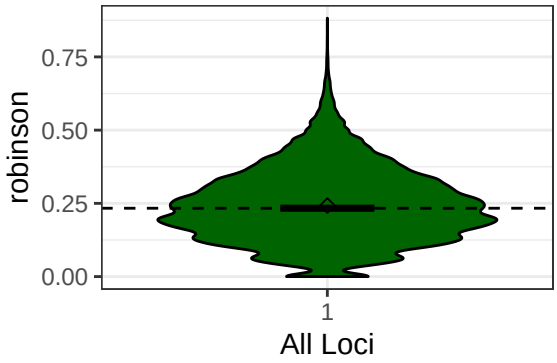
plots

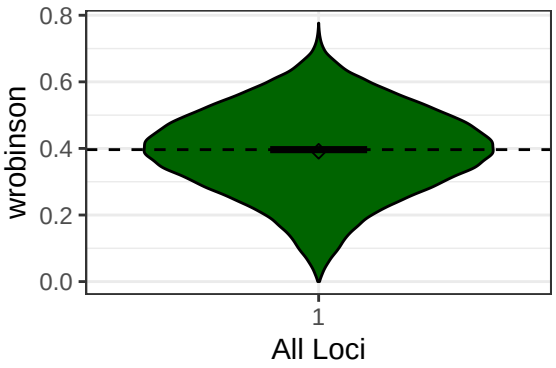
Leann M. Biancani

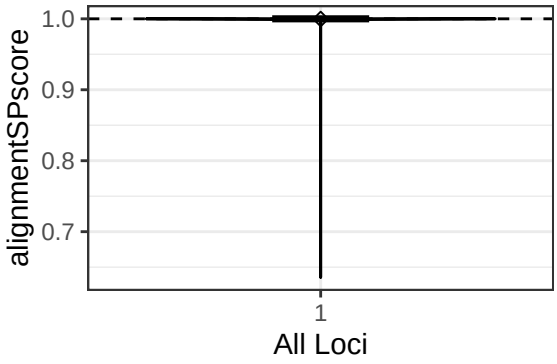
2025-03-04

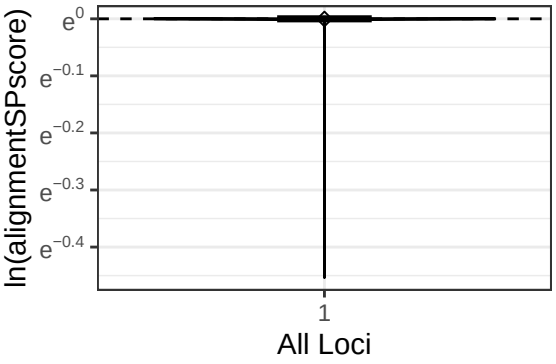
```
# column names
feat_names = colnames(features)[-1]

violin_plot <- function(features, y, fill_color = "darkgreen", logscale = FALSE) {
  # Remove NAs
  #df <- features %>% filter(!is.na(.data[[y]]))
  # Remove NAs and non-positive values for log scale
  df <- features %>% filter(!is.na(.data[[y]]) & (!logscale | .data[[y]] > 0))

  # Define scale transformation conditionally
  scale_y <- if (logscale) {
    scale_y_continuous(trans = "log",
                       breaks = scales::trans_breaks("log", function(x) exp(x)),
                       labels = scales::trans_format("log", scales::math_format(e^.x)))
  } else {
    scale_y_continuous()  # Default linear scale
  }

  # Generate plot
  ggplot(df, aes(x = factor(1), y = .data[[y]], fill = fill_color)) +  # Use factor(1) for a single x v
    theme_bw() +
    geom_violin(color = "black") +  # Violin plot with black border
    geom_hline(yintercept = median(df[[y]], na.rm = TRUE), linetype = "dashed", linewidth = 0.5) +  # D
    stat_summary(fun = median, fun.min = median, fun.max = median,
                 geom = "crossbar", width = 0.25) +  # Crossbar at median
    stat_summary(fun = mean, fun.min = mean, fun.max = mean,
                 geom = "point", shape = 5) +  # Diamond shape for mean
    scale_y +  # Apply the chosen scale (log or linear)
    scale_fill_manual(values = fill_color) +  # Custom fill color
    theme(legend.position = "none") +  # Remove legend
    labs(x = "All Loci", y = ifelse(logscale, paste0("ln(", y, ")"), y))  # Update y label to indicate
}

log_list <- c("alignmentSPscore", "tree_rate", "tree_rate_var",
              "base_composition_variance", "alignment_length", "phyloinf")
counter <- 1
for (i in feat_names) {
  plot <- violin_plot(features, i) #plot the feature
  name <- paste0("p", counter)  #generate a dynamic variable name
  assign(name, plot) #assign a plot to the dynamically named variable
  print(plot)
  if (i %in% log_list) {
```

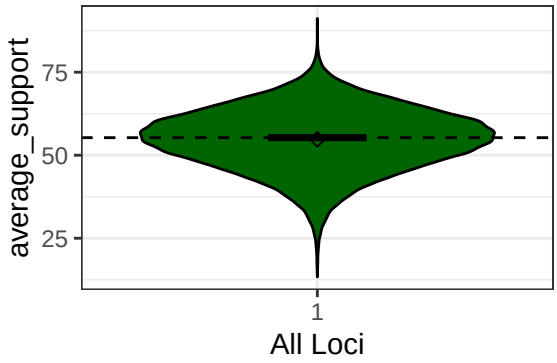
```
    plot <- violin_plot(features, i, fill_color = "green", logscale = TRUE) #plot feature with log scal
    print(plot)
  }
  counter <- counter + 1  #increment counter
}
```

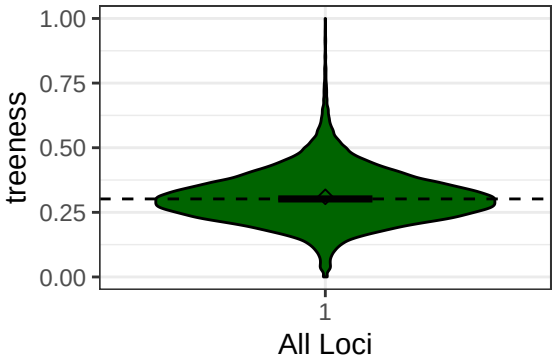


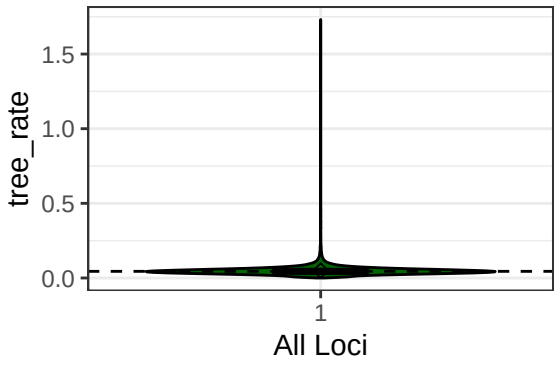


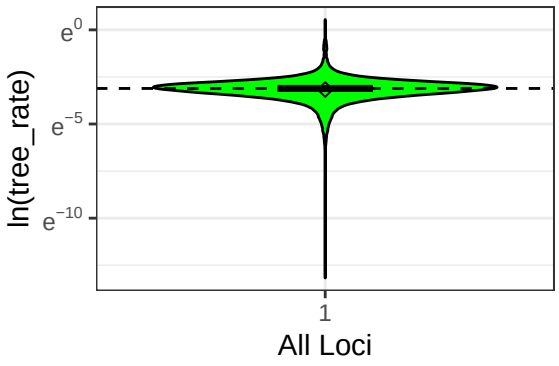


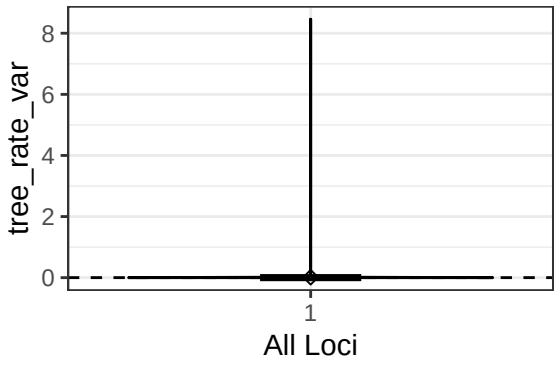


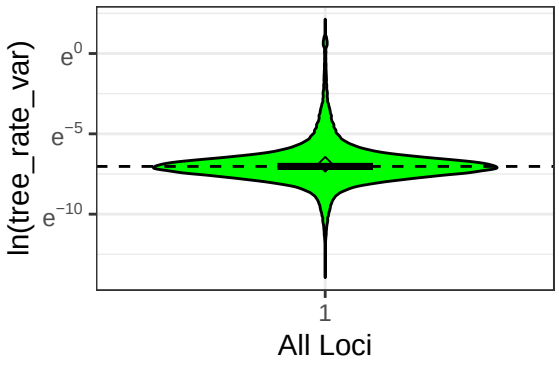


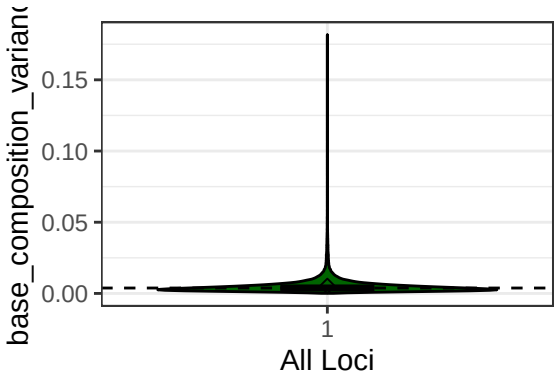


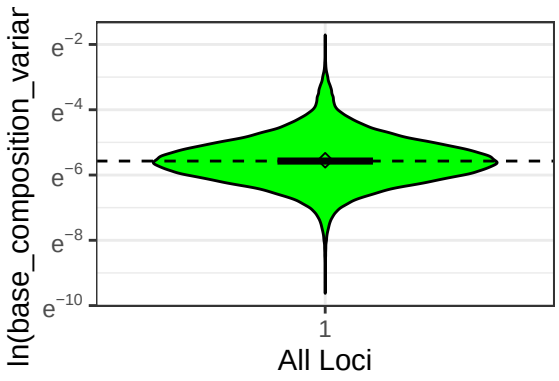

tree_rate
1.5
1.0
0.5
0.0
1
All Loci

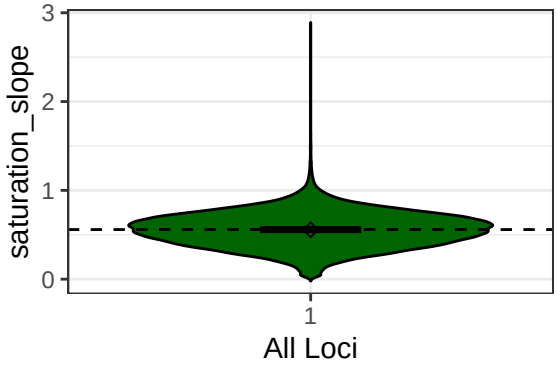

ln(tree_rate)
e^0
e^−5
e^−10
1
All Loci

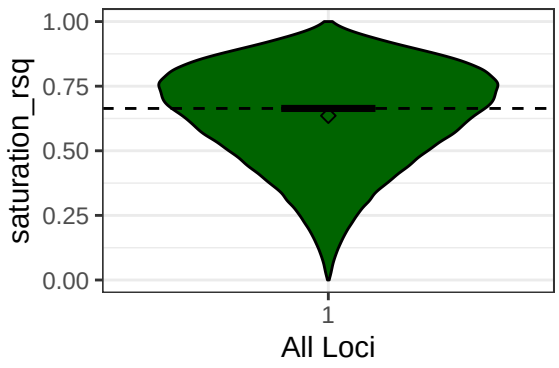

tree_rate_var
8
6
4
2
0
1
All Loci

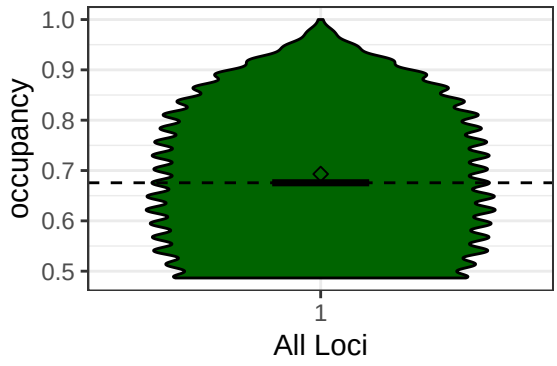

ln(tree_rate_var)
e^0
e^−5
e^−10
1
All Loci

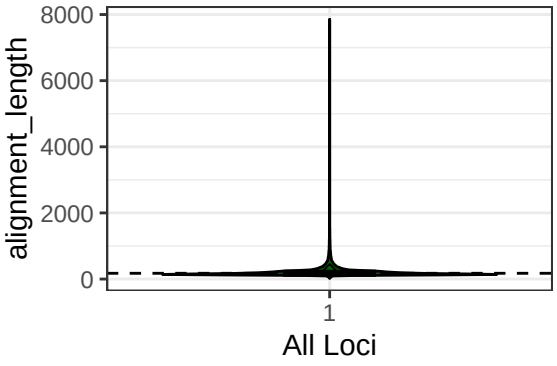

base_composition_varianc
0.15
0.10
0.05
0.00
1
All Loci

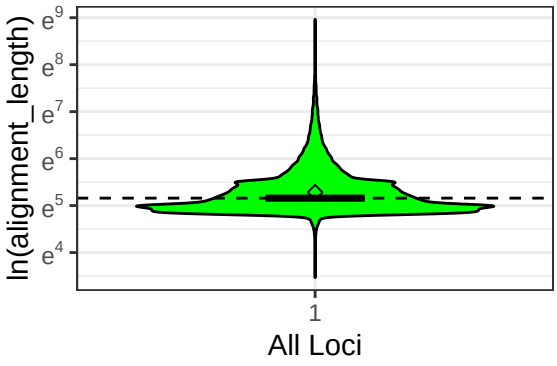

ln(base_composition_variar
e^−2
e^−4
e^−6
e^−8
e^−10
1
All Loci

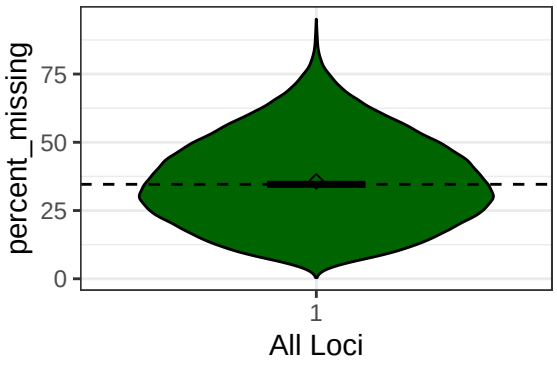

saturation_slope
3
2
1
0
1
All Loci

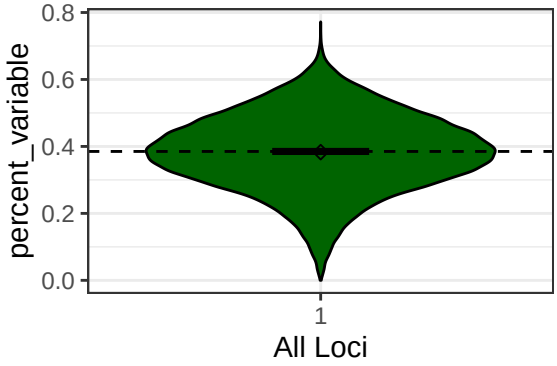

saturation_rsq
1.00
0.75
0.50
0.25
0.00
1
All Loci

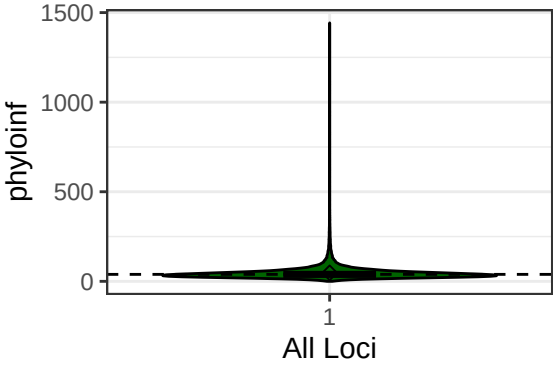

occupancy
1.0
0.9
0.8
0.7
0.6
0.5
1
All Loci

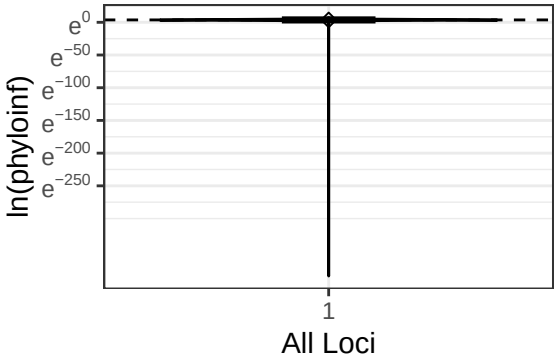

alignment_length
8000
6000
4000
2000
0
1
All Loci


ln(alignment_length)
e9
e8
e7
e6
e5
e4
1
All Loci


percent_missing
75
50
25
0
1
All Loci


percent_variable
0.8
0.6
0.4
0.2
0.0
1
All Loci


phyloinf
1500
1000
500
0
1
All Loci


ln(phyloinf)
e0
e−50
e−100
e−150
e−200
e−250
1
All Loci