Metagenomics K-mer Testing

Contents

Import Libraries

```
library(tidyverse)    # Data manipulation
library(ggpubr)       # ggplot2 extensions
library(egg)          # ggplot2 extensions
library(RColorBrewer) # Color palettes
library(sensitivity)  # Sensitivity analysis
library(lme4)         # Linear mixed-effects models
library(pls)          # Partial least squares regression
library(loadings)     # Loadings for PLS
```

Define functions

```
# Define custom negation for `%in%`
`%notin%` <- Negate(`%in%`)

# Define function to calculate 95%CI
CI95 <- function(stde,sample.n){
  alpha <- 0.05
  degrees.freedom <- sample.n - 1
  t.score <- qt(p=alpha/2, df=degrees.freedom,lower.tail=F)
  CI <- t.score * stde
  return(CI)
}

# Define function to get colors
get_colors <- function(dataDf) {
  # Get variable names
  vars <- unique(dataDf$Variables)

  # Get number of variables
  N_vars <- length(vars)

  # Set the color palette
  colors <- tibble(
    color = brewer.pal(n = N_vars, name = "Dark2"),
    Variables = vars
```

```r
  )

  return(colors)
}

# Define a Function to Format Data
format_data <- function(dataDf, metric){
  # Prepare Data for Sensitivity Analysis
  dataDf_filtered <- dataDf %>%
    filter(Metric == metric) %>%
    select(-Metric) %>%
    unite("KComb", Kmer1, Kmer2, Kmer3, Kmer4, sep = "_", remove = FALSE)

  # Separate Gold Standard and Compute Differences
  dataDf_gs <- dataDf_filtered %>%
    filter(KComb=="GS_GS_GS_GS") %>%
    select(Sample, Value) %>% rename(GS=Value)

  dataDf_exp <- dataDf_filtered %>%
    filter(KComb!="GS_GS_GS_GS") %>%
    left_join(dataDf_gs, by="Sample") %>%
    mutate(Diff = Value - GS)

  return(dataDf_exp)
}

# Define a Function to Perform Sensitivity Analysis
analyze_sensitivity <- function(dataDf, metric) {

  # Prepare Data
  data_exp <- format_data(dataDf, metric)

  # Perform Sensitivity Analysis
  srcOriginal <- NULL
  srcError <- NULL
  srcBias <- NULL

  for (sample in unique(dataDf$Sample)){
    sample_df <- data_exp %>%
      filter(Sample == sample)

    predictors <- sample_df %>% select(Kmer1, Kmer2, Kmer3, Kmer4) %>%
                          mutate(across(everything(), as.numeric))

    targets <- as.numeric(sample_df$Value)

    srcSample <- src(predictors, targets, nboot = 1000, conf = 0.95)
    srcSample_original <- c(sample,as.array(t(srcSample$SRC))[1,])
    srcSample_sterror <- c(sample,as.array(t(srcSample$SRC))[3,])
    srcSample_bias <- c(sample,as.array(t(srcSample$SRC))[2,])

    srcOriginal <- rbind(srcOriginal, srcSample_original)
    srcError <- rbind(srcError, srcSample_sterror)
```

```r
    srcBias <- rbind(srcBias, srcSample_bias)

  }

  srcOriginal <- as.data.frame(srcOriginal)
  srcError <- as.data.frame(srcError)
  srcBias <- as.data.frame(srcBias)

  colnames(srcOriginal)[1] <- "id"
  colnames(srcError)[1] <- "id"
  colnames(srcBias)[1] <- "id"




  return(srcResult <- list(srcOriginal = srcOriginal,
                           srcError = srcError,
                           srcBias = srcBias))
}

# Define a Plotting Function
plot_data <- function(dataDf,title,y_label,colorsDf = get_colors(dataDf)) {

  # Get the number of unique ids
  N_ids <- length(unique(dataDf$id))

  # Gather the data
  dataDf <- dataDf %>% gather(key = "Variables", value = "SRC", -id) %>%
             mutate(Variables = as.factor(Variables),
                    SRC = as.numeric(SRC))

  # Summarize the data
  dataDf_summary <- dataDf %>% group_by(Variables) %>%
                    summarise(mean = mean(SRC), sd = sd(SRC),
                          se=sd/sqrt(N_ids), ci=CI95(se,N_ids),
                    ci2 = list(enframe(Hmisc::smean.cl.boot(SRC))),
                                .groups = "drop") %>%
                    unnest(cols = c(ci2)) %>%
                    spread(name, value) %>%
                    select(-Mean) %>%
                    rename(ci_boot_lower = Lower, ci_boot_upper = Upper)

  # Join the data
  dataDf <- dataDf %>%
            left_join(dataDf_summary, by = c("Variables" = "Variables")) %>%
            mutate(mean = as.numeric(mean),
                   sd = as.numeric(sd),
                   se = as.numeric(se)) %>%
            left_join(colorsDf, by = c("Variables" = "Variables"))

  # Set Plotting Parameters
  alph <- 0.4
  pt.size <- 0.8
```

```r
  pos.jitter <- 0.2
  mean.size <- 3
  mean.shape <- 18
  cex <- 0.8

  ggplot(dataDf, aes(x = Variables, y = SRC)) +
    geom_point(position = position_jitter(pos.jitter), col = dataDf$color,
               alpha = alph, size = pt.size) +
    geom_point(aes(y = mean), col = dataDf$color,
               size = mean.size, shape = mean.shape) +
    geom_errorbar(aes(ymin=ci_boot_lower, ymax=ci_boot_upper),
                  width=0.1, col = dataDf$color ) +
    labs(title = title, x = element_blank(), y = y_label) +
    theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
          legend.position = "none") +
    theme_pubclean()
}
```

Load Data

```r
# Set Working Directory
result_tsv <- "../../Results/Metaquast_summary.tsv"

# Load Data
data <- read_tsv(result_tsv, show_col_types = FALSE) %>%
            pivot_longer(cols = -Metric,
                         names_to = "Sample",
                         values_to = "Value") %>%
            separate(Sample, into = c("Sample", "Kmers"), sep = "_") %>%
            separate(Kmers,
                     into = c("Kmer1", "Kmer2", "Kmer3", "Kmer4"),
                     sep = "-") %>%
            mutate(Kmer1 = str_replace(Kmer1, "k", ""),
                   Sample = str_replace(Sample, "S", ""))
```

Sensitivity Analysis

Total Length

```r
src_TL <- analyze_sensitivity(data, "Total length")

# Plot Original, Bias, and Error
annotate_figure(
  ggarrange(ncol = 3, nrow = 1,
            plot_data(src_TL$srcOriginal, "Original SRC", "SRC"),
            plot_data(src_TL$srcBias, "Bias SRC", "Bias"),
            plot_data(src_TL$srcError, "Std Error for SRC", "Std Err")),
  top = text_grob("Total length", color = "black",
                  face = "bold", size = 14)
)
```

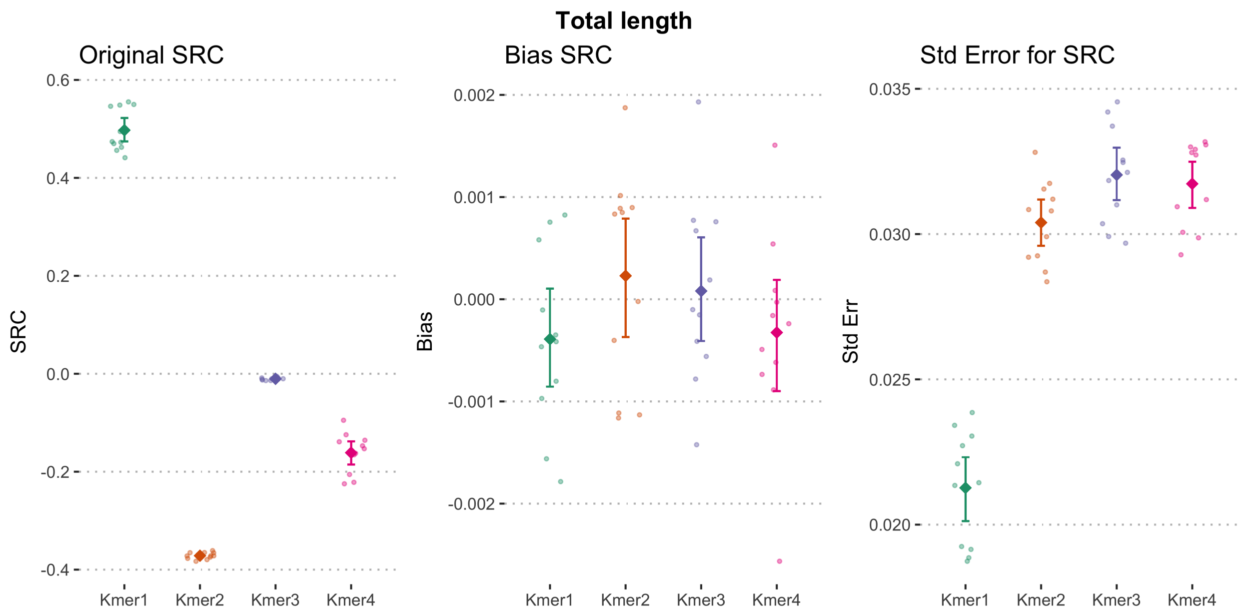


Figure 1: Sensitivity analysis for Total Length metric. Bars represent the 95% confidence interval with boostrap for the mean.

N50

```
src_N50 <- analyze_sensitivity(data, "N50")

# Plot Original, Bias, and Error
annotate_figure(
  ggarrange(ncol = 3, nrow = 1,
            plot_data(src_N50$srcOriginal, "Original SRC", "SRC"),
            plot_data(src_N50$srcBias, "Bias SRC", "Bias"),
            plot_data(src_N50$srcError, "Std Error for SRC", "Std Err")),
  top = text_grob("N50", color = "black",
                    face = "bold", size = 14)
)
```

GC Content

```
src_gc <- analyze_sensitivity(data, "GC (%)")

# Plot Original, Bias, and Error
annotate_figure(
  ggarrange(ncol = 3, nrow = 1,
            plot_data(src_gc$srcOriginal, "Original SRC", "SRC"),
            plot_data(src_gc$srcBias, "Bias SRC", "Bias"),
            plot_data(src_gc$srcError, "Std Error for SRC", "Std Err")),
  top = text_grob("GC Content", color = "black",
                    face = "bold", size = 14)
)
```

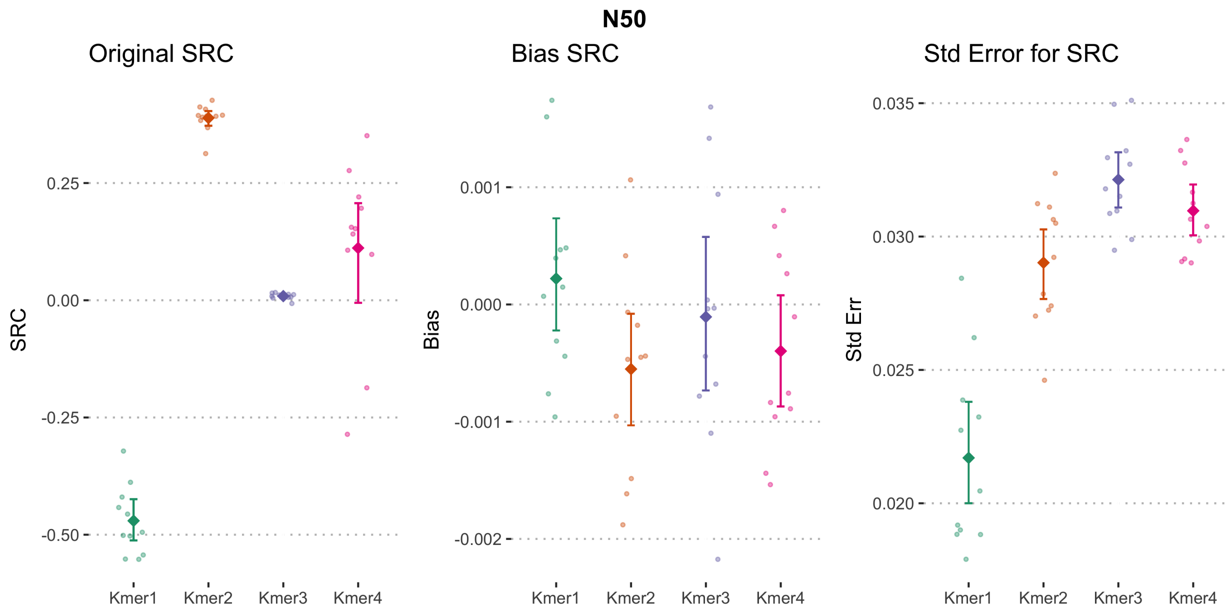


Figure 2: Sensitivity analysis for N50 metric. Bars represent the 95% confidence interval with boostrap for the mean.

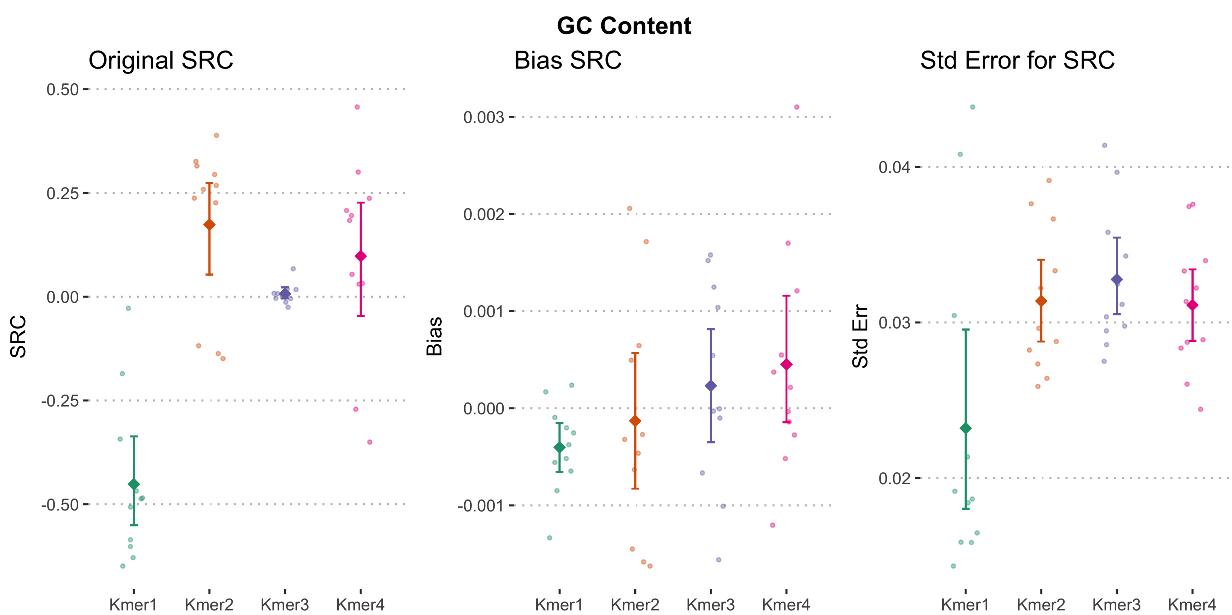


Figure 3: Sensitivity analysis for GC Content metric. Bars represent the 95% confidence interval with bootstrap for the mean.

Number of contigs

```
src_contig <- analyze_sensitivity(data, "# contigs")

# Plot Original, Bias, and Error
annotate_figure(
  ggarrange(ncol = 3, nrow = 1,
            plot_data(src_contig$srcOriginal, "Original SRC", "SRC"),
            plot_data(src_contig$srcBias, "Bias SRC", "Bias"),
            plot_data(src_contig$srcError, "Std Error for SRC", "Std Err")),
  top = text_grob("Number of Contigs", color = "black",
                  face = "bold", size = 14)
)
```

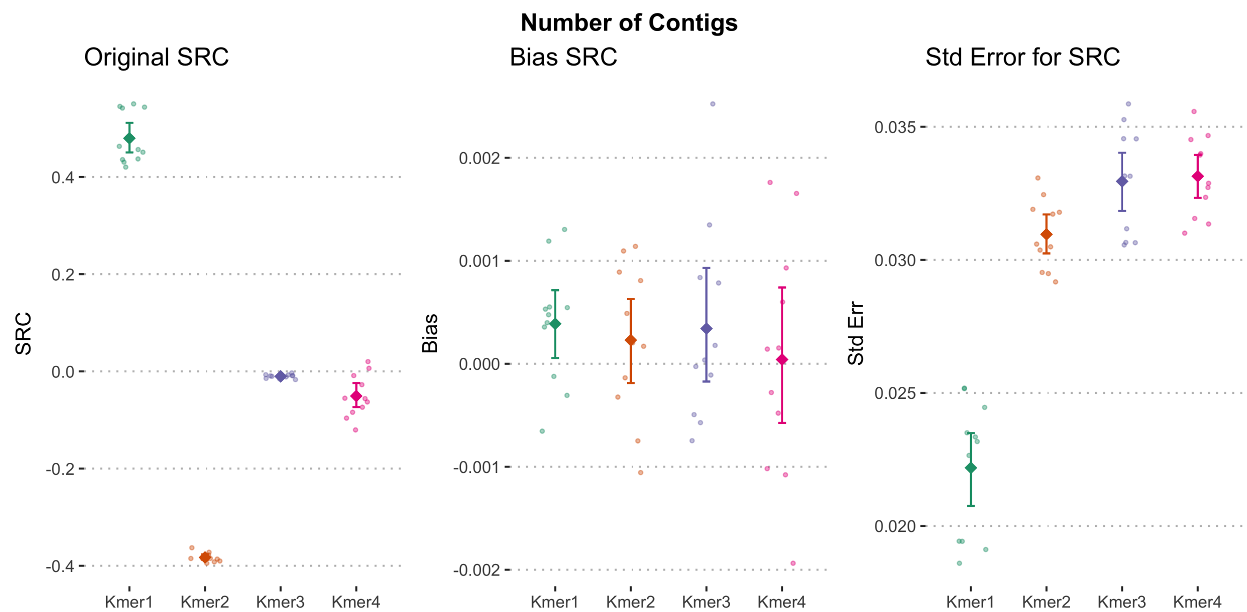


Figure 4: Sensitivity analysis for Number of Contigs metric. Bars represent the 95% confidence interval with bootstrap for the mean.

Unified Sensitivity Analysis

```
srcUnified <- mutate(pivot_longer(src_TL$srcOriginal, cols = !id,
                                  names_to = "Variables", values_to = "SRC"),
                     Metric = "Total Length") %>%
  bind_rows(mutate(pivot_longer(src_N50$srcOriginal, cols = !id,
                                names_to = "Variables", values_to = "SRC"),
                   Metric = "N50")) %>%
  bind_rows(mutate(pivot_longer(src_gc$srcOriginal, cols = !id,
                                names_to = "Variables", values_to = "SRC"),
                   Metric = "GC Content")) %>%
  bind_rows(mutate(pivot_longer(src_contig$srcOriginal, cols = !id,
                                names_to = "Variables", values_to = "SRC"),
                   Metric = "Number of Contigs")) %>%
  mutate(Variables = as.factor(Variables),
         Metric = factor(Metric, levels=c("Total Length", "Number of Contigs",
```

```
                                      "N50","GC Content")),
         SRC = as.numeric(SRC))

# Get the number of unique ids
N_ids <- length(unique(srcUnified$id))

# Summarize the data
srcUnified_summary <- srcUnified %>% group_by(Variables, Metric) %>%
                      summarise(mean = mean(SRC), sd = sd(SRC),
                                se=sd/sqrt(N_ids), ci=CI95(se,N_ids),
                                ci2 = list(enframe(Hmisc::smean.cl.boot(SRC))),
                                  .groups = "drop") %>%
                      unnest(cols = c(ci2)) %>%
                      spread(name, value) %>%
                      select(-Mean) %>%
                      rename(ci_boot_lower = Lower, ci_boot_upper = Upper)


# Join the data
dataDf <- srcUnified %>%
            left_join(srcUnified_summary, by = c("Variables", "Metric")) %>%
            mutate(mean = as.numeric(mean),
                   sd = as.numeric(sd),
                   se = as.numeric(se)) %>%
            left_join(get_colors(srcUnified), by = c("Variables"))

# Set Plotting Parameters
alph <- 0.4
pt.size <- 0.8
pos.jitter <- 0.2
mean.size <- 3
mean.shape <- 18
cex <- 0.8

plot <- ggplot(dataDf, aes(x = Variables, y = SRC)) +
  geom_point(position = position_jitter(pos.jitter),
             col = dataDf$color, alpha = alph, size = pt.size) +
  geom_point(aes(y = mean), col = dataDf$color,
             size = mean.size, shape = mean.shape) +
  geom_errorbar(aes(ymin=ci_boot_lower, ymax=ci_boot_upper),
                width=.1, col = dataDf$color ) +
  labs(title = element_blank(), x = element_blank()) +
  theme_pubclean() +
  facet_wrap(~Metric, ncol = 4) +
  theme(panel.border = element_rect(fill = NA, color = "black"),
        panel.spacing = unit(0, "lines"))

tag_facet(plot, x = -Inf, y = Inf, hjust = -0.5,
          open = "", close = ")", tag_pool = LETTERS) +
    theme(strip.text = element_text(colour = "black",
                                    face = "bold",
                                    vjust = 1,
                                    margin = margin(0.2,0,0.2,0, "cm")))
```

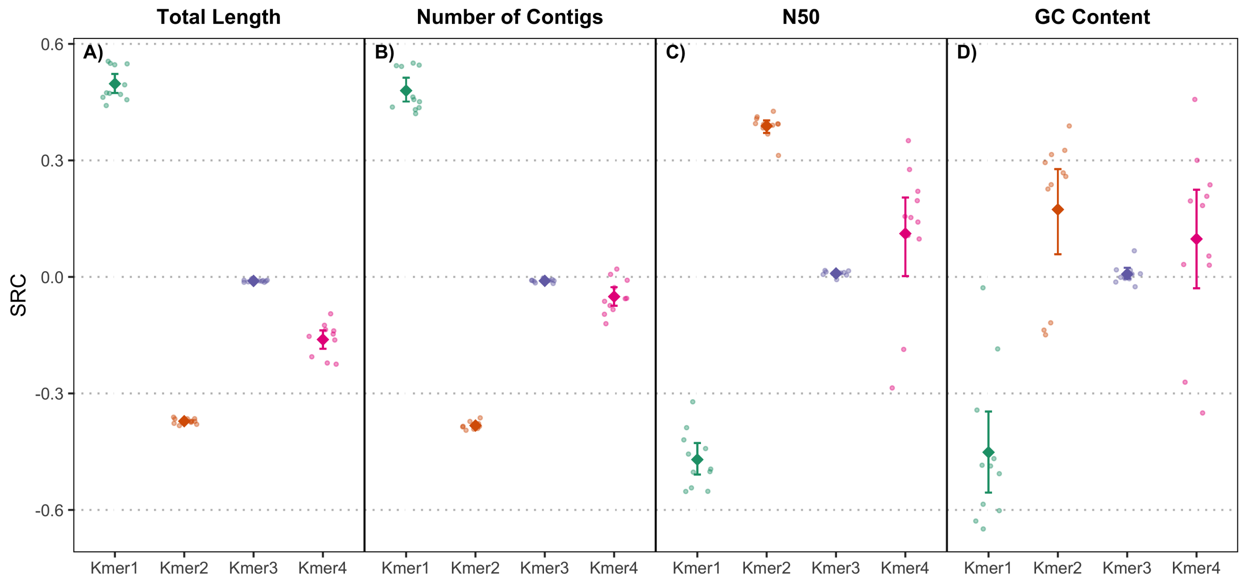


Figure 5: Unified sensitivity analysis for all metrics. Bars represent the 95% confidence interval with bootstrap for the mean.

PLS Regression

Data Preparation

```
# Filter data for a specific metric
data_TL <- format_data(data, "Total length") %>%
              mutate(Diff_TL = Diff, .keep="unused") %>%
              select(-Value, -GS, -KComb)
data_N50 <- format_data(data, "N50") %>%
              mutate(Diff_N50 = Diff, .keep="unused") %>%
              select(-Value, -GS, -KComb)
data_GC <- format_data(data, "GC (%)") %>%
              mutate(Diff_GC = Diff, .keep="unused") %>%
              select(-Value, -GS, -KComb)
data_contigs <- format_data(data, "# contigs") %>%
              mutate(Diff_contigs = Diff, .keep="unused") %>%
              select(-Value, -GS, -KComb)

pls_data <- data_TL %>%
              left_join(data_N50, by = join_by(Sample, Kmer1, Kmer2,
                                               Kmer3, Kmer4)) %>%
              left_join(data_GC, by = join_by(Sample, Kmer1, Kmer2,
                                              Kmer3, Kmer4)) %>%
              left_join(data_contigs, by = join_by(Sample, Kmer1, Kmer2,
                                                   Kmer3, Kmer4)) %>%
              mutate(across(c(Kmer1, Kmer2, Kmer3, Kmer4), as.numeric)) %>%
              mutate(Std_Diff_TL = Diff_TL/sd(Diff_TL),
                     Std_Diff_N50 = Diff_N50/sd(Diff_N50),
                     Std_Diff_GC = Diff_GC/sd(Diff_GC),
                     Std_Diff_contigs = Diff_contigs/sd(Diff_contigs))
```

Model Fitting

```r
# Separate predictors and responses
predictors <- pls_data %>% select(Kmer1, Kmer2, Kmer3, Kmer4)
responses <- pls_data %>% select(Diff_TL, Diff_N50, Diff_GC, Diff_contigs)
responses <- pls_data %>% select(Std_Diff_TL, Std_Diff_N50,
                                 Std_Diff_GC, Std_Diff_contigs)

# Fit the PLS model with cross-validation
pls_model <- plsr(as.matrix(responses) ~ as.matrix(predictors),
                  data = pls_data, validation = "LOO")

# Summary of the PLS model
summary(pls_model)

# Loadings
loadings(pls_model)

Yloadings(pls_model)

loading.weights(pls_model)

# Display the explained variance
explvar(pls_model)
```

```
## Data:    X dimension: 6600 4
##  Y dimension: 6600 4
## Fit method: kernelpls
## Number of components considered: 4
##
## VALIDATION: RMSEP
## Cross-validated using 6600 leave-one-out segments.
##
## Response: Std_Diff_TL
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9988   0.9989    0.999   0.9991
## adjCV            1   0.9988   0.9989    0.999   0.9991
##
## Response: Std_Diff_N50
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9946   0.9948   0.9948   0.9948
## adjCV            1   0.9946   0.9948   0.9948   0.9948
##
## Response: Std_Diff_GC
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9993   0.9995   0.9997   0.9997
## adjCV            1   0.9993   0.9995   0.9997   0.9997
##
## Response: Std_Diff_contigs
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9946   0.9947   0.9946   0.9946
## adjCV            1   0.9946   0.9947   0.9946   0.9946
##
## TRAINING: % variance explained
```

```
##                   1 comps  2 comps  3 comps   4 comps
## X                 22.9529  48.1977  74.5754  100.0000
## Std_Diff_TL        0.3156   0.3156   0.3156    0.3156
## Std_Diff_N50       1.1626   1.1771   1.1799    1.1799
## Std_Diff_GC        0.1906   0.1908   0.1993    0.1994
## Std_Diff_contigs   1.1835   1.2180   1.2198    1.2198
##
## Loadings:
##                             Comp 1 Comp 2 Comp 3 Comp 4
## as.matrix(predictors)Kmer1  0.799 -0.177 -0.589
## as.matrix(predictors)Kmer2 -0.540  0.166 -0.807
## as.matrix(predictors)Kmer3                       0.999
## as.matrix(predictors)Kmer4 -0.267 -0.970
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings     1.001   1.00   1.00   1.00
## Proportion Var  0.250   0.25   0.25   0.25
## Cumulative Var  0.250   0.50   0.75   1.00
##
## Loadings:
##                  Comp 1 Comp 2 Comp 3 Comp 4
## Std_Diff_TL
## Std_Diff_N50
## Std_Diff_GC
## Std_Diff_contigs
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings     0.002      0      0      0
## Proportion Var  0.000      0      0      0
## Cumulative Var  0.000      0      0      0
##
## Loadings:
##                             Comp 1 Comp 2 Comp 3 Comp 4
## as.matrix(predictors)Kmer1  0.789 -0.174 -0.589
## as.matrix(predictors)Kmer2 -0.565  0.170 -0.807
## as.matrix(predictors)Kmer3                       0.999
## as.matrix(predictors)Kmer4 -0.240 -0.970
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
##   Comp 1   Comp 2   Comp 3   Comp 4
## 22.95286 25.24483 26.37772 25.42459
```

```r
# Separate predictors and responses
predictors <- pls_data %>% select(Kmer1, Kmer2, Kmer3, Kmer4)
responses <- pls_data %>% select(Std_Diff_TL, Std_Diff_N50,
                                 Std_Diff_GC, Std_Diff_contigs)

# Fit the PLS model with cross-validation
pls_model_std <- plsr(as.matrix(responses) ~ as.matrix(predictors),
                      data = pls_data, validation = "LOO")
```

```r
# Summary of the PLS model
summary(pls_model_std)

# Loadings
loadings(pls_model_std)

Yloadings(pls_model_std)

loading.weights(pls_model_std)

# Display the explained variance
explvar(pls_model_std)
```

Standardized Metrics

```
## Data:    X dimension: 6600 4
##  Y dimension: 6600 4
## Fit method: kernelpls
## Number of components considered: 4
##
## VALIDATION: RMSEP
## Cross-validated using 6600 leave-one-out segments.
##
## Response: Std_Diff_TL
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9988   0.9989    0.999   0.9991
## adjCV            1   0.9988   0.9989    0.999   0.9991
##
## Response: Std_Diff_N50
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9946   0.9948   0.9948   0.9948
## adjCV            1   0.9946   0.9948   0.9948   0.9948
##
## Response: Std_Diff_GC
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9993   0.9995   0.9997   0.9997
## adjCV            1   0.9993   0.9995   0.9997   0.9997
##
## Response: Std_Diff_contigs
##        (Intercept)  1 comps  2 comps  3 comps  4 comps
## CV               1   0.9946   0.9947   0.9946   0.9946
## adjCV            1   0.9946   0.9947   0.9946   0.9946
##
## TRAINING: % variance explained
##                   1 comps  2 comps  3 comps   4 comps
## X                 22.9529  48.1977  74.5754  100.0000
## Std_Diff_TL        0.3156   0.3156   0.3156    0.3156
## Std_Diff_N50       1.1626   1.1771   1.1799    1.1799
## Std_Diff_GC        0.1906   0.1908   0.1993    0.1994
## Std_Diff_contigs   1.1835   1.2180   1.2198    1.2198
##
## Loadings:
##                            Comp 1 Comp 2 Comp 3 Comp 4
## as.matrix(predictors)Kmer1  0.799 -0.177 -0.589
```

```
## as.matrix(predictors)Kmer2 -0.540  0.166 -0.807
## as.matrix(predictors)Kmer3                      0.999
## as.matrix(predictors)Kmer4 -0.267 -0.970
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings     1.001   1.00   1.00   1.00
## Proportion Var  0.250   0.25   0.25   0.25
## Cumulative Var  0.250   0.50   0.75   1.00
##
## Loadings:
##                  Comp 1 Comp 2 Comp 3 Comp 4
## Std_Diff_TL
## Std_Diff_N50
## Std_Diff_GC
## Std_Diff_contigs
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings     0.002      0      0      0
## Proportion Var  0.000      0      0      0
## Cumulative Var  0.000      0      0      0
##
## Loadings:
##                            Comp 1 Comp 2 Comp 3 Comp 4
## as.matrix(predictors)Kmer1  0.789 -0.174 -0.589
## as.matrix(predictors)Kmer2 -0.565  0.170 -0.807
## as.matrix(predictors)Kmer3                      0.999
## as.matrix(predictors)Kmer4 -0.240 -0.970
##
##                Comp 1 Comp 2 Comp 3 Comp 4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
##   Comp 1   Comp 2   Comp 3   Comp 4
## 22.95286 25.24483 26.37772 25.42459
```

Cross-validation Results

```r
# Plot the cross-validation results to determine the optimal number of components
validationplot(pls_model, val.type = "RMSEP", legendpos = "bottomright",
               intercept = TRUE, main = "Cross-validation Results", nCols=2)
```

Optimal coefficients

```r
# Get the coefficients of the PLS model for the optimal number of components
optimal_components <- 4
pls_coefficients <- coef(pls_model, ncomp = optimal_components)

# Print the coefficients
print(pls_coefficients)

# Plot the coefficients
plot(pls_coefficients, main = "PLS Coefficients")
```

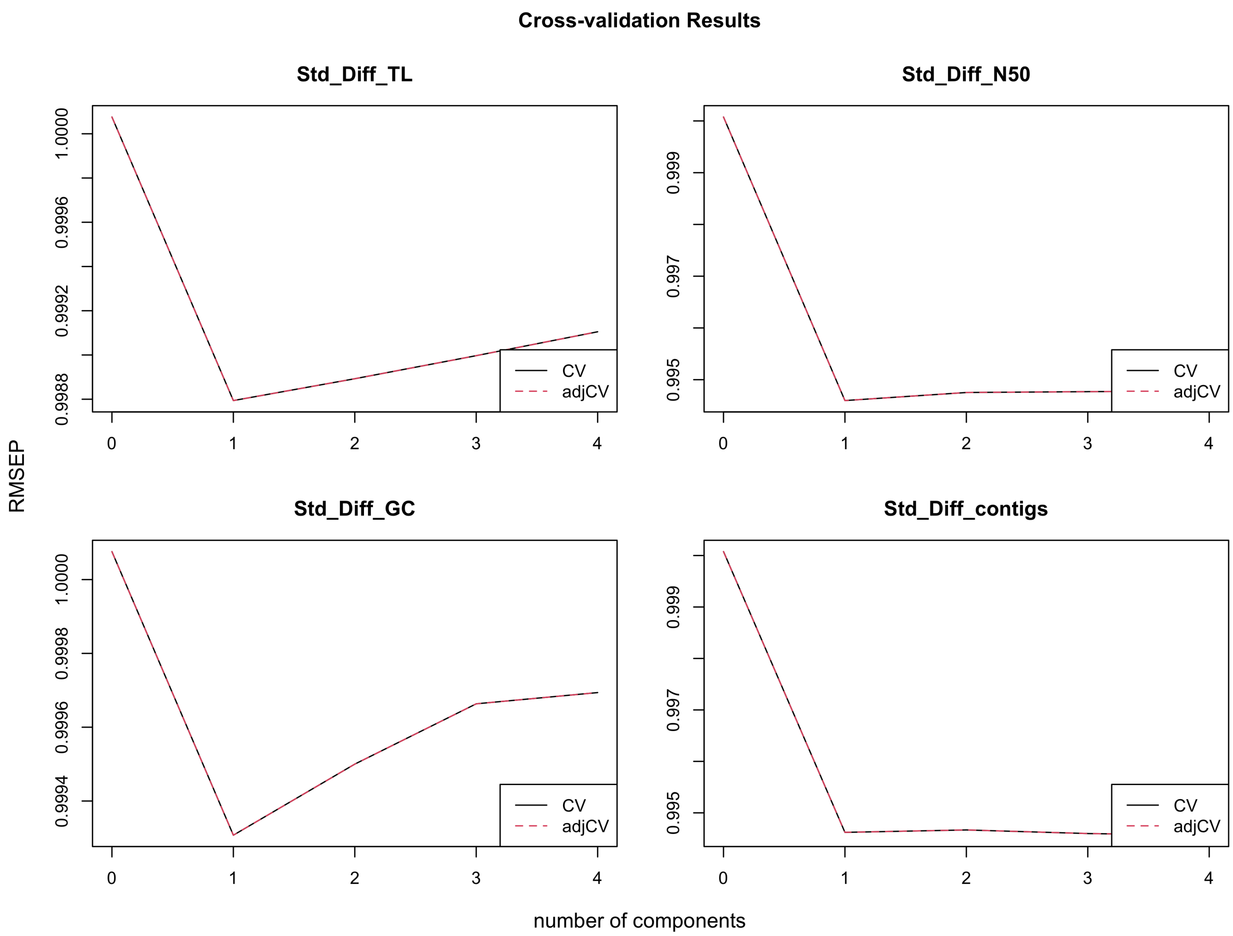


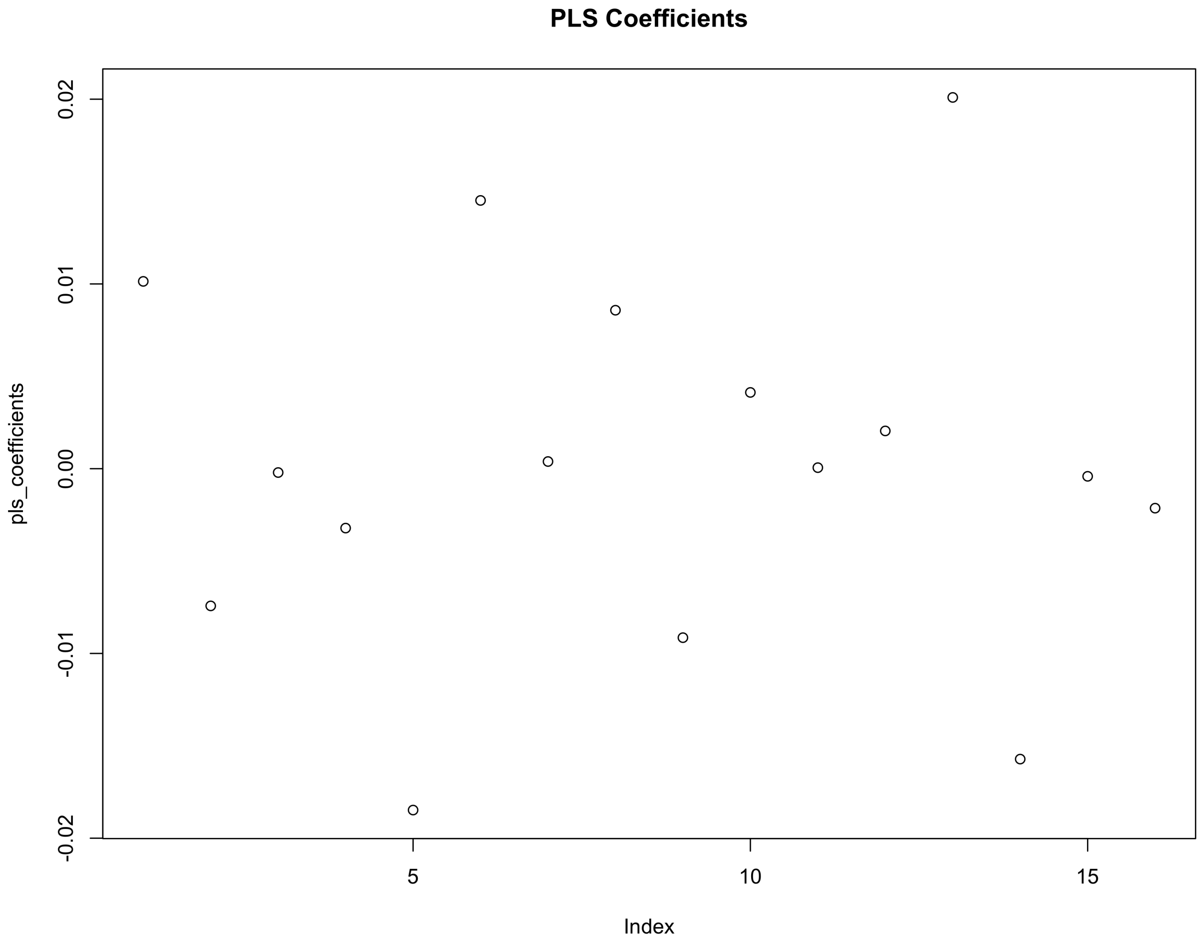
Figure 6: Cross-validation results for the PLS model. The plot shows the Root Mean Squared Error of Prediction (RMSEP) for different numbers of components.

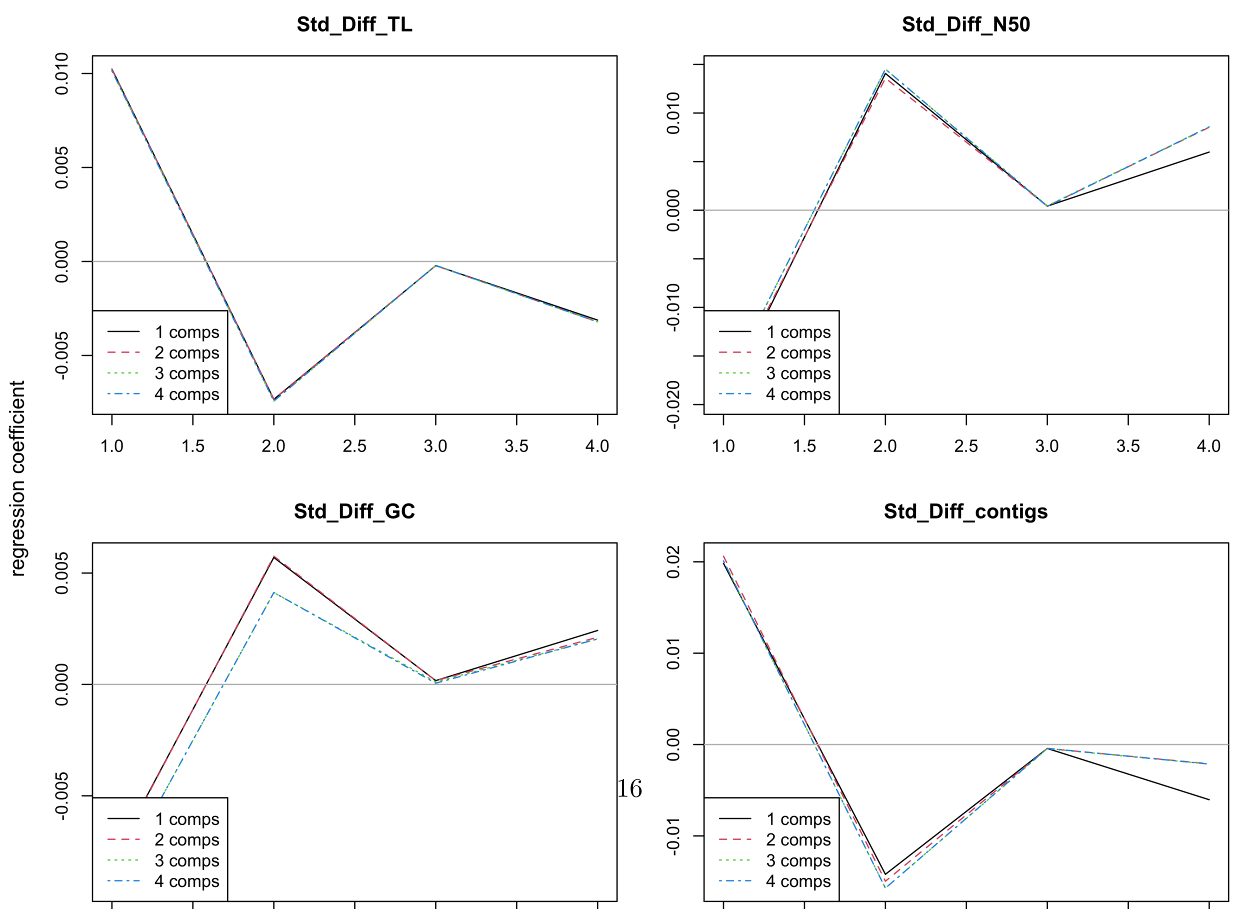
```r
plot(pls_model, plottype = "coef", ncomp=1:4,
     legendpos = "bottomleft", nCols=2 )
```

```
## , , 4 comps
##
##                               Std_Diff_TL  Std_Diff_N50   Std_Diff_GC
## as.matrix(predictors)Kmer1  0.0101382850 -0.0184773430 -9.145968e-03
## as.matrix(predictors)Kmer2 -0.0074260894  0.0145194096  4.127577e-03
## as.matrix(predictors)Kmer3 -0.0002079648  0.0003879909  5.636515e-05
## as.matrix(predictors)Kmer4 -0.0032152612  0.0085773299  2.044518e-03
##                            Std_Diff_contigs
## as.matrix(predictors)Kmer1     0.0200944145
## as.matrix(predictors)Kmer2    -0.0157185305
## as.matrix(predictors)Kmer3    -0.0004133761
## as.matrix(predictors)Kmer4    -0.0021368809
```

Optimal k-mer combination

```r
# Define the error function
calculate_error <- function(kmer_values) {
  # Create a data frame with k-mer values for prediction
  new_data <- data.frame(
    Kmer1 = kmer_values[1],
    Kmer2 = kmer_values[2],
    Kmer3 = kmer_values[3],
    Kmer4 = kmer_values[4]
  )

  # Predict deviations using the PLS model
  optimal_components <- 2  # Set based on cross-validation results
  predicted_deviation <- predict(pls_model, newdata = as.matrix(new_data), ncomp = optimal_components)

  # Calculate total error as the sum of absolute deviations
  total_error <- sum(abs(predicted_deviation))

  return(total_error)
}

# Set initial values for Kmer1, Kmer2, Kmer3, and Kmer4
initial_kmers <- c(21, 33, 55, 77)

# Define bounds for each k-mer (if needed)
lower_bounds <- c(15, 27, 49, 71)  # Minimum k-mer values
upper_bounds <- c(27, 39, 61, 83)  # Maximum k-mer values

# Run optimization to minimize the error function
optimal_kmers <- optim(
  par = initial_kmers,
  fn = calculate_error,
  method = "L-BFGS-B",  # Allows for bounds on parameters
  lower = lower_bounds,
  upper = upper_bounds
)
```


PLS Coefficients
pls_coefficients
Index
0.02
0.01
0.00
-0.01
-0.02
5
10
15


Std_Diff_TL
Std_Diff_N50
Std_Diff_GC
Std_Diff_contigs
regression coefficient
1 comps
2 comps
3 comps
4 comps

```r
# Print the optimal k-mer combination
optimal_kmers$par
```

Direct evaluation

```
## [1] 27 27 49 71
```

```r
# Step 1: Generate all possible k-mer combinations
Kmer1_values <- c(15, 17, 21, 25, 27)
Kmer2_values <- c(27, 29, 33, 37, 39)
Kmer3_values <- c(49, 51, 55, 59, 61)
Kmer4_values <- c(71, 73, 77, 81, 83)

kmer_combinations <- expand.grid(
  Kmer1 = Kmer1_values,
  Kmer2 = Kmer2_values,
  Kmer3 = Kmer3_values,
  Kmer4 = Kmer4_values
)

### Step 2: Predict Assembly Metrics
optimal_components <- 2  # Set based on cross-validation results

# Predict assembly metrics for all k-mer combinations
predicted_metrics <- predict(
  pls_model,
  newdata = as.matrix(kmer_combinations),
  ncomp = optimal_components
)

# Convert predicted metrics to a data frame
predicted_metrics_df <- as.data.frame(predicted_metrics)

# Name the columns appropriately
colnames(predicted_metrics_df) <- c("Pred_Diff_TL", "Pred_Diff_N50", "Pred_Diff_GC", "Pred_Diff_contigs

# Combine the k-mer combinations with the predicted metrics
results_df <- cbind(kmer_combinations, predicted_metrics_df)

### Step 3: Define the Objective Function

# Calculate standard deviations from training data
sd_TL <- sd(pls_data$Diff_TL)
sd_N50 <- sd(pls_data$Diff_N50)
sd_contigs <- sd(pls_data$Diff_contigs)
sd_GC <- sd(pls_data$Diff_GC)

# Standardize the predicted deviations
results_df$Std_Pred_Diff_TL <- results_df$Pred_Diff_TL / sd_TL
results_df$Std_Pred_Diff_N50 <- results_df$Pred_Diff_N50 / sd_N50
results_df$Std_Pred_Diff_contigs <- results_df$Pred_Diff_contigs / sd_contigs
results_df$Std_Pred_Diff_GC <- results_df$Pred_Diff_GC / sd_GC
```

```r
# Define weights based on priorities
w_N50 <- 1      # Maximize N50
w_contigs <- 1  # Maximize the number of contigs
w_TL <- 1       # Maximize total length
w_GC <- 1       # Not prioritizing GC content

# Calculate the score
results_df$Score <- (w_N50 * results_df$Std_Pred_Diff_N50) +
                    (w_contigs * results_df$Std_Pred_Diff_contigs) +
                    (w_TL * results_df$Std_Pred_Diff_TL) +
                    (w_GC * results_df$Std_Pred_Diff_GC)

### Step 4: Select the Optimal K-mer Combination

# Find the combination with the maximum score
best_combination <- results_df[which.max(results_df$Score), ]

# Print the best combination and its predicted metrics
print(best_combination)
```

Using a grid search

```
##     Kmer1 Kmer2 Kmer3 Kmer4 Pred_Diff_TL Pred_Diff_N50 Pred_Diff_GC
## 621    15    39    61    83    -2.773896     0.5487886    0.5216348
##     Pred_Diff_contigs Std_Pred_Diff_TL Std_Pred_Diff_N50 Std_Pred_Diff_contigs
## 621         -2.531692    -5.863957e-08      0.0001072036         -9.373218e-05
##     Std_Pred_Diff_GC    Score
## 621         1.079227 1.079241
```