# Processing Google Takeout Fitbit Data

Genevieve Roberts

2025-08-25

## Load Packages and Setup Functions and Constants

```
library(here)
library(tidyverse)
```

```
source(here::here("fitbit/takeout_fitbit_processing_functions.R"))
source(here::here("fitbit/fitbit_plotting_functions.R"))
```

```
#current path
data_path <- here::here("fitbit/sample_fitbit_takeout_data/9Aug25_groberts_fitbit_takeout/Fitbit")
print(data_path)
```

```
## [1] "/Users/gen-omix/Documents/umass/VIGOR-surveys/fitbit/sample_fitbit_takeout_data/9Aug25_groberts_
```

## Explore some FitBit data

```
#define some constants for the nb
start_date="2025-07-07"
end_date="2025-08-09"

#old dates (2021)
old_start_date="2021-07-07"
old_end_date="2021-08-09"

#add some dates of interest to highlight
dates_of_interest_start = "2025-07-29"
dates_of_interest_end = "2025-08-01"
```

### Heart Rate Variablility

```
# Combined detailed + summary
hrv_data <- load_fitbit_hrv(start_date = start_date,
                            end_date = end_date,
                            root_dir = data_path,
                            summary_only = FALSE)

# Only summary data
hrv_data_only <- load_fitbit_hrv(start_date = start_date,
                                 end_date = end_date,
                                root_dir = data_path,
                                summary_only = TRUE)

pander(sample_n(hrv_data, 5))
```

Table 1: Table continues below

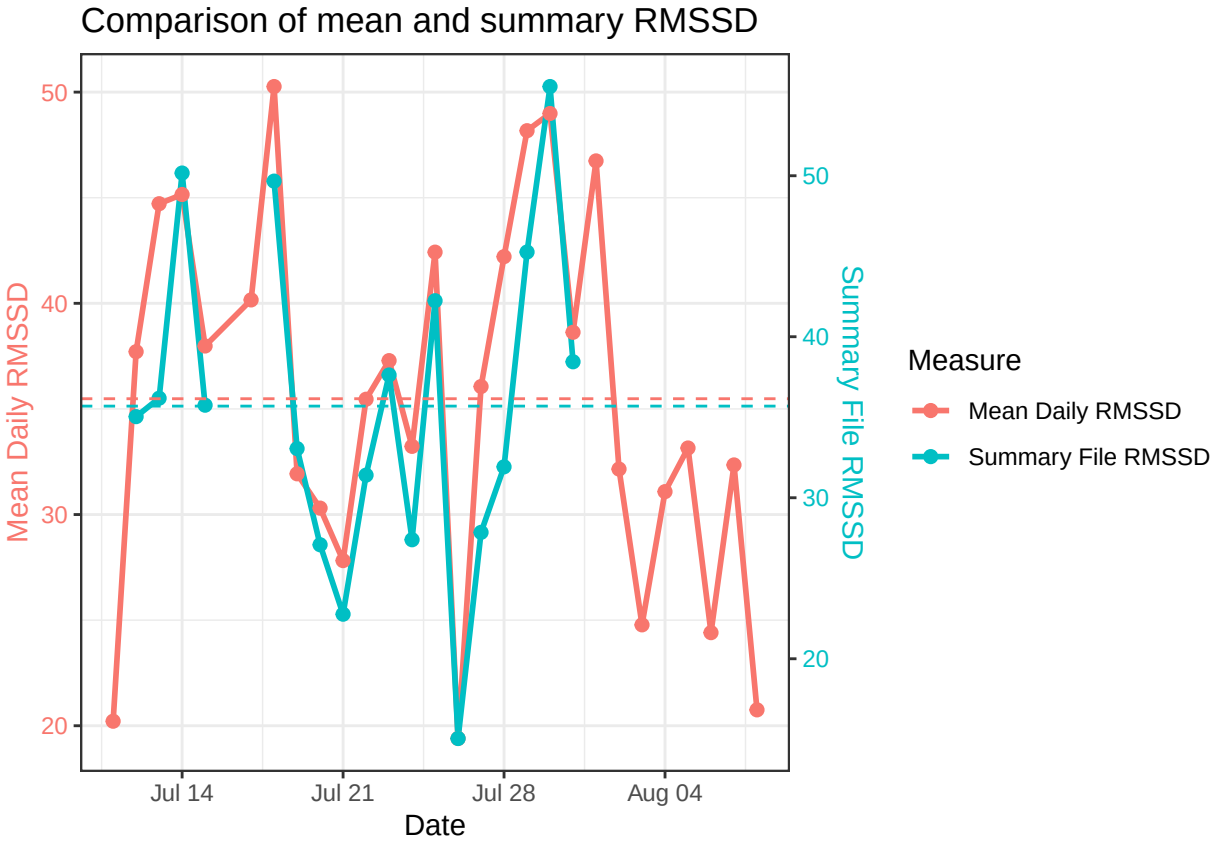
| timestamp_detail | rmssd_detail | coverage | low_frequency | high_frequency |
|---|---|---|---|---|
| 2025-07-28 01:40:00 | 41.38 | 0.99 | 897.2 | 285.4 |
| 2025-07-26 07:25:00 | 12.98 | 1.002 | 294.3 | 83.16 |
| 2025-07-27 03:45:00 | 13.52 | 0.984 | 739.2 | 25.67 |
| 2025-08-05 02:50:00 | 30.1 | 0.98 | 872.4 | 148.9 |
| 2025-07-22 04:20:00 | 65.98 | 0.936 | 3490 | 900.3 |

| file_date | timestamp_summary | rmssd_summary | nremhr | entropy |
|---|---|---|---|---|
| 2025-07-28 | 2025-07-28 | 31.92 | 66.26 | 2.7 |
| 2025-07-26 | 2025-07-26 | 15.05 | 87.78 | 2.098 |
| 2025-07-27 | 2025-07-27 | 27.85 | 67.94 | 2.593 |
| 2025-08-05 | NA | NA | NA | NA |
| 2025-07-22 | 2025-07-22 | 31.4 | 68.38 | 2.606 |

Here, I want to know if the `rmssd_summary` from the summary HRV files is simply the mean across all of the "detail" datapoints for that day. The plot below suggests they are not the same, but they are closely related.

```
check_if_mean_equals_summary <- hrv_data %>%
  group_by(file_date) %>%
  summarize(
    mean_rmssd_detail = mean(rmssd_detail, na.rm = TRUE),
    rmssd_summary = first(rmssd_summary)  # summary has one value per date
  ) %>%
  ungroup()

plot_dual_axis(
  data = check_if_mean_equals_summary,
  col1 = mean_rmssd_detail,
  col2 =  rmssd_summary,
  label1 = "Mean Daily RMSSD",
  label2 = "Summary File RMSSD",
  title = "Comparison of mean and summary RMSSD"
)
```



### Add Resting Heart Rate Data

```
rhr_data <- load_fitbit_resting_hr(start_date = start_date,
                                   end_date = end_date,
                                   root_dir = data_path)

# Combine by day (default)
combined <- combine_fitbit_data(hrv_data, rhr_data)
pander(sample_n(combined, 5))
```

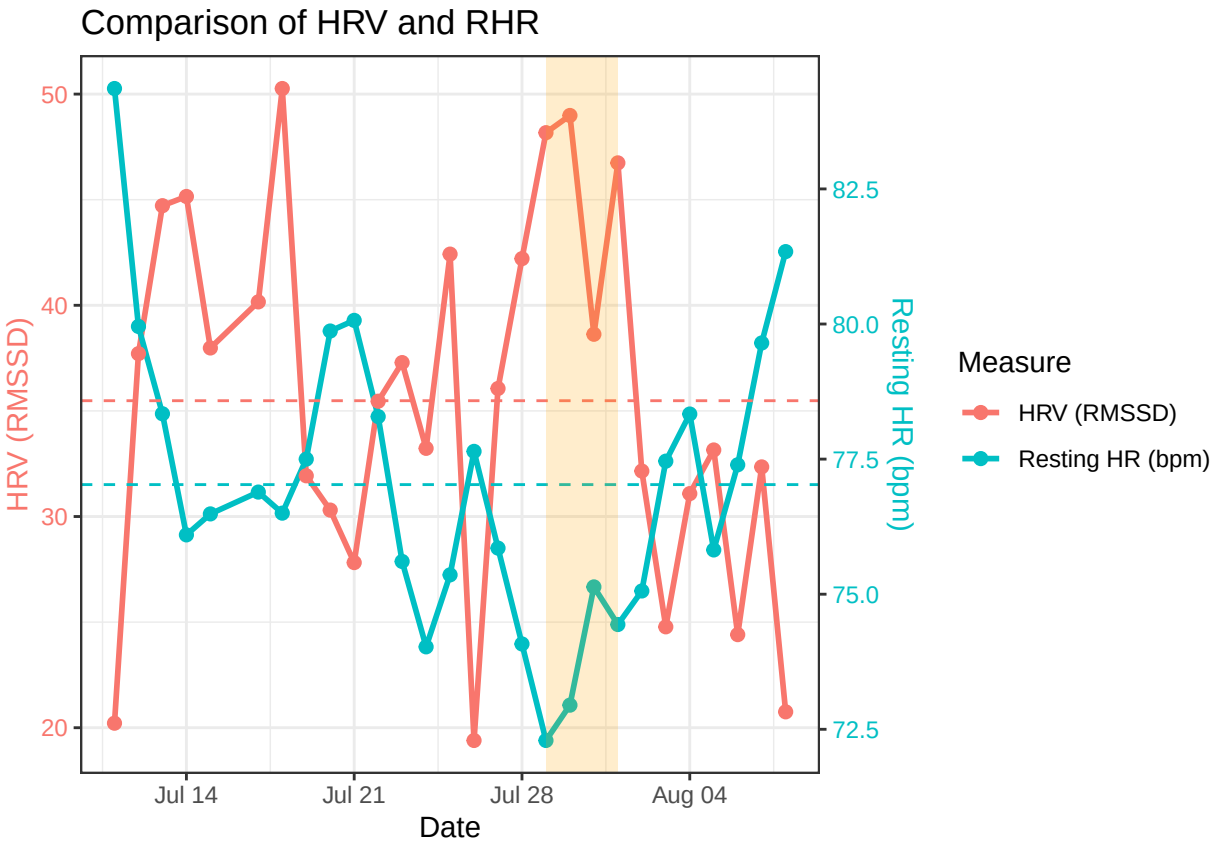
Table 3: Table continues below

| file_date | timestamp_detail | rmssd_detail | coverage | low_frequency |
|---|---|---|---|---|
| 2025-07-30 | 2025-07-30 02:45:00 | 22.83 | 1 | 1429 |
| 2025-07-12 | 2025-07-12 01:10:00 | 16.87 | 0.993 | 279.3 |
| 2025-07-15 | 2025-07-15 06:40:00 | 29.18 | 0.962 | 600.6 |
| 2025-07-15 | 2025-07-15 02:05:00 | 45.81 | 0.999 | 973.7 |
| 2025-07-21 | 2025-07-21 08:00:00 | 22.78 | 0.93 | 381.8 |

| high_frequency | timestamp_summary | rmssd_summary | nremhr | entropy | resting_hr |
|---|---|---|---|---|---|
| 63.47 | 2025-07-30 | 55.54 | 65.51 | 2.73 | 72.95 |
| 66.12 | 2025-07-12 | 35.04 | 75.33 | 2.529 | 79.95 |
| 295 | 2025-07-15 | 35.75 | 74.72 | 2.744 | 76.49 |
| 775.2 | 2025-07-15 | 35.75 | 74.72 | 2.744 | 76.49 |
| 243.4 | 2025-07-21 | 22.77 | 78.3 | 2.661 | 80.07 |

The plot below compares heart rate variability to resting heart rate. You can see there is a rough inverse correlation.

```
plot_dual_axis(
  data = combined,
  col1 = rmssd_detail,
  col2 = resting_hr,
  label1 = "HRV (RMSSD)",
  label2 = "Resting HR (bpm)",
  title = "Comparison of HRV and RHR",
  highlight_start = dates_of_interest_start,
  highlight_end = dates_of_interest_end
)
```



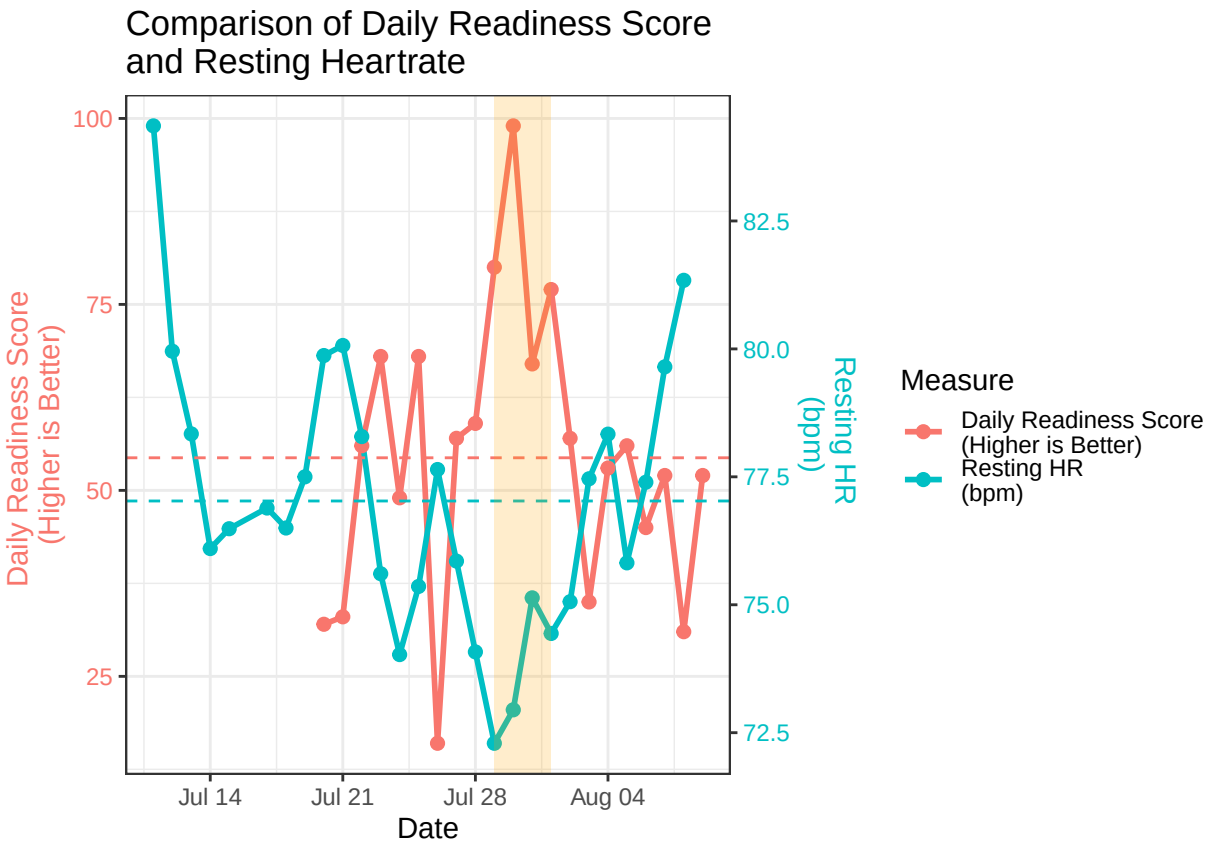
### Daily Readiness Score

```
# Load the daily readiness score and combine it with the other data
daily_ready <- load_fitbit_daily_readiness(start_date = start_date,
                                           end_date = end_date,
                                           root_dir = data_path)

# Combine by day
combined <- combine_fitbit_data(combined, daily_ready)
```

The plot below shows a roughly inverse correlation between Daily Readiness Score and Resting Heartrate:

```
plot_dual_axis(
  data = combined,
  col1 = daily_readiness_score,
  col2 = resting_hr,
  label1 = "Daily Readiness Score \n(Higher is Better)",
  label2 = "Resting HR \n(bpm)",
  title = "Comparison of Daily Readiness Score\nand Resting Heartrate",
  highlight_start = dates_of_interest_start,
  highlight_end = dates_of_interest_end
)
```



### Combine Old (2021) and Newer (2025) Data

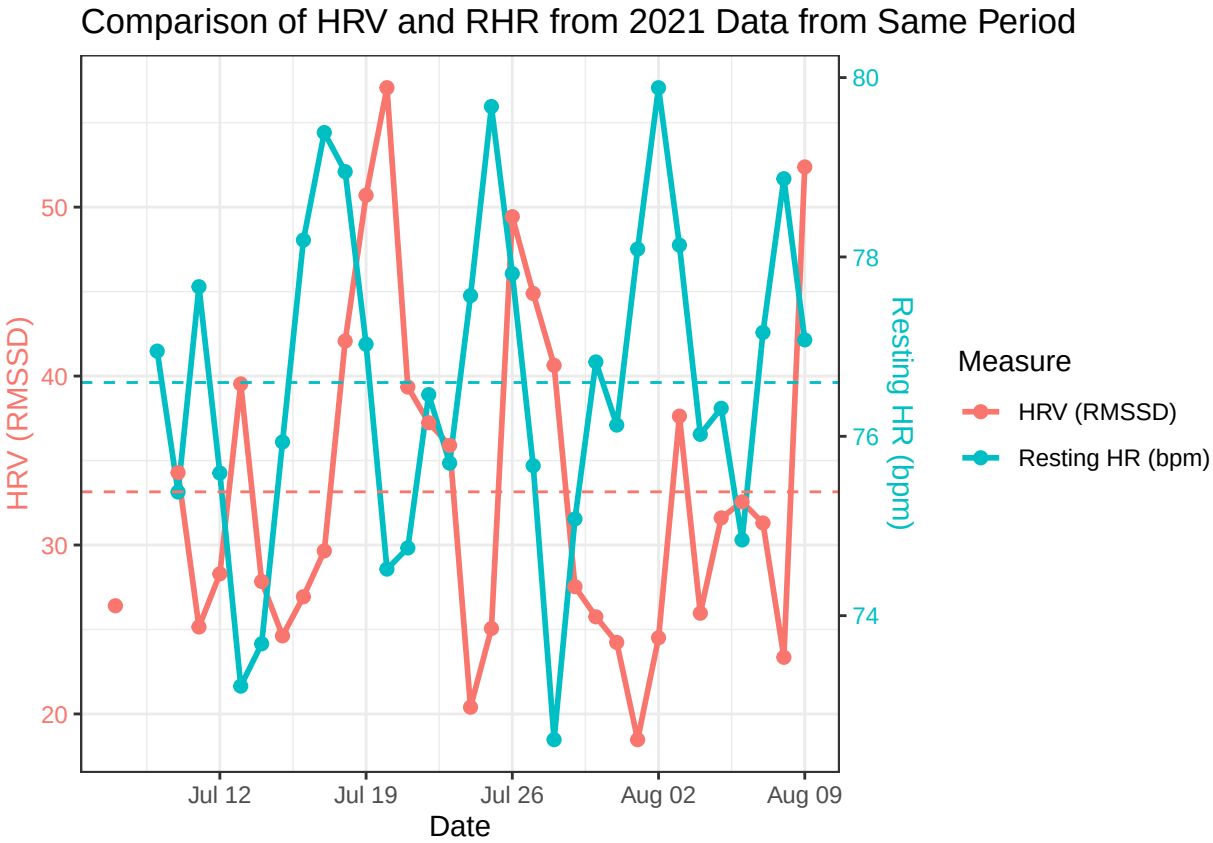
```
#read in 2021 data
old_hrv_data <- load_fitbit_hrv(start_date = old_start_date,
                                end_date = old_end_date,
                                root_dir = data_path)

old_rhr_data <- load_fitbit_resting_hr(start_date = old_start_date,
                                end_date = old_end_date,
                                root_dir = data_path)

# Combine by day (default)
old_combined <- combine_fitbit_data(old_hrv_data, old_rhr_data)

# Combine with new data
new_old_combined <- bind_rows(combined, old_combined)
```

```
plot_dual_axis(
  data = old_combined,
  col1 = rmssd_detail,
  col2 = resting_hr,
  label1 = "HRV (RMSSD)",
  label2 = "Resting HR (bpm)",
  title = "Comparison of HRV and RHR from 2021 Data from Same Period"
)
```



```r
# Create year variable from file_date
new_old_combined <- new_old_combined %>%
  mutate(year = as.factor(year(file_date)))

# Reshape data to long format for faceting
data_long <- new_old_combined %>%
  pivot_longer(cols = c(resting_hr, rmssd_detail),
               names_to = "metric",
               values_to = "value") %>%
  mutate(metric = case_when(
    metric == "resting_hr" ~ "Resting Heart Rate (bpm)",
    metric == "rmssd_detail" ~ "RMSSD Detail (ms)"
  ))

# Perform statistical tests for each metric
statistical_results <- data_long %>%
  group_by(metric) %>%
  summarise(
    # Kolmogorov-Smirnov test - compares entire distributions
    ks_p = ks.test(value[year == "2021"], value[year == "2025"])$p.value,
    ks_statistic = ks.test(value[year == "2021"], value[year == "2025"])$statistic,
    # Mann-Whitney U test (Wilcoxon rank-sum test) - compares medians
    wilcox_p = wilcox.test(value[year == "2021"], value[year == "2025"])$p.value,
    # Summary statistics for context
    median_2021 = median(value[year == "2021"], na.rm = TRUE),
    median_2025 = median(value[year == "2025"], na.rm = TRUE),
    mean_2021 = mean(value[year == "2021"], na.rm = TRUE),
    mean_2025 = mean(value[year == "2025"], na.rm = TRUE),
    # Sample sizes
    n_2021 = sum(year == "2021"),
    n_2025 = sum(year == "2025"),
    .groups = "drop"
  ) %>%
  mutate(
    # Calculate differences (2025 - 2021)
    mean_diff = mean_2025 - mean_2021,
    median_diff = median_2025 - median_2021,
    # Format p-values appropriately
    ks_p_formatted = ifelse(ks_p < 0.001, "p < 0.001",
                            paste("p =", format(round(ks_p, 3), nsmall = 3))),
    wilcox_p_formatted = ifelse(wilcox_p < 0.001, "p < 0.001",
                                paste("p =", format(round(wilcox_p, 3), nsmall = 3))),
    # Create significance labels for KS test
    ks_significance = case_when(
      ks_p < 0.001 ~ "***",
      ks_p < 0.01 ~ "**",
      ks_p < 0.05 ~ "*",
      ks_p < 0.1 ~ "†",
      TRUE ~ "ns"
    ),
    # Create significance labels for Wilcoxon test
    wilcox_significance = case_when(
      wilcox_p < 0.001 ~ "***",
```

```
      wilcox_p < 0.01 ~ "**",
      wilcox_p < 0.05 ~ "*",
      wilcox_p < 0.1 ~ "†",
      TRUE ~ "ns"
    ),
    # Create annotation text with differences and statistical tests
    annotation_text = paste0(
      "Diff Median: ", ifelse(median_diff >= 0, "+", ""), format(round(median_diff, 1), nsmall = 1),
      " | Diff Mean: ", ifelse(mean_diff >= 0, "+", ""), format(round(mean_diff, 1), nsmall = 1), "\n",
      "KS: ", ks_p_formatted, " ", ks_significance, " | ",
      "Wilcoxon: ", wilcox_p_formatted, " ", wilcox_significance
    )
  )

# Create annotation data frame for adding p-values to plot
annotation_df <- statistical_results %>%
  # Calculate y-position for annotations (top of each facet)
  left_join(
    data_long %>%
      group_by(metric) %>%
      summarise(max_val = max(value, na.rm = TRUE), .groups = "drop"),
    by = "metric"
  ) %>%
  mutate(
    y_pos = max_val * 1.05,  # Position slightly above maximum value
    x_pos = 1.5  # Center between the two years
  )

# Create the violin plot with statistical annotations
p <- ggplot(data_long, aes(x = year, y = value, fill = year)) +
  geom_violin(alpha = 0.7, trim = FALSE) +
  geom_boxplot(width = 0.1, alpha = 0.8, outlier.shape = NA) +
  # Add statistical annotation
  geom_text(data = annotation_df,
            aes(x = x_pos, y = y_pos, label = annotation_text),
            inherit.aes = FALSE,
            size = 3,
            fontface = "italic",
            hjust = 0.5) +
  facet_wrap(~metric, scales = "free_y", ncol = 2) +
  labs(
    title = "Heart Rate Variability and Resting Heart Rate: 2021 vs 2025",
    subtitle = "Violin plots showing distribution of measurements with statistical comparisons",
    x = "Year",
    y = "Value",
    fill = "Year",
    caption = "Statistical significance: *** p<0.001, ** p<0.01, * p<0.05, † p<0.1, ns = not significant
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    plot.subtitle = element_text(size = 12, hjust = 0.5),
    strip.text = element_text(size = 11, face = "bold"),
```

```
    legend.position = "bottom",
    panel.grid.minor = element_blank(),
    plot.caption = element_text(size = 9, hjust = 0)
  )

p
```