)

ggplot(df_mod, aes(x = group)) +
  geom_rect(aes(xmin = 30, xmax = 60, ymin = -Inf, ymax = Inf), fill = "lightblue", alpha = 0.2) +
  # draw points of max
  geom_point(aes(x = group, y = sum_sum), color = "blue") +
  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")) +
  geom_vline(xintercept = index, color = "red", linetype = "dashed") +
  theme_bw() +
  labs(y = "Sum of IG", color = "Statistic", x = "AA Index",
       title = "Compared to wild type") +
  # set x region to 30:60 only
  coord_cartesian(xlim = c(30, 60))
```



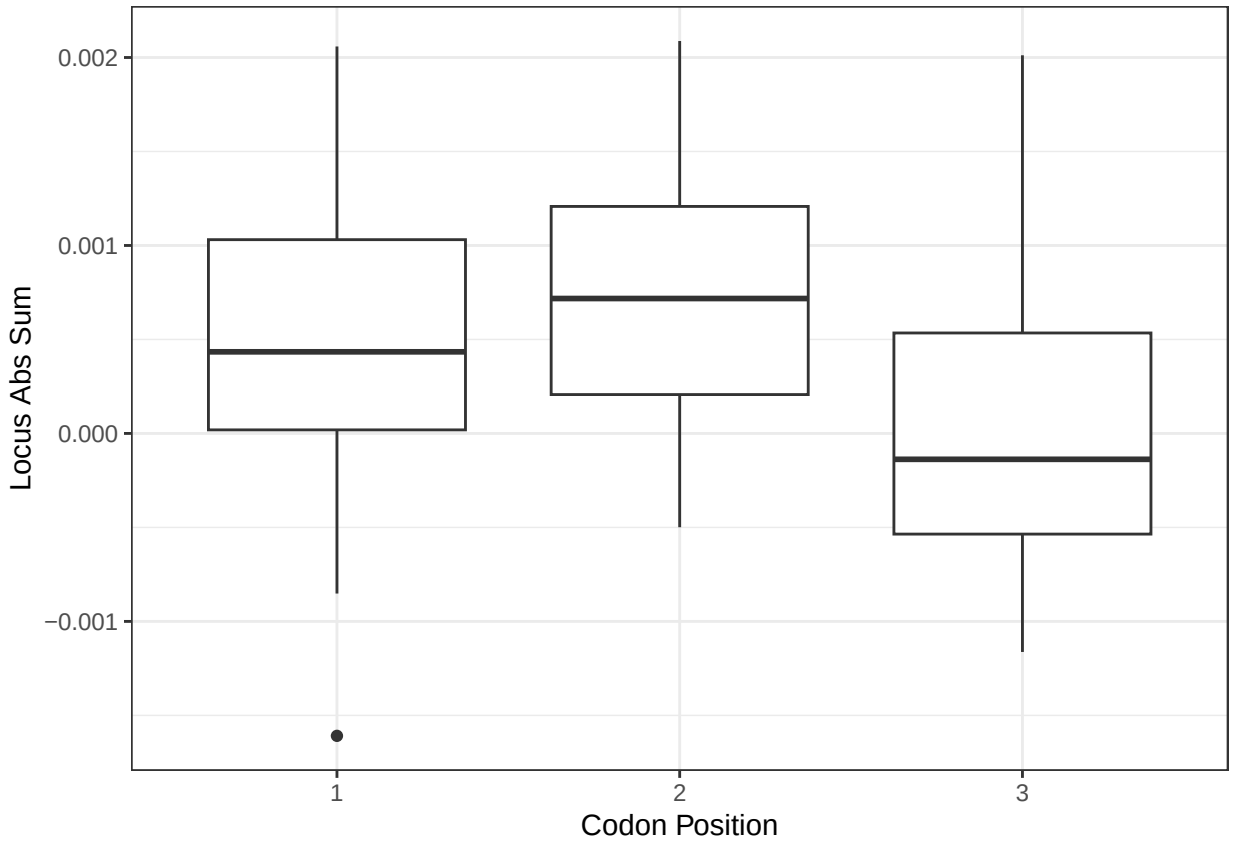

```
ggplot(df2_mod, aes(x = group)) +
  geom_rect(aes(xmin = 30, xmax = 60, ymin = -Inf, ymax = Inf), fill = "grey", alpha = 0.2) +
  # draw points of max
  geom_point(aes(x = group, y = sum_sum), color = "blue") +
  geom_smooth(aes(y = sum_sum, color = "Sum"), method = "auto", se = FALSE) +
  geom_smooth(aes(y = 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")) +
  geom_vline(xintercept = index2, color = "lightgrey", linetype = "dashed") +
  theme_bw() +
  labs(y = "Sum of IG", color = "Statistic", x = "AA Index",
       title = "Compared to abnormal")
```




```r
mean_codon <- sapply(1:3, function(i) {
  mean(df$sum[seq(i, 477, 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 Sum") +
  # x tick only 1,2,3, integer
  scale_x_continuous(breaks = 1:3, minor_breaks = NULL, labels = c("1", "2", "3"))
```




```
codon_data <- data.frame(
  position = rep(1:3, each = length(df$sum) / 3),
  sum = unlist(lapply(1:3, function(i) df$sum[seq(i, 477, by = 3)]))
)
#omit rows where sum ==0
codon_data <- codon_data[codon_data$sum != 0, ]

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




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

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

```
##
## Call:
## lm(formula = sum ~ factor(position), data = codon_data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.110e-03 -5.869e-04 -7.598e-05  5.673e-04  1.945e-03
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.0005013  0.0001950   2.571   0.0127 *
## factor(position)2  0.0001911  0.0002842   0.672   0.5039
## factor(position)3 -0.0004346  0.0002499  -1.739   0.0872 .
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.0008273 on 59 degrees of freedom
## Multiple R-squared:  0.101,  Adjusted R-squared:  0.07049
## F-statistic: 3.313 on 2 and 59 DF,  p-value: 0.04329
```

```r
# ok-mutate based on instance, 100 times, saved to pcss_df
# just use permute_sequence, nothing else
pcss_df <- data.frame(permuted = instance_pasted)
for (i in 1:999) {
  permuted_instance <- permute_sequence(instance, type = "ok", min.subs = 80,
                                        max.subs = 120, 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 = 477, start_ind = 1, vocabulary = c("A", "C", "G", "T"))
})
```

```r
csv_file_path <- "pcssdata_igm.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 = 1:477)

# 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 = 3,
      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)
}
```

```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))

# 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")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).
```