MGE Data Processing

Introduction

This document was generated by concatenating text with the output of code blocks. In the first code block, several commonly used library packages are loaded. These are generic tools that simplify slicing and dicing of complex data.

In the second code block, tab-separated tables, which are the output of the barcode validator combined with analytics files and parameter settings, are read into a single data frame. Some of the contents are processed further, for example:

- all values that are `None` (the empty variable in Python) are parsed as `NA`, i.e. missing data in R.
- numerous columns that are counts, such as number of stop codons, ambiguous characters, or sequence lengths, are parsed as integers
- some column, whose values were wrapped in brackets (because they were YAML lists) have their brackets removed
- some input files indicated whether they were processed as `standard` or using `fastp`. This was inserted in the `method` column.

The end result is a single data frame suitable for further analysis.

```
tsv_files <- list.files(path = "../data",
                        pattern = "^mge_.*\\.tsv$",
                        full.names = TRUE)

clean_bracketed_value <- function(x) {
  gsub("\\[|\\]|'", "", x)
}

process_tsv <- function(file_path) {
  method <- gsub("^mge_([^_]+)_.*\\.tsv$", "\\1", basename(file_path))

  df <- read_tsv(file_path, na = "None", show_col_types = FALSE) %>%
    mutate(
      genes = clean_bracketed_value(genes),
      r = as.numeric(clean_bracketed_value(r)),
      s = as.integer(clean_bracketed_value(s)),
      method = method,
      across(any_of(c("ambig_basecount", "ambig_full_basecount", "length",
                      "n_aligned", "n_reads", "nuc_basecount",
                      "nuc_full_basecount", "stop_codons")), as.integer)
    )

  return(df)
}

combined_data <- map_df(tsv_files, process_tsv)
```

In this code block, a column `is_valid` is created, which gives a boolean `TRUE` if:

- the barcode length is >= 500
- there are no stop codons
- number of ambiguous bases <= 6
- the taxonomic blasting at family level matched the expected family

```
combined_data$is_valid <- with(combined_data,
  nuc_basecount >= 500 &
  stop_codons == 0 &
  ambig_basecount <= 6 &
  map2_lgl(identification, obs_taxon,
           ~any(trimws(strsplit(.y, ",")[[1]]) == .x))
)

cat("Number of valid rows:", sum(combined_data$is_valid, na.rm=TRUE), "\n")
```

```
## Number of valid rows: 3595
```

```
cat("Total rows:", nrow(combined_data), "\n")
```

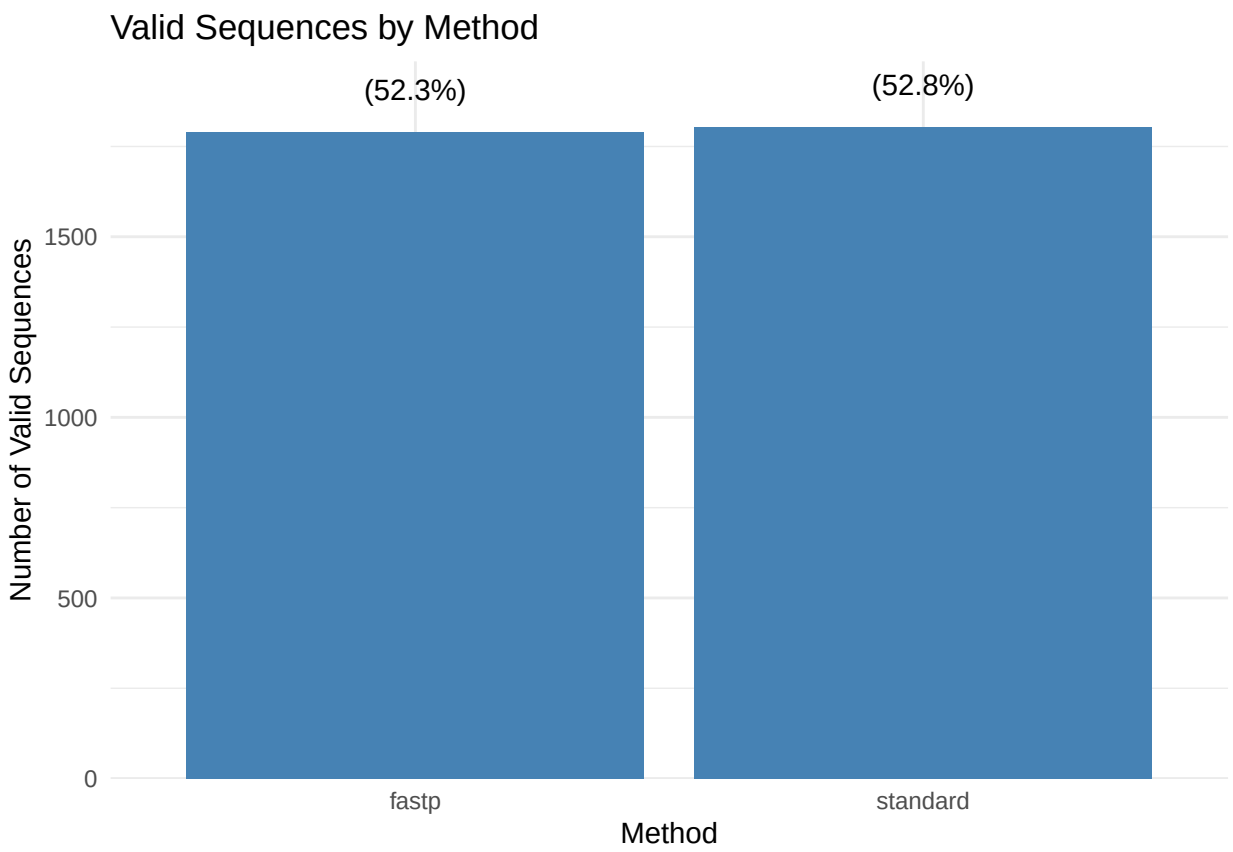
```
## Total rows: 6840
```

```
cat("Percentage valid:", round(mean(combined_data$is_valid, na.rm=TRUE) * 100, 1), "%\n")
```

```
## Percentage valid: 52.6 %
```

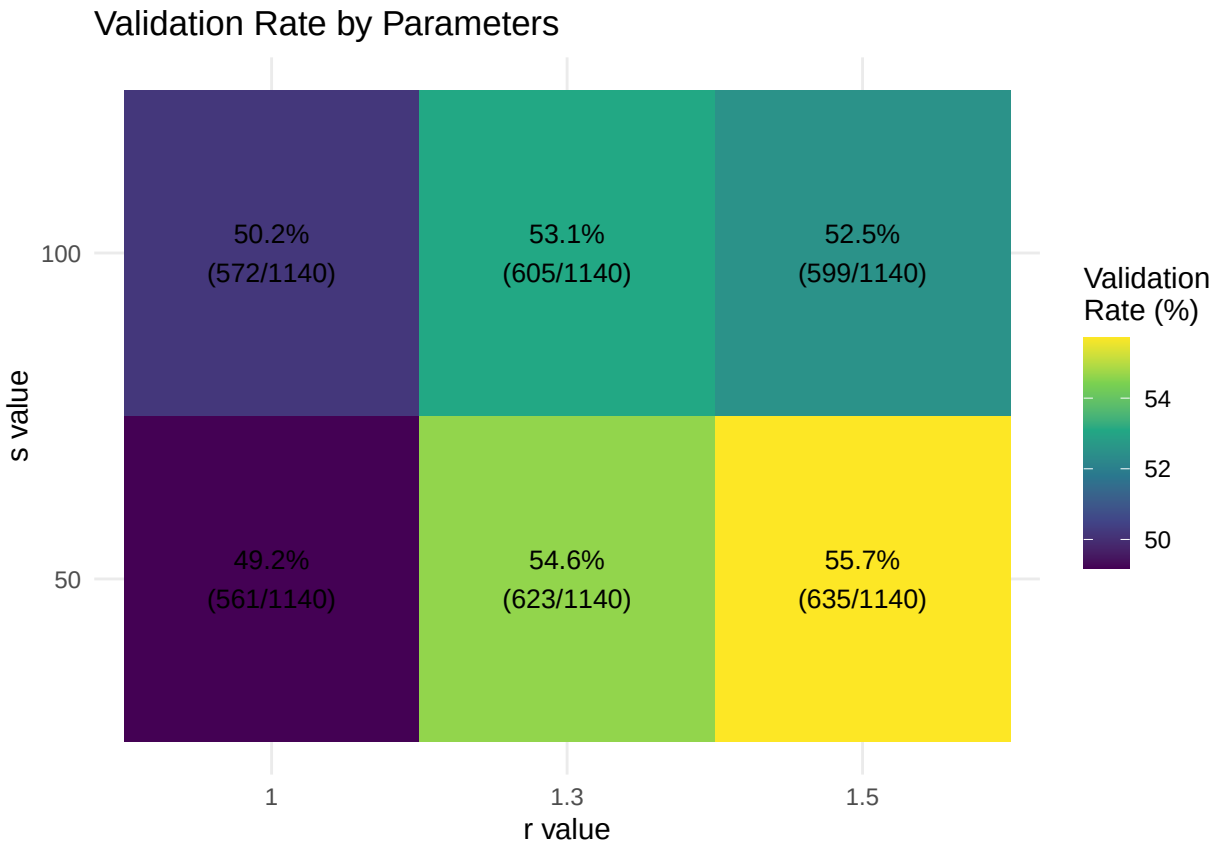
In this block, we create a bar chart to verify whether there's a difference between `fastp` and `standard`:

```
library(ggplot2)

validation_summary <- combined_data %>%
  group_by(method) %>%
  summarize(
    valid_sequences = sum(is_valid, na.rm=TRUE),
    total_sequences = n(),
    percentage_valid = round(valid_sequences/total_sequences * 100, 1)
  )

ggplot(validation_summary, aes(x=method, y=valid_sequences)) +
  geom_bar(stat="identity", fill="steelblue") +
  geom_text(aes(label=sprintf("%d\n(%.1f%%)", valid_sequences, percentage_valid)),
            vjust=-0.5) +
  labs(title="Valid Sequences by Method",
       x="Method",
       y="Number of Valid Sequences") +
  theme_minimal() +
  scale_y_continuous(expand = expansion(mult = c(0, 0.1)))  # Add space for labels
```



In the following block, we assess the overall success rate as influenced by `r` and `s`:

```
library(ggplot2)

# Create summary of validation rates for each r,s combination
parameter_summary <- combined_data %>%
  group_by(r, s) %>%
  summarize(
    total_sequences = n(),
    valid_sequences = sum(is_valid, na.rm=TRUE),
    validation_rate = round(valid_sequences/total_sequences * 100, 1),
    .groups = 'drop'
  )

# Create tile plot
ggplot(parameter_summary, aes(x=factor(r), y=factor(s), fill=validation_rate)) +
  geom_tile() +
  geom_text(aes(label=sprintf("%.1f%%\n(%d/%d)",
                              validation_rate, valid_sequences, total_sequences)),
            size=3.5) +
  scale_fill_viridis_c(name="Validation\nRate (%)", option="viridis") +
  labs(title="Validation Rate by Parameters",
       x="r value",
       y="s value") +
  theme_minimal()
```

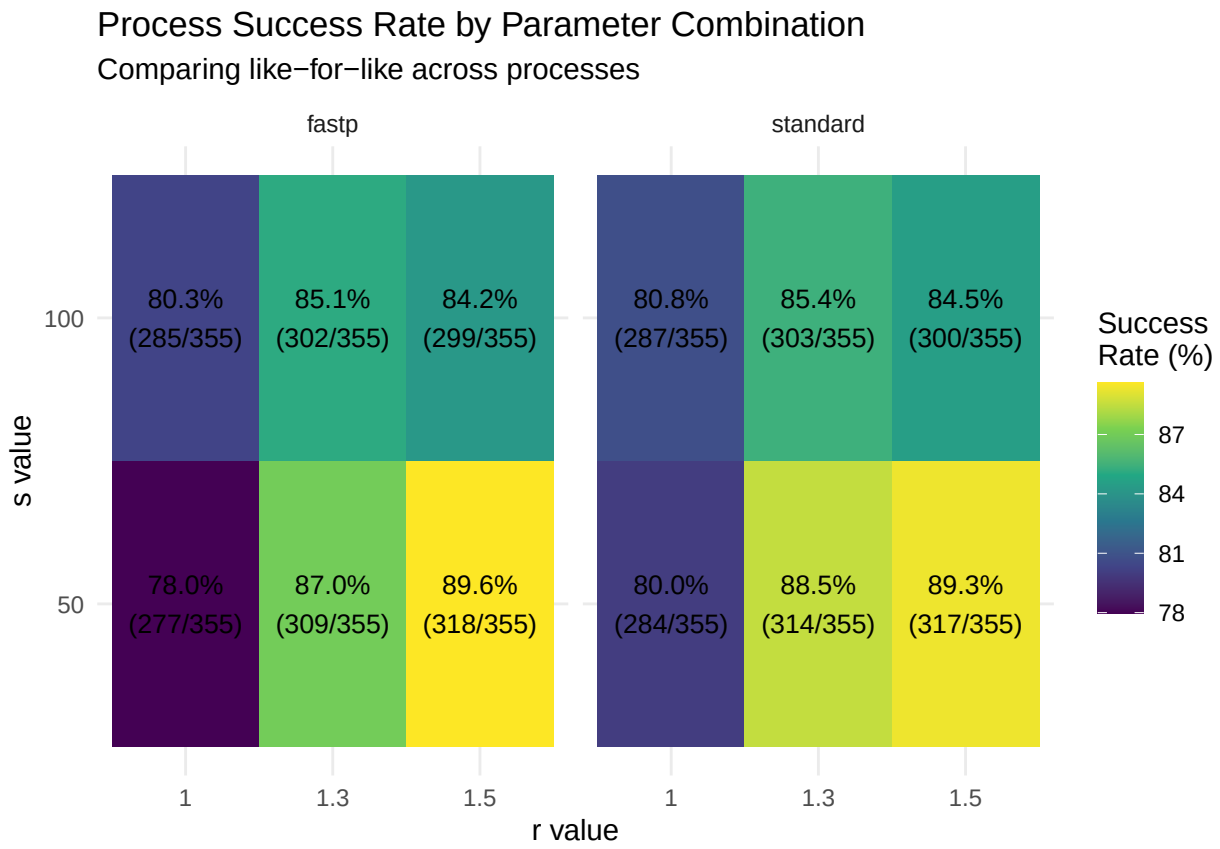


The next code block:

1. Looks at each process_id and determines if it had any valid results
2. For processes that could produce valid results, compares parameter combinations
3. Shows success rates faceted by method
4. Displays both percentages and raw counts

```
library(ggplot2)

# First, let's summarize the validation rates per process_id
process_summary <- combined_data %>%
  group_by(process_id) %>%
  # Calculate which combination gave the best result for each process
  mutate(
    total_combinations = n(),
    best_combination = any(is_valid)  # Did this process have any valid results?
  ) %>%
  filter(best_combination) %>%  # Only keep processes that had at least one valid result
  group_by(process_id, r, s, method) %>%
  summarize(
    valid = sum(is_valid, na.rm=TRUE),
    .groups = 'drop'
  ) %>%
  # Convert to success rate within each parameter combination
  group_by(r, s, method) %>%
  summarize(
    n_processes = n(),
    n_valid = sum(valid),
```

```
    success_rate = round(n_valid/n_processes * 100, 1),
    .groups = 'drop'
  )

# Create a faceted tile plot
ggplot(process_summary, aes(x=factor(r), y=factor(s), fill=success_rate)) +
  geom_tile() +
  geom_text(aes(label=sprintf("%.1f%%\n(%d/%d)",
                              success_rate, n_valid, n_processes)),
            size=3.5) +
  facet_wrap(~method) +
  scale_fill_viridis_c(name="Success\nRate (%)", option="viridis") +
  labs(title="Process Success Rate by Parameter Combination",
       subtitle="Comparing like-for-like across processes",
       x="r value",
       y="s value") +
  theme_minimal()
```



The next block gives:

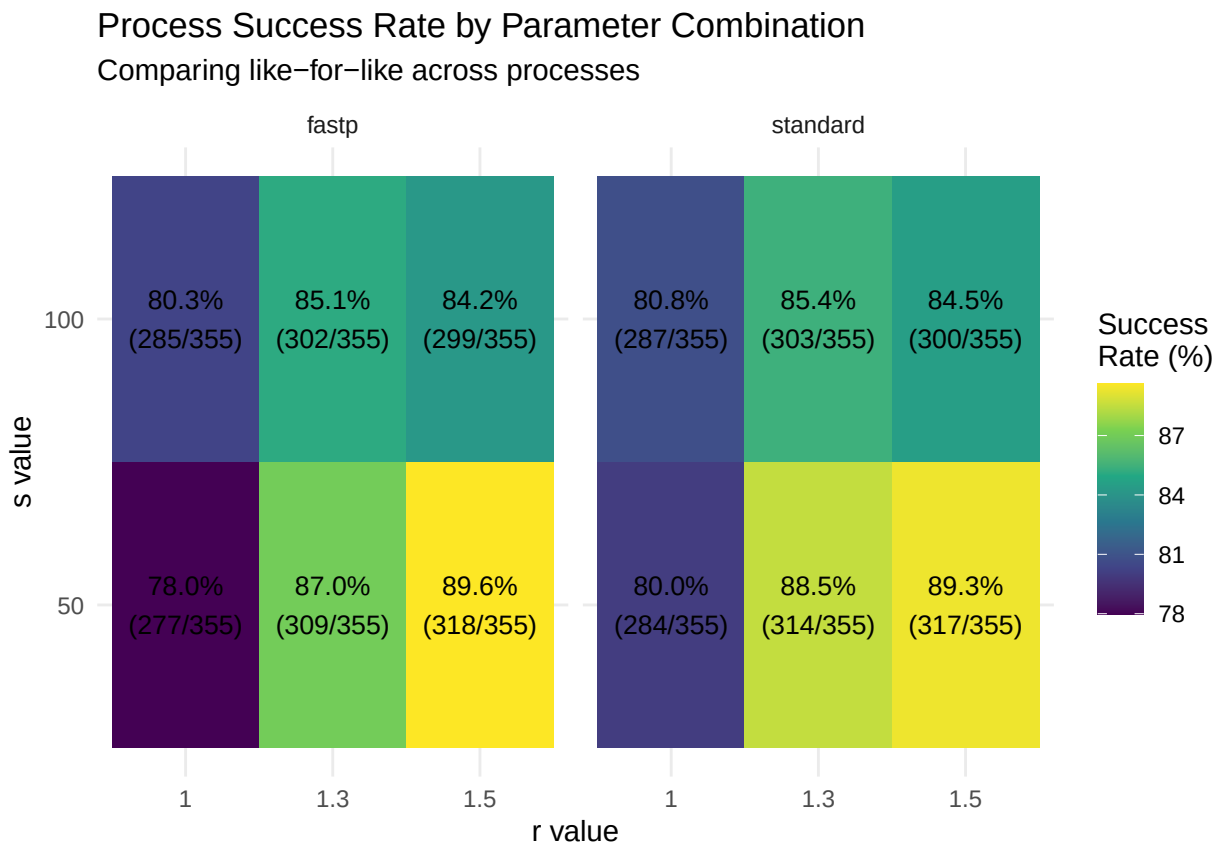
1. The original faceted tile plot showing success rates
2. A visualization of which parameter combinations most frequently give the best results
3. A stability analysis showing how changing each parameter affects outcomes
4. A parallel coordinates plot showing the "paths" through parameter space that lead to success

```
library(ggplot2)
library(tidyr)
library(GGally)
```

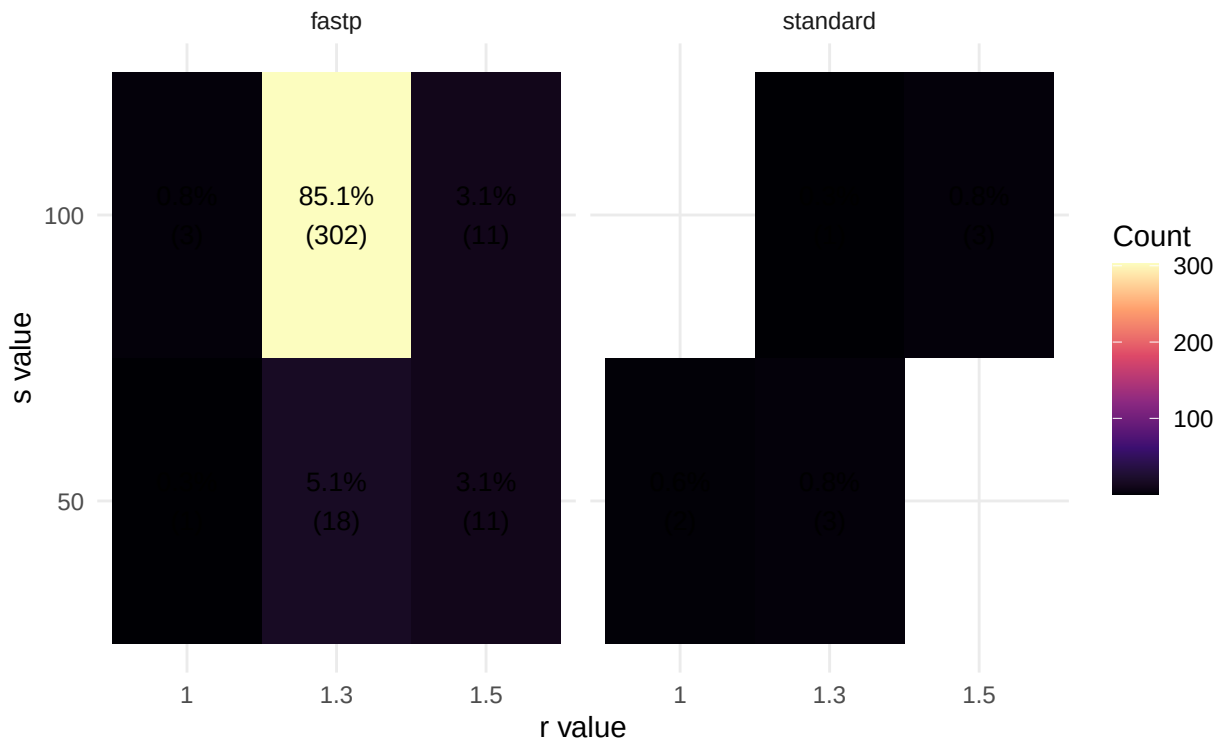
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
# 1. Base success rates by parameter combination
process_summary <- combined_data %>%
 group_by(process_id) %>%
 mutate(
   total_combinations = n(),
   best_combination = any(is_valid)
 ) %>%
 filter(best_combination) %>%
 group_by(process_id, r, s, method) %>%
 summarize(
   valid = sum(is_valid, na.rm=TRUE),
   .groups = 'drop'
 ) %>%
 group_by(r, s, method) %>%
 summarize(
   n_processes = n(),
   n_valid = sum(valid),
   success_rate = round(n_valid/n_processes * 100, 1),
   .groups = 'drop'
 )

# Plot 1: Faceted tile plot
p1 <- ggplot(process_summary, aes(x=factor(r), y=factor(s), fill=success_rate)) +
 geom_tile() +
 geom_text(aes(label=sprintf("%.1f%%\n(%d/%d)",
                             success_rate, n_valid, n_processes)),
           size=3.5) +
 facet_wrap(~method) +
 scale_fill_viridis_c(name="Success\nRate (%)", option="viridis") +
 labs(title="Process Success Rate by Parameter Combination",
      subtitle="Comparing like-for-like across processes",
      x="r value",
      y="s value") +
 theme_minimal()

# 2. Best parameter combination per process
best_params <- combined_data %>%
 group_by(process_id) %>%
 filter(any(is_valid)) %>%
 mutate(is_best = is_valid & row_number() == which.max(is_valid)) %>%
 filter(is_best) %>%
 ungroup() %>%
 count(r, s, method) %>%
 mutate(proportion = round(n/sum(n) * 100, 1))

p2 <- ggplot(best_params, aes(x=factor(r), y=factor(s), fill=n)) +
 geom_tile() +
 geom_text(aes(label=sprintf("%.1f%%\n(%d)", proportion, n)), size=3.5) +
 facet_wrap(~method) +
 scale_fill_viridis_c(name="Count", option="magma") +
```

```
 labs(title="Most Successful Parameter Combinations",
      subtitle="Number of processes where each combination gave the best result",
      x="r value",
      y="s value") +
 theme_minimal()

# 3. Enhanced stability analysis
stability_analysis <- combined_data %>%
 group_by(process_id) %>%
 filter(any(is_valid)) %>%
 summarize(
   r_effect = sum(is_valid[r == max(r)]) - sum(is_valid[r == min(r)]),
   s_effect = sum(is_valid[s == max(s)]) - sum(is_valid[s == min(s)]),
   method_effect = sum(is_valid[method == "fastp"]) - sum(is_valid[method == "standard"]),
   .groups = 'drop'
 ) %>%
 pivot_longer(cols=ends_with("effect"),
              names_to="parameter",
              values_to="effect")

p3 <- ggplot(stability_analysis, aes(x=parameter)) +
 geom_bar(aes(y=..count.., fill=factor(sign(effect))), width=0.7) +
 scale_fill_manual(values=c("red", "gray", "green"),
                   name="Direction",
                   labels=c("Negative", "No change", "Positive")) +
 geom_text(stat="count",
           aes(label=..count.., group=factor(sign(effect)), y=..count..),
           position=position_stack(vjust=0.5)) +
 labs(title="Parameter Effect Analysis",
      subtitle="Effect of increasing parameter values: r (1→1.5), s (50→100), method (standard→fastp)",
      x="Parameter",
      y="Number of Processes") +
 theme_minimal() +
 theme(axis.text.x = element_text(angle=45, hjust=1))

# 4. Parallel coordinates plot
p4 <- combined_data %>%
 filter(is_valid) %>%
 select(r, s, method) %>%
 mutate(method = as.factor(method)) %>%
 ggparcoord(
   columns = 1:3,
   scale = "std",
   alphaLines = 0.2
 ) +
 theme_minimal() +
 labs(title="Parameter Paths to Success",
      subtitle="Parallel coordinates plot of successful parameter combinations")

# Display all plots
print(p1)
```

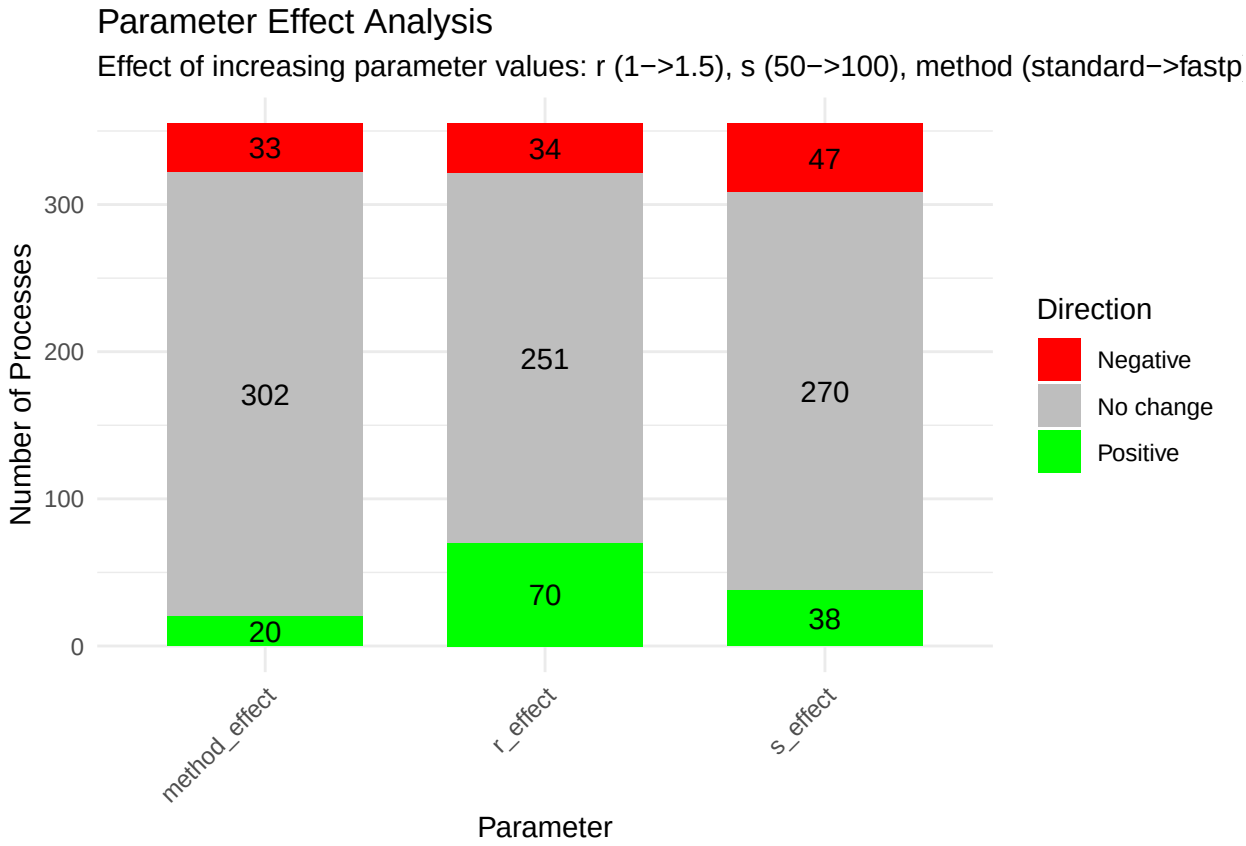

Process Success Rate by Parameter Combination
Comparing like−for−like across processes
fastp
standard
100
50
s value
80.3%
(285/355)
85.1%
(302/355)
84.2%
(299/355)
80.8%
(287/355)
85.4%
(303/355)
84.5%
(300/355)
78.0%
(277/355)
87.0%
(309/355)
89.6%
(318/355)
80.0%
(284/355)
88.5%
(314/355)
89.3%
(317/355)
1
1.3
1.5
1
1.3
1.5
r value
Success
Rate (%)
87
84
81
78

```
print(p2)
```

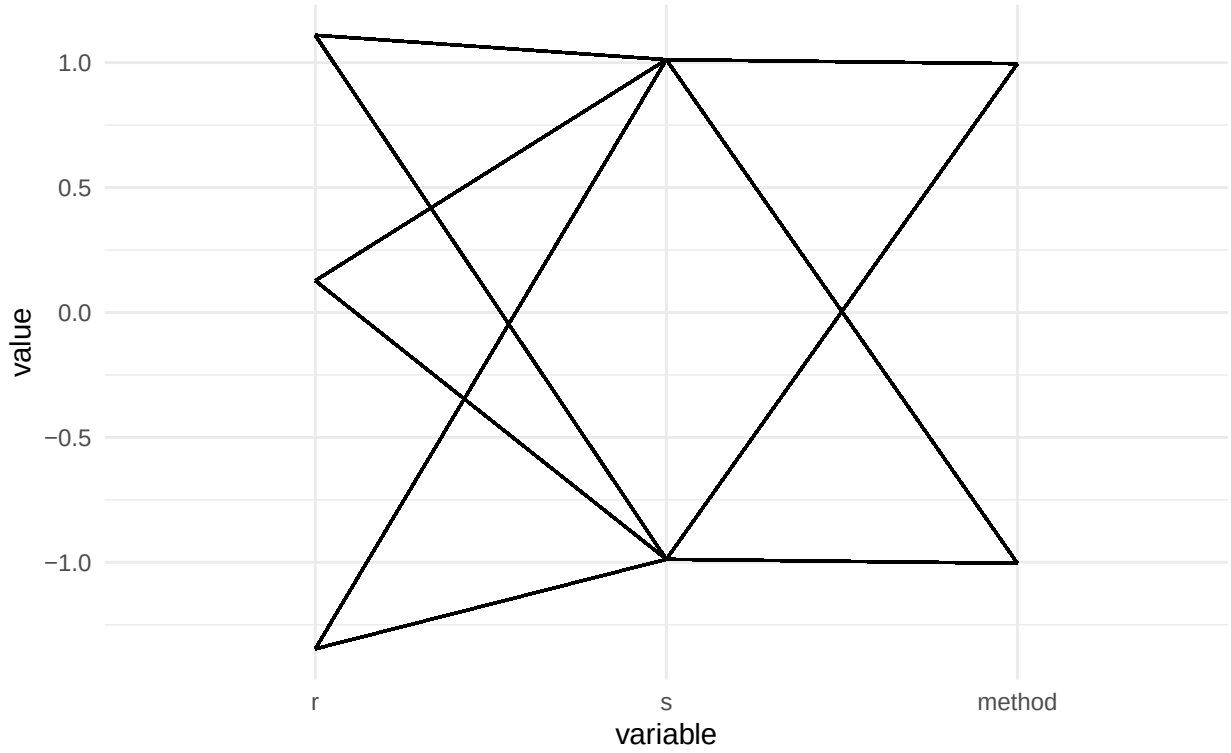



```
print(p3)
```

```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Parameter Effect Analysis
Effect of increasing parameter values: r (1−>1.5), s (50−>100), method (standard−>fastp
Number of Processes
300
200
100
0
33
302
20
34
251
70
47
270
38
method_effect
r_effect
s_effect
Parameter
Direction
Negative
No change
Positive

```
print(p4)
```


Parameter Paths to Success
Parallel coordinates plot of successful parameter combinations
1.0
0.5
0.0
−0.5
−1.0
value
r
s
method
variable