Summary Plot for Faba Bean Correction: Perimeter & Aspect Ratio & Circularity

Hao Nan Tobey Wang

2025-12-16

```
rm(list=ls())
```

```
# --- 0. Load Libraries and Setup ---
library(dplyr)
library(tidyr)
library(ggplot2)
library(boot)
library(readxl)
library(here)
library(stringr)

# --- Global Definitions ---
all_methods_for_boxplot <- c(
  "GT-MM", "GT-DM", "CP", "SVD", "min", "colorbox"
  # "CP-Affine", "SVD-Affine", "min-Affine"
)

palette <- c(
  "GT-MM" = "#6a3d9a", "GT-DM" = "#956",
  "CP" = "#1b9e77",    "SVD" = "#d95f02",
  "min" = "#7570b3",    "colorbox" = "#a6761d"
  # "CP-Affine" = "#33a02c", "SVD-Affine" = "#553300",
  # "min-Affine" = "#880000"
)

unit_labels_new <- c(
  "Perimeter" = " (mm)", "Circularity" = " (unitless)",
  "Aspect-Ratio" = " (unitless)", "Length" = " (mm)",
  "Width" = " (mm)", "Area" = " (mm\u00B2)"
)

# --- Output Directory Setup ---
current_date <- format(Sys.Date(), "%Y-%m-%d")
output_dir <- here::here(paste0("plots_output_section2_", current_date))

if (!dir.exists(output_dir)) {
  dir.create(output_dir, recursive = TRUE)
  message(paste("Created output directory:", output_dir))
} else {
  message(paste("Output directory already exists:", output_dir))
```

```
}

# --- Data Loading and Preparation (Initial) ---
df_original <- read_excel("Summary_Faba Bean Correction.xlsx", sheet = "Sheet8")

# 1. Initial Filtering (Group2 based) and Manual Exclusion
df_original <- df_original %>% mutate(Group2 = sub("-(\\d+)$","", Code))
groups_to_remove_manual <- c("460","117","198","619","283","90","615","456","280","139","620")

removed_manual_indices <- grepl(paste0("Faba-Seed-CC_Vf(", paste(groups_to_remove_manual, collapse="|")
df_original <- df_original %>% filter(!removed_manual_indices)

# 2. Convert to numeric and replace non-numerics/blanks with NA
cols_to_convert <- colnames(df_original)[grepl("-", colnames(df_original)) & !grepl("Code|Group2|^\\.\\
df_original <- df_original %>%
  mutate(across(all_of(cols_to_convert), ~ as.numeric(trimws(as.character(.)))))

regression_metrics_ci <- function(gt, pred, nboot=2000, seed=502){
  df_tmp <- data.frame(GT=gt, Pred=pred) %>% drop_na()

  if (nrow(df_tmp) < 2) {
    return(list(
      R2 = NA, Bias = NA, RMSE = NA, Slope = NA, Intercept = NA,
      Boot_Slope_CI = c(NA, NA), Boot_Intercept_CI = c(NA, NA),
      Boot_R2_CI = c(NA, NA), N = 0,
      Std_Slope_CI = c(NA, NA), Std_Intercept_CI = c(NA, NA)
    ))
  }
  diff <- df_tmp$Pred - df_tmp$GT
  bias <- mean(diff)
  rmse <- sqrt(mean(diff^2))

  fit <- lm(Pred ~ GT, data=df_tmp)
  r2 <- summary(fit)$r.squared

  # --- NEW CODE BLOCK: Calculate Standard (Analytical) CIs ---
  std_ci <- confint(fit, level = 0.95)
  std_intercept_ci <- std_ci[1, ]
  std_slope_ci <- std_ci[2, ]
  # ----------------------------------------------------------

  set.seed(seed)
  boot_fun <- function(data, indices){
    d <- data[indices,]
    fit_boot <- lm(Pred ~ GT, data=d)
    c(coef(fit_boot), summary(fit_boot)$r.squared)
  }

  boot_res <- tryCatch(boot(df_tmp, boot_fun, R=nboot), error=function(e) NULL)

  boot_ci_intercept <- tryCatch(boot.ci(boot_res, type="perc", index=1)$percent[4:5], error=function(e)
  boot_ci_slope <- tryCatch(boot.ci(boot_res, type="perc", index=2)$percent[4:5], error=function(e) c(N
  boot_ci_r2 <- tryCatch(boot.ci(boot_res, type="perc", index=3)$percent[4:5], error=function(e) c(NA, N
```

```
  list(
    R2 = r2, Bias = bias, RMSE = rmse, Slope = coef(fit)[2], Intercept = coef(fit)[1],
    Boot_Slope_CI = boot_ci_slope, Boot_Intercept_CI = boot_ci_intercept,
    Boot_R2_CI = boot_ci_r2, N = nrow(df_tmp),
    Std_Slope_CI = std_slope_ci, # <--- NEW
    Std_Intercept_CI = std_intercept_ci # <--- NEW
  )
}

# --- 1. Boxplot Function Definition (MODIFIED for clearer outlier labels) ---
create_boxplot_new <- function(df, col_names, measure, method_levels = all_methods_for_boxplot, color_pa

  y_label <- paste0(measure, unit_labels[measure])
  plot_title <- paste0(measure, " Comparison Across Methods")

  df_long <- df %>%
    select(Code, all_of(col_names)) %>%
    pivot_longer(-Code, names_to="Method_Col", values_to="Value") %>%
    mutate(Method = sub(paste0(measure, "-"), "", Method_Col)) %>%
    drop_na(Value)

  df_long <- df_long %>%
    filter(Method %in% all_methods_for_boxplot)

  if (nrow(df_long) == 0) {
    message(paste("No data available for boxplot:", plot_title))
    return(NULL)
  }
  present_methods <- unique(df_long$Method)
  df_long$Method <- factor(df_long$Method, levels = intersect(method_levels, present_methods))

  # --- STATS CALCULATION (including lower whisker for outlier identification) ---
  stats <- df_long %>%
    group_by(Method) %>%
    summarise(
      boxplot_stats = list(boxplot.stats(Value)$stats),
      median_val = median(Value, na.rm = TRUE),
      min_val = min(Value, na.rm = TRUE),
      max_val = max(Value, na.rm = TRUE),
      n = sum(!is.na(Value)),
      .groups = 'drop'
    ) %>%
    mutate(
      lower_whisker = sapply(boxplot_stats, '[', 1),
      upper_whisker = sapply(boxplot_stats, '[', 5)
    ) %>%
    select(-boxplot_stats) %>%
    filter(n > 0)

  if (nrow(stats) == 0) return(NULL)

  # --- IDENTIFY LOWER OUTLIERS ---
  outlier_fences <- df_long %>%
```

```
  group_by(Method) %>%
  summarise(
    Q1 = quantile(Value, 0.25, na.rm = TRUE),
    Q3 = quantile(Value, 0.75, na.rm = TRUE),
    IQR = Q3 - Q1,
    lower_fence = Q1 - 1.5 * IQR,
    .groups = 'drop'
  )

df_long_with_fences <- df_long %>%
  left_join(outlier_fences %>% select(Method, lower_fence), by = "Method")
lower_outliers <- df_long_with_fences %>%
  filter(Value < lower_fence)

min_y_global <- min(df_long$Value, na.rm = TRUE)
max_y_global <- max(stats$upper_whisker, na.rm = TRUE)
y_range <- max_y_global - min_y_global

# Position for N/Median/Min/Max text
stats <- stats %>%
  mutate(text_y_pos = upper_whisker + 0.02 * y_range)

p <- ggplot(df_long, aes(x = Method, y = Value, fill = Method)) +
  geom_boxplot(outlier.shape = NA, alpha = 0.7) + # Remove default outliers

  # --- ADD LOWER OUTLIER POINTS AND LABELS ---
  geom_point(data = lower_outliers,
             aes(y = Value, color = Method),
             shape = 21, size = 3, stroke = 1.5) +
  # MODIFICATION 1: Add position_jitter to spread the labels horizontally
  geom_text(data = lower_outliers,
            aes(y = Value, label = Code),
            vjust = 1.5, size = 3, color = "black",
            position = position_jitter(width = 0.15, height = 0.038, seed = 1)) +

  scale_fill_manual(values = color_palette, breaks = stats$Method) +
  scale_color_manual(values = color_palette, breaks = stats$Method) +
  theme_minimal(base_size = 14) +
  labs(title = plot_title, x = "Measurement Method", y = y_label) +
  theme(
    legend.position = "none",
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) +
  geom_text(
    data = stats,
    aes(x = Method, y = text_y_pos,
        label = paste0(
          "N: ", n, "\n",
          "Median: ", round(median_val, 3), "\n",
          "Min: ", round(min_val, 3), "\n",
          "Max: ", round(max_val, 3)
      )
    ),
```

```r
      vjust = 0, size = 3,
      color = "black"
    ) +
    coord_cartesian(ylim = c(min_y_global, max(stats$text_y_pos, na.rm=TRUE) * 1.05))

  return(p)
}

# --- 2. Scatterplot Function Definition (MODIFIED to display both CIs) ---
create_scatterplot_new <- function(df, x_col, y_col, measure, color_palette = palette, unit_labels = uni

    tmp <- df %>%
        select(Code, all_of(x_col), all_of(y_col)) %>%
        rename(GT = all_of(x_col), Pred = all_of(y_col)) %>%
        drop_na() # Ensures both X and Y values are present for regression

    if (nrow(tmp) < 2) {
        message(paste("Not enough data to run regression for:", x_col, "vs", y_col))
        return(NULL)
    }
    measure <- trimws(measure)
    unit <- unit_labels[measure]

    if (is.na(unit)) {
        unit <- ""
        warning(paste("Unit lookup failed for measure:", measure, ". Using empty string for axis labels,
    }

    gt_method <- sub(paste0("^", measure, "-"), "", x_col)
    pred_method <- sub(paste0("^", measure, "-"), "", y_col)

    gt_method <- trimws(gt_method)
    pred_method <- trimws(pred_method)
    gt_method <- if (str_detect(gt_method, "GT-MM")) "GT-MM" else gt_method

    metrics <- regression_metrics_ci(tmp$GT, tmp$Pred)

    # --- MODIFIED: Include both Standard (Std) and Bootstrap (Boot) CIs ---
    annotation_text <- paste0(
        "N=", metrics$N, "\n",
        "R²=", round(metrics$R2, 3), " (Boot 95% CI: [", round(metrics$Boot_R2_CI[1], 3), ", ", round(m
        "Bias=", round(metrics$Bias, 3), unit, "\n",
        "RMSE=", round(metrics$RMSE, 3), unit, "\n",
        "Slope=", round(metrics$Slope, 3), "\n",
        " 95% CI: [", round(metrics$Std_Slope_CI[1], 3), ", ", round(metrics$Std_Slope_CI[2], 3), "]\n"
        " Boot 95% CI: [", round(metrics$Boot_Slope_CI[1], 3), ", ", round(metrics$Boot_Slope_CI[2], 3)
        "Intercept=", round(metrics$Intercept, 3), unit, "\n",
        " 95% CI: [", round(metrics$Std_Intercept_CI[1], 3), ", ", round(metrics$Std_Intercept_CI[2], 3
        " Boot 95% CI: [", round(metrics$Boot_Intercept_CI[1], 3), ", ", round(metrics$Boot_Intercept_C
    )
    # ------------------------------------------------------------------------

    max_val <- max(tmp$GT, tmp$Pred, na.rm=TRUE)
```

```
    min_val <- min(tmp$GT, tmp$Pred, na.rm=TRUE)
    plot_range <- max_val - min_val

    x_pos_text <- min_val + plot_range * 0.05
    y_pos_text <- max_val - plot_range * 0.05

    point_color <- if (pred_method %in% names(color_palette)) color_palette[pred_method] else "black"

    p_scatter <- ggplot(tmp, aes(x=GT, y=Pred)) +
        geom_point(color=point_color, alpha=0.8, size=1.5) +
        geom_smooth(method="lm", se=TRUE, color="navy", fill=alpha("navy",0.2)) +
        geom_abline(slope=1, intercept=0, linetype="dashed") +
        annotate("text", x=min(tmp$GT, na.rm = TRUE) - 0.15, y=y_pos_text,
                 label=annotation_text, hjust=0, vjust=1, size=5, fontface="bold") +
        labs(title=paste0(measure, " - ", pred_method, " vs ", gt_method),
             x=paste0(gt_method, unit),
             y=paste0(pred_method," Measurement", unit)) +
        theme_bw(base_size = 12) +
        coord_fixed(ratio=1)
    p_scatter <- p_scatter +
        expand_limits(x = min(tmp$GT, na.rm = TRUE), y = min(tmp$GT, na.rm = TRUE))
    return(p_scatter)
}

# - NEW 2b. Altman-Bland Plot Function Definition -
create_altman_plot <- function(df, x_col, y_col, measure, color_palette = palette, unit_labels = unit_la

  tmp <- df %>%
    select(Code, all_of(x_col), all_of(y_col)) %>%
    rename(Method1 = all_of(x_col), Method2 = all_of(y_col)) %>%
    drop_na()

  if (nrow(tmp) < 2) {
    message(paste("Not enough data to run Altman-Bland plot for:", x_col, "vs", y_col))
    return(NULL)
  }

  # Calculate Difference and Average
  tmp <- tmp %>%
    mutate(
      Difference = Method2 - Method1,
      Average = (Method1 + Method2) / 2
    )

  measure <- trimws(measure)
  unit <- unit_labels[measure]

  gt_method <- sub(paste0("^", measure, "-"), "", x_col)
  pred_method <- sub(paste0("^", measure, "-"), "", y_col)
  gt_method <- trimws(gt_method)
  pred_method <- trimws(pred_method)
  gt_method <- if (str_detect(gt_method, "GT-MM")) "GT-MM" else gt_method
```

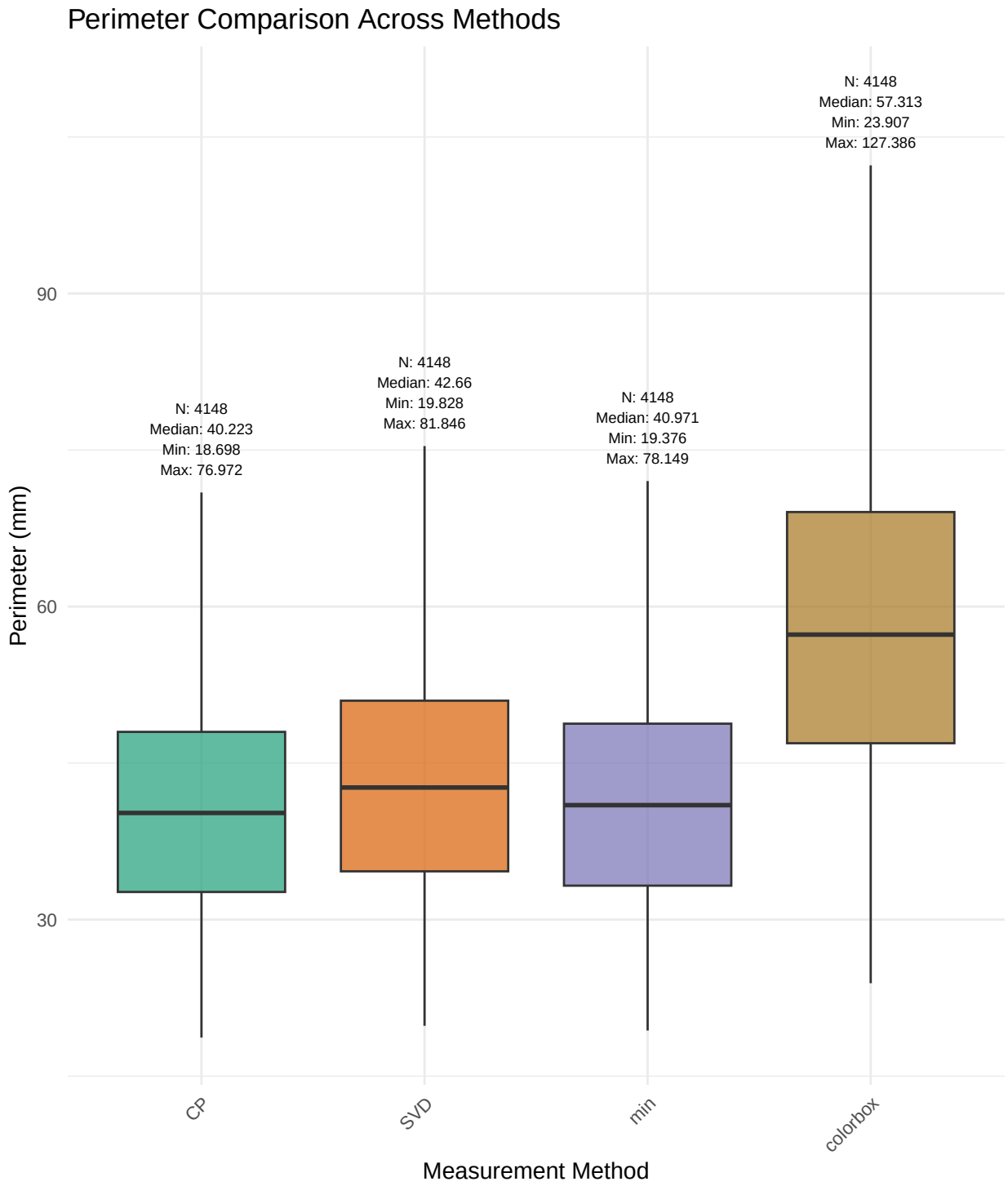
```r
# Calculate statistics for Altman-Bland
mean_diff <- mean(tmp$Difference, na.rm = TRUE)
sd_diff <- sd(tmp$Difference, na.rm = TRUE)
loa_upper <- mean_diff + 1.96 * sd_diff
loa_lower <- mean_diff - 1.96 * sd_diff

# Title and Labels
plot_title <- paste0("Altman-Bland Plot for ", measure, "\n(", pred_method, " vs ", gt_method, ")")
x_label <- paste0("Average of ", gt_method, " and ", pred_method, " Measurement", unit)
y_label <- paste0("Difference (", pred_method, " - ", gt_method, ")", unit)

# Annotation for Mean Difference and LoA
annotation_text <- paste0(
  "Mean Diff: ", round(mean_diff, 3), "\n",
  "Upper LoA (+1.96 SD): ", round(loa_upper, 3), "\n",
  "Lower LoA (-1.96 SD): ", round(loa_lower, 3)
)

# Determine point color
point_color <- if (pred_method %in% names(color_palette)) color_palette[pred_method] else "black"

# Determine annotation position (bottom-left)
avg_min <- min(tmp$Average, na.rm = TRUE)
avg_max <- max(tmp$Average, na.rm = TRUE)
avg_range <- avg_max - avg_min

diff_min <- min(c(tmp$Difference, loa_lower), na.rm = TRUE)
diff_max <- max(c(tmp$Difference, loa_upper), na.rm = TRUE)
diff_range <- diff_max - diff_min

x_pos_text <- avg_min + 0.02 * avg_range   # slightly right from left edge
y_pos_text <- diff_min + 0.02 * diff_range # slightly above bottom edge

p_altman <- ggplot(tmp, aes(x = Average, y = Difference)) +
  geom_point(color = point_color, alpha = 0.8, size = 1.5) +

  # Mean Difference line
  geom_hline(yintercept = mean_diff, linetype = "solid", color = "red", size = 1) +

  # Limits of Agreement (LoA) lines
  geom_hline(yintercept = loa_upper, linetype = "dashed", color = "blue", size = 1) +
  geom_hline(yintercept = loa_lower, linetype = "dashed", color = "blue", size = 1) +

  # Annotation at bottom-left
  annotate("text", x = x_pos_text, y = y_pos_text,
           label = annotation_text, hjust = 0, vjust = 0, size = 6.2, fontface = "bold") +

  labs(title = plot_title, x = x_label, y = y_label) +
  theme_bw(base_size = 12) +
  # Ensure all data and text is visible
  coord_cartesian(ylim = c(diff_min - 0.1 * diff_range, diff_max + 0.1 * diff_range))

return(p_altman)
```

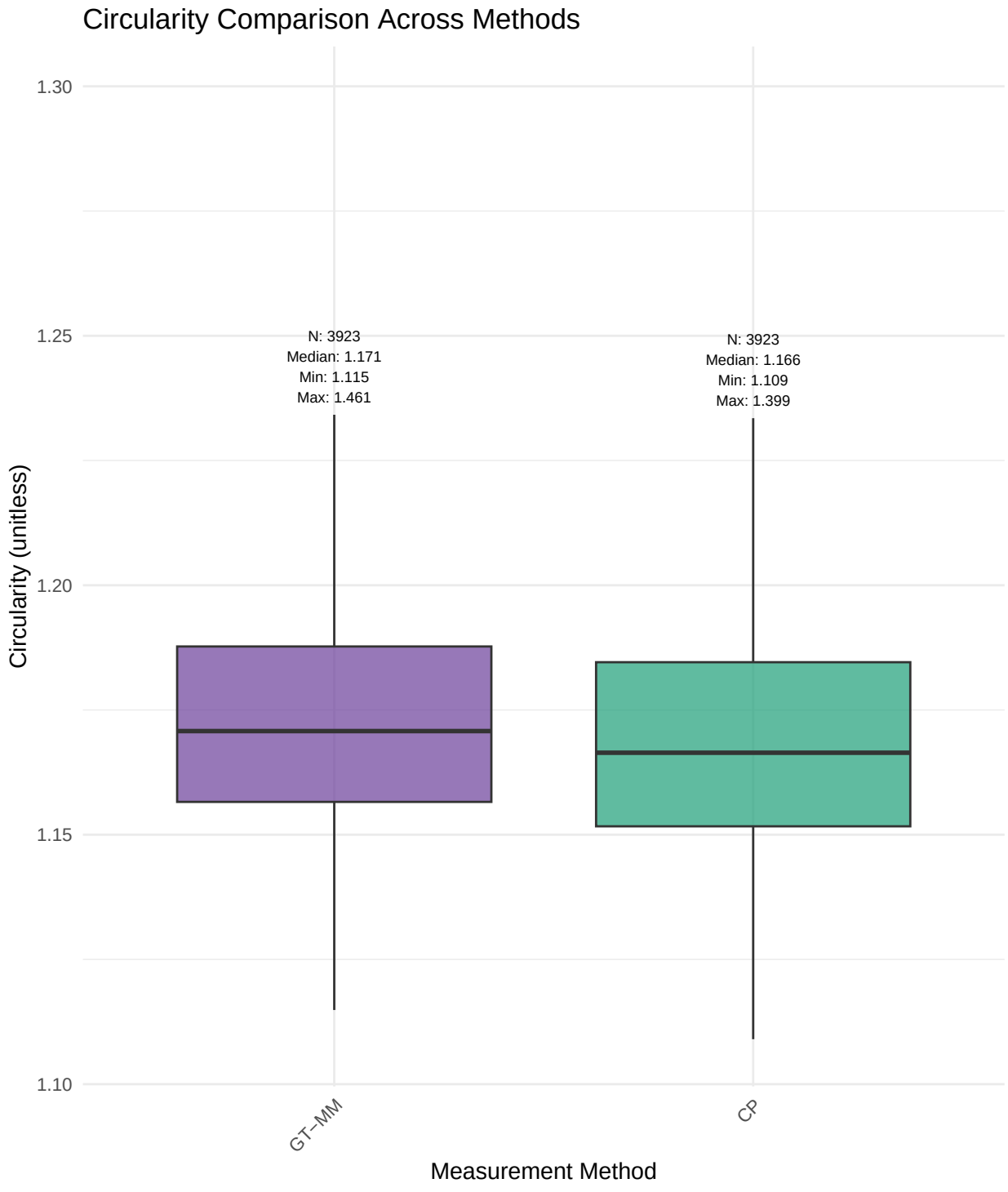
```
}

# --- Helper Function: Within-Group Sorting (No Change) ---
sort_within_group <- function(df, cols) {
  existing_cols <- intersect(cols, names(df))
  df %>%
    group_by(Group2) %>%
    mutate(across(all_of(existing_cols), ~ sort(.x, na.last = TRUE))) %>%
    ungroup()
}

# --- 3. Master Function to Generate and Save All Plots (MODIFIED) ---
generate_analysis_plots <- function(df_original, measure_groups_for_boxplot, scatterplot_pairs, output_

  # --- Define the measures that require strict group-wise filtering/sorting ---
  # Only Circularity is strictly filtered, as Aspect-Ratio seems too aggressive.
  strict_measures <- c("Circularity")

  # ------------------------------------------------------------------------
  # PHASE 1: Perimeter Boxplot (Method-Oriented Filtering - Max Retention)
  # ------------------------------------------------------------------------
  message("\n--- PHASE 1: Generating Perimeter Boxplot (Method-Oriented Filtering) ---")

  perimeter_measure <- "Perimeter"
  if (perimeter_measure %in% measure_groups_for_boxplot) {
    cols_to_plot <- grep(paste0("^", perimeter_measure, "-"), colnames(df_original), value = TRUE)

    if (length(cols_to_plot) > 0) {
      p_box <- create_boxplot_new(df_original, cols_to_plot, perimeter_measure)

      if (!is.null(p_box)) {
        filename <- file.path(output_dir, paste0("Boxplot_", perimeter_measure, "_PHASE1.png"))
        tryCatch({
          invisible(print(p_box))
          ggsave(filename, p_box, width=8, height=6, units="in", dpi=300)
          message(paste("  [SAVED BOXPLOT] ->", basename(filename)))
        }, error = function(e) {
          warning(paste("  [ERROR] Could not save Boxplot for", perimeter_measure, ". Error:", e$messag
        })
      }
    } else {
      warning(paste("No columns found for measure group:", perimeter_measure, ". Skipping."))
    }
  }

  # ------------------------------------------------------------------------
  # PHASE 2: Strict Group-Wise Filtering (Applied ONLY to Circularity GT)
  # ------------------------------------------------------------------------
  message("\n--- PHASE 2: Applying Strict Group-Wise Filter (Circularity GT Only) ---")

  # FIX: Only filter groups based on Circularity GT being NA or 0
  cols_for_group_filter_GT_Circular <- c("Circularity-GT-MM")
```

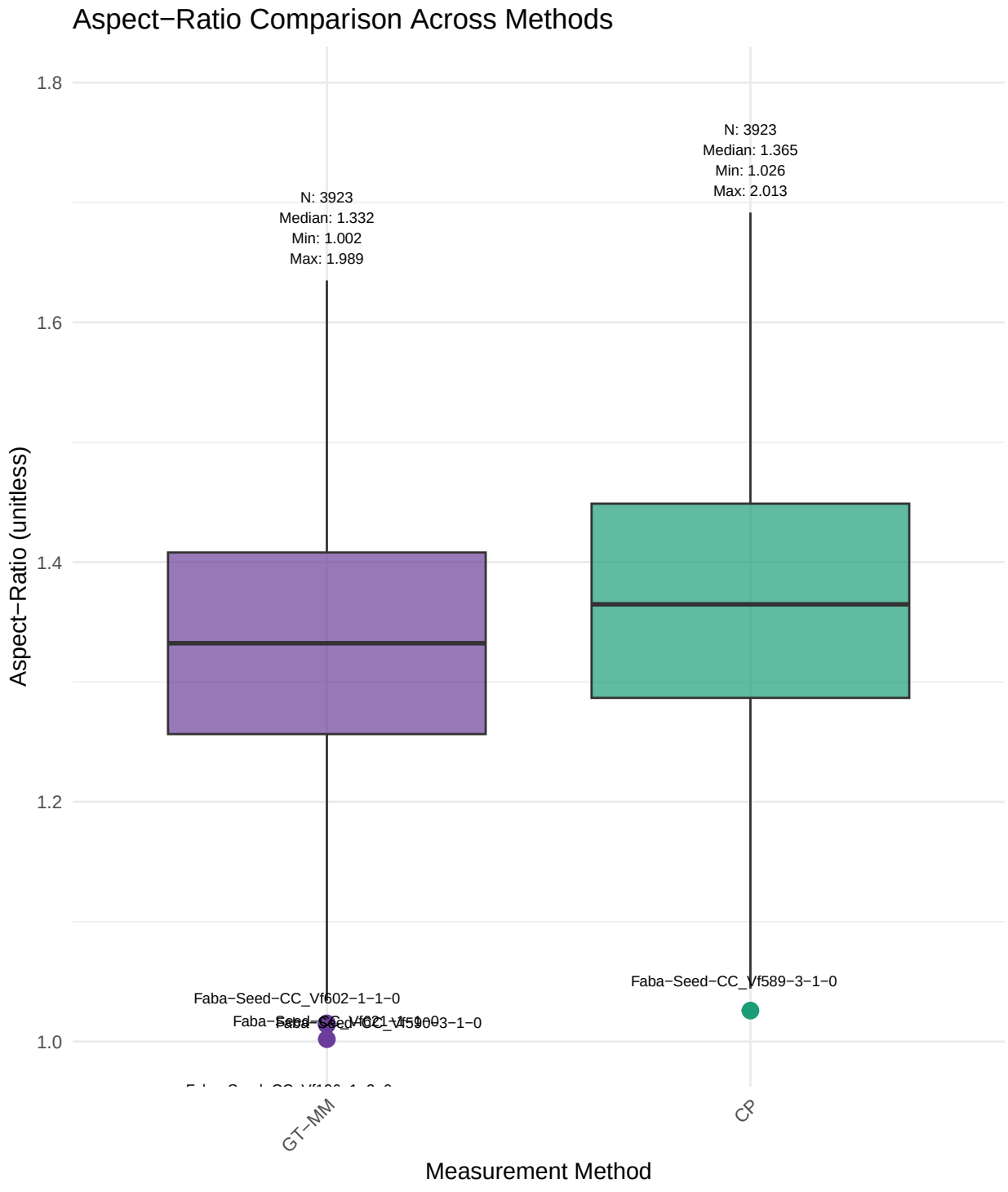
```r
df_filtered <- df_original # Initialize the filtered data frame

if (length(cols_for_group_filter_GT_Circular) > 0) {

  # 1. Identify groups to remove based on NA/0 in Circularity GT column
  df_removed_groups <- df_filtered %>%
    select(Group2, all_of(cols_for_group_filter_GT_Circular)) %>%
    # Convert 0 to NA for filtering purposes
    mutate(across(all_of(cols_for_group_filter_GT_Circular), ~ ifelse(.x == 0, NA, .x))) %>%
    group_by(Group2) %>%
    # Check if ANY row in the current group has an NA in the required GT column
    summarise(
      remove_group = any(rowSums(is.na(pick(all_of(cols_for_group_filter_GT_Circular)))) > 0),
      .groups = 'drop'
    ) %>%
    filter(remove_group)

  # 2. Apply the filter to create the strictly filtered data frame
  groups_to_remove_na_zero <- df_removed_groups$Group2
  df_filtered <- df_original %>% filter(!Group2 %in% groups_to_remove_na_zero)

  message(paste("  Removed", length(groups_to_remove_na_zero), "groups due to NA or 0 in Circularity-G

  if (length(groups_to_remove_na_zero) > 0) {
      removed_codes <- df_original %>%
          filter(Group2 %in% groups_to_remove_na_zero) %>%
          select(Code, Group2) %>%
          mutate(Reason = "NA or 0 in GT Circularity (Group-Wise)")
      write.csv(removed_codes, file.path(output_dir, "Removed_Groups_NA_Zero_Groupwise_Circ_GT_Only.cs
      message(paste("  Saved Group-Wise Exclusions to:", file.path(output_dir, "Removed_Groups_NA_Zero
  }

} else {
  message("No Circularity GT columns found for group filtering. Proceeding with all data for Phase 3.")
}

# --- 2a. WITHIN-GROUP SORTING (Applied ONLY to filtered data) ---
# Sorting is done for BOTH Circularity AND Aspect-Ratio on the slightly filtered data.
cols_circularity <- grep("^Circularity-", colnames(df_original), value = TRUE)
cols_aspect_ratio <- grep("^Aspect-Ratio-", colnames(df_original), value = TRUE)

df_filtered <- df_filtered %>%
  sort_within_group(cols_aspect_ratio) %>%
  sort_within_group(cols_circularity)

message("  Applied within-group sorting for Aspect-Ratio and Circularity to filtered data.")

# ------------------------------------------------------------------------
# PHASE 3: Generate Remaining Plots (Circularity and Aspect-Ratio)
# ------------------------------------------------------------------------
message("\n--- PHASE 3: Starting Circularity and Aspect-Ratio Plot Generation ---")

measures_to_plot_phase3 <- c("Circularity", "Aspect-Ratio") # All remaining shape metrics
```

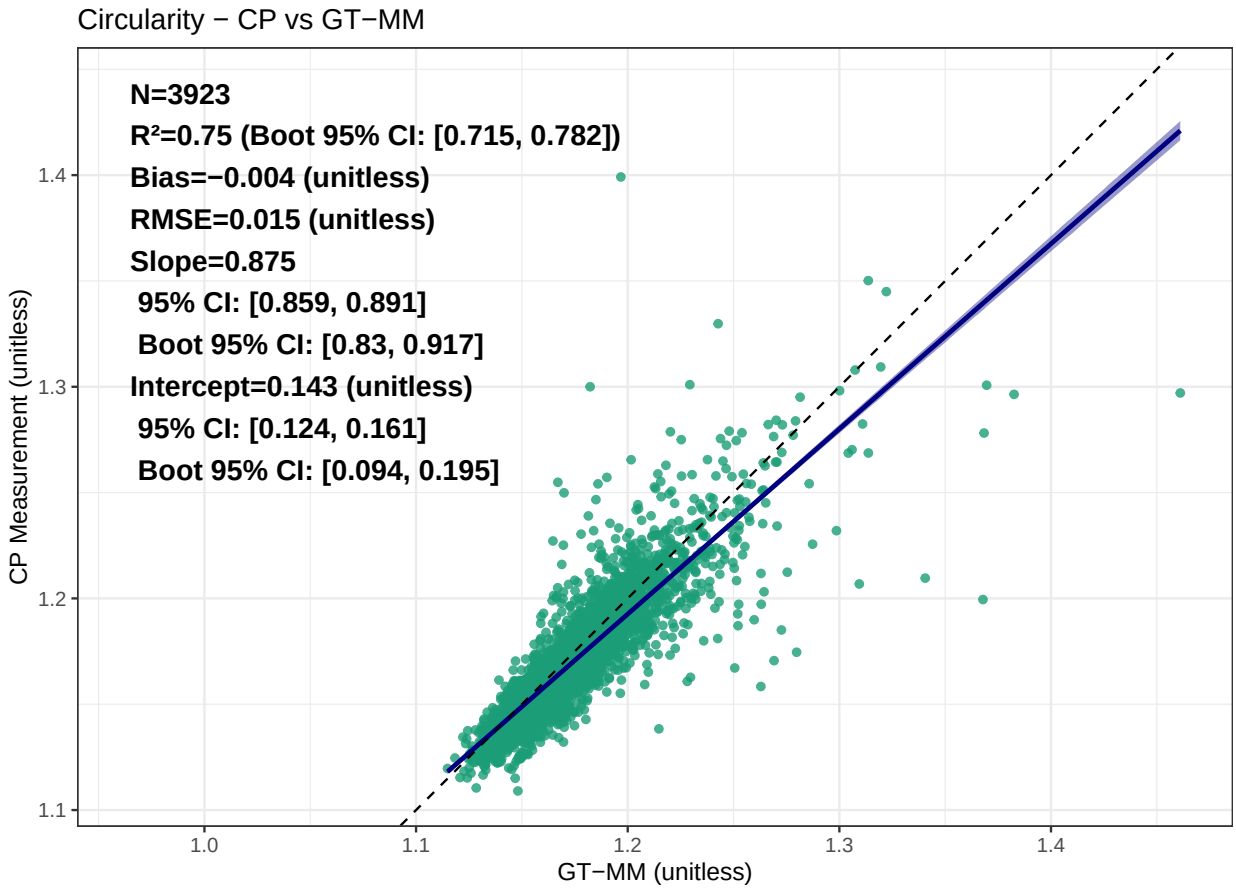
```r
# --- Boxplot Generation (Circularity/Aspect-Ratio) ---
for (measure in measures_to_plot_phase3) {
  cols_to_plot <- grep(paste0("^", measure, "-"), colnames(df_filtered), value = TRUE)

  if (length(cols_to_plot) == 0) {
    warning(paste("No columns found for measure group:", measure, ". Skipping boxplot."))
    next
  }

  p_box <- create_boxplot_new(df_filtered, cols_to_plot, measure)

  if (!is.null(p_box)) {
    filename <- file.path(output_dir, paste0("Boxplot_", measure, "_PHASE3.png"))
    tryCatch({
      invisible(print(p_box))
      ggsave(filename, p_box, width=8, height=6, units="in", dpi=300)
      message(paste("  [SAVED BOXPLOT] ->", basename(filename)))
    }, error = function(e) {
      warning(paste("  [ERROR] Could not save Boxplot for", measure, ". Error:", e$message))
    })
  }
}

# --- Scatterplot and Altman-Bland Plot Generation (Circularity/Aspect-Ratio) ---
for (i in 1:length(scatterplot_pairs)) {
  pair <- scatterplot_pairs[[i]]
  x_col <- pair$x_col
  y_col <- pair$y_col

  measure <- sub("-.*", "", x_col)

  # Use the filtered data for all specified scatterplots
  if (!(x_col %in% colnames(df_filtered)) || !(y_col %in% colnames(df_filtered))) {
    warning(paste("Columns not found in filtered data:", x_col, "or", y_col, ". Skipping plots."))
    next
  }

  # --- SCATTERPLOT ---
  p_scatter <- create_scatterplot_new(df_filtered, x_col, y_col, measure)

  if (!is.null(p_scatter)) {
    measure_for_name <- trimws(sub("-.*", "", x_col))
    gt_method_for_name <- sub(paste0("^", measure_for_name, "-"), "", x_col)
    pred_method_for_name <- sub(paste0("^", measure_for_name, "-"), "", y_col)

    gt_method_for_name <- if (str_detect(gt_method_for_name, "GT-MM")) "GT-MM" else gt_method_for_nam

    base_name <- paste0(measure_for_name, "_", pred_method_for_name, "vs", gt_method_for_name)
    current_filename <- file.path(output_dir, paste0("Scatter_", base_name, "_PHASE3.png"))

    tryCatch({
      invisible(print(p_scatter))
      ggsave(current_filename, p_scatter, width=8, height=8, units="in", dpi=300)
```

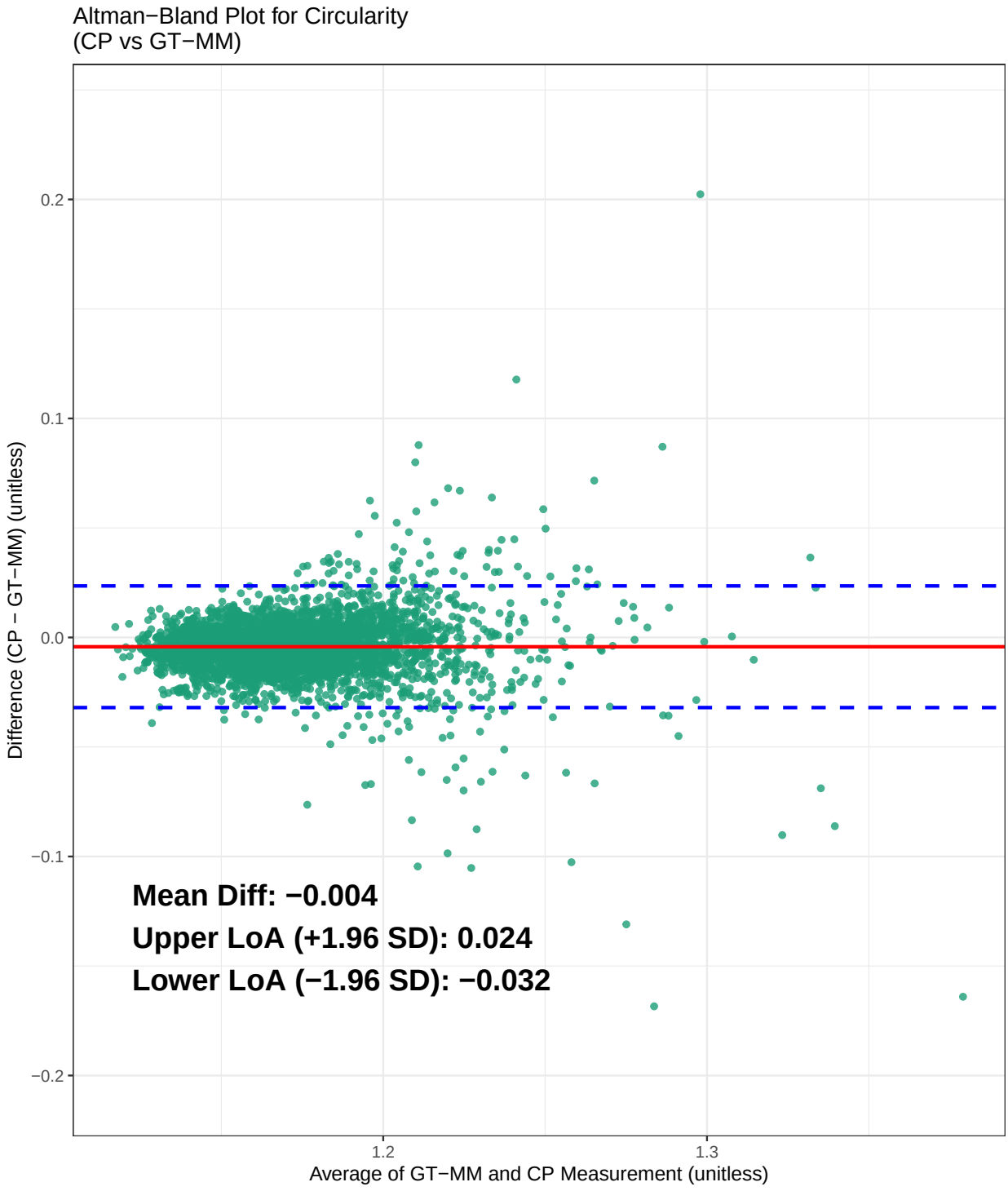
```
        message(paste("  [SAVED SCATTERPLOT] ->", basename(current_filename)))
      }, error = function(e) {
        warning(paste("  [ERROR] Could not save Scatterplot for", base_name, ". Error:", e$message))
      })
    }

    # - ALTMAN-BLAND PLOT -
    p_altman <- create_altman_plot(df_filtered, x_col, y_col, measure)

    if (!is.null(p_altman)) {
      measure_for_name <- trimws(sub("-.*", "", x_col))
      gt_method_for_name <- sub(paste0("^", measure_for_name, "-"), "", x_col)
      pred_method_for_name <- sub(paste0("^", measure_for_name, "-"), "", y_col)

      gt_method_for_name <- if (str_detect(gt_method_for_name, "GT-MM")) "GT-MM" else gt_method_for_nam

      base_name <- paste0(measure_for_name, "_", pred_method_for_name, "vs", gt_method_for_name)
      current_filename <- file.path(output_dir, paste0("AltmanBland_", base_name, "_PHASE3.png"))

      tryCatch({
        invisible(print(p_altman))
        ggsave(current_filename, p_altman, width=8, height=8, units="in", dpi=300)
        message(paste("  [SAVED ALTMAN-BLAND PLOT] ->", basename(current_filename)))
      }, error = function(e) {
        warning(paste("  [ERROR] Could not save Altman-Bland Plot for", base_name, ". Error:", e$messag
      })
    }
  }
  message("\n--- Analysis Complete ---")
}

# --- USER-DEFINED INPUTS (11 PLOTS Total: 3 Boxplots + 4 Scatterplots + 4 Altman-Bland Plots) ---
measure_groups_for_boxplot <- c(
  "Perimeter",
  "Circularity",
  "Aspect-Ratio"
)

scatterplot_pairs <- list(
  list(x_col = "Circularity-GT-MM", y_col = "Circularity-CP"),
  list(x_col = "Circularity-GT-MM", y_col = "Circularity-CP-Affine"),
  list(x_col = "Aspect-Ratio-GT-MM", y_col = "Aspect-Ratio-CP"),
  list(x_col = "Aspect-Ratio-GT-MM", y_col = "Aspect-Ratio-CP-Affine")
)

# --- EXECUTE THE FULL ANALYSIS ---
# Pass the df_original (after manual/numeric cleanup) to the master function
generate_analysis_plots(df_original, measure_groups_for_boxplot, scatterplot_pairs, output_dir)
```

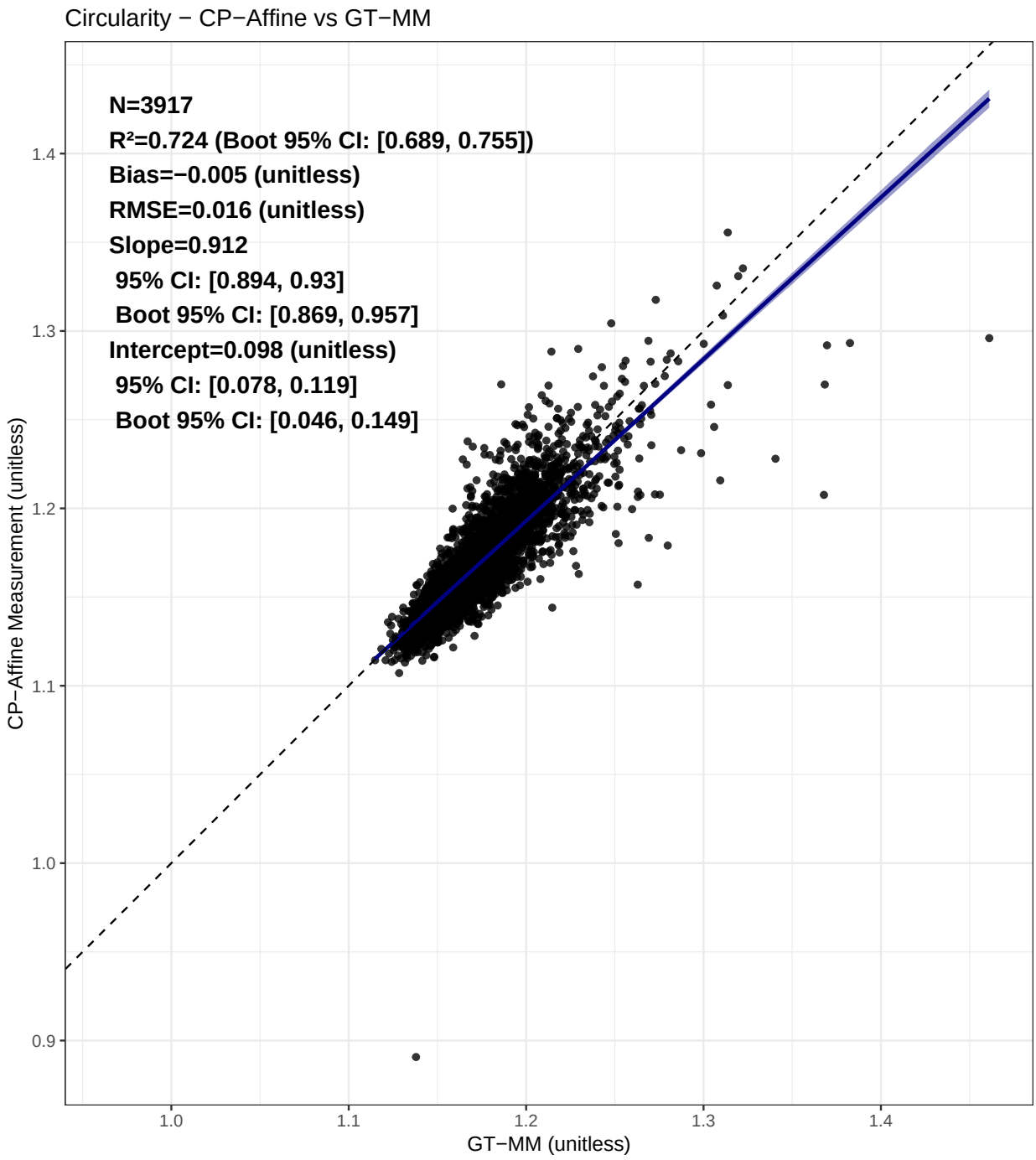

Perimeter Comparison Across Methods
N: 4148
Median: 57.313
Min: 23.907
Max: 127.386
N: 4148
Median: 42.66
Min: 19.828
Max: 81.846
N: 4148
Median: 40.971
Min: 19.376
Max: 78.149
N: 4148
Median: 40.223
Min: 18.698
Max: 76.972
90
60
30
Perimeter (mm)
CP
SVD
min
colorbox
Measurement Method

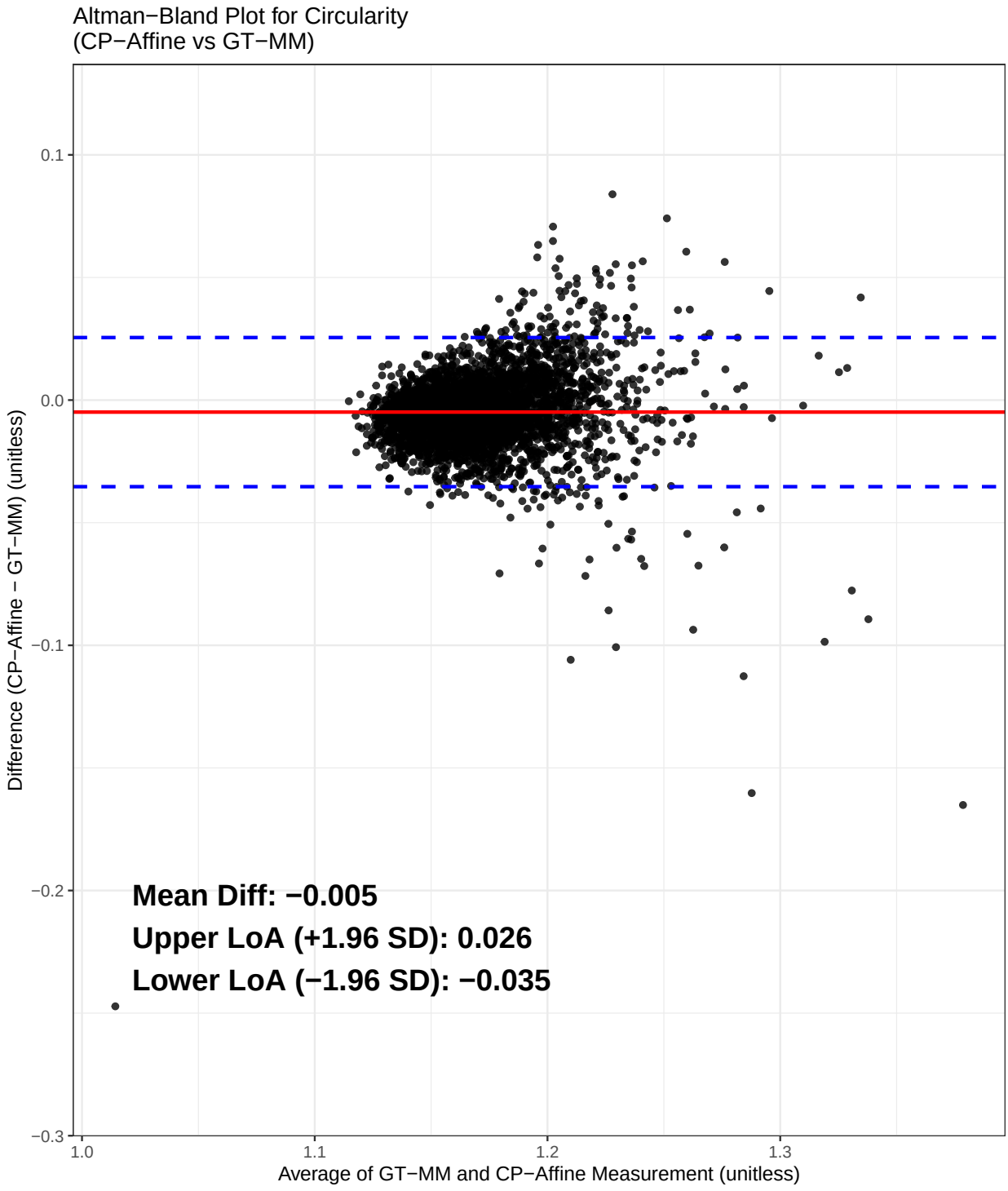

Circularity Comparison Across Methods
1.30
1.25
1.20
1.15
1.10
Circularity (unitless)
N: 3923
Median: 1.171
Min: 1.115
Max: 1.461
N: 3923
Median: 1.166
Min: 1.109
Max: 1.399
GT−MM
CP
Measurement Method

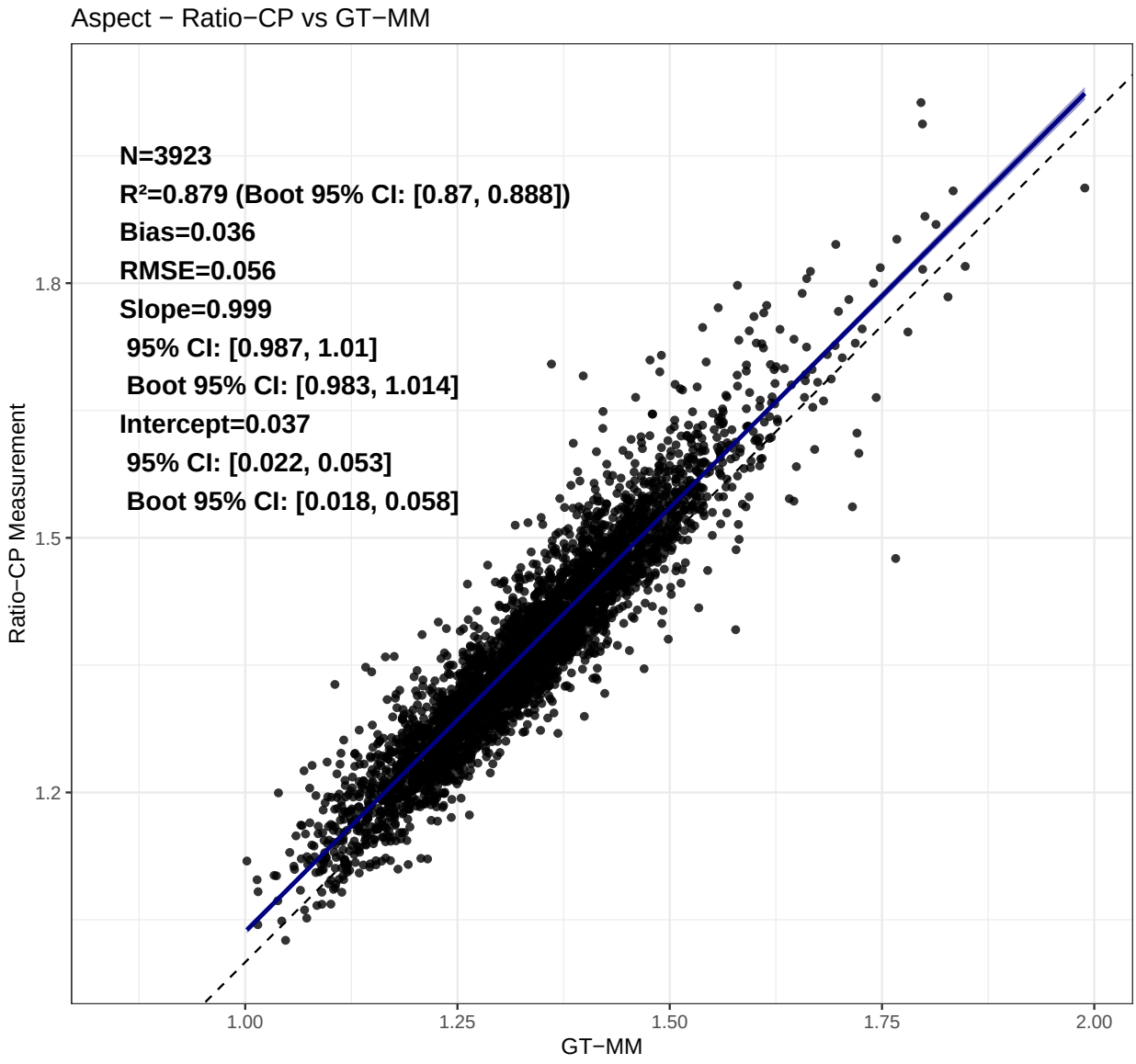

Aspect−Ratio Comparison Across Methods
Aspect−Ratio (unitless)
1.8
1.6
1.4
1.2
1.0
N: 3923
Median: 1.332
Min: 1.002
Max: 1.989
N: 3923
Median: 1.365
Min: 1.026
Max: 2.013
Faba−Seed−CC_Vf602−1−1−0
Faba−Seed−CC_Vf589−3−1−0
GT−MM
CP
Measurement Method

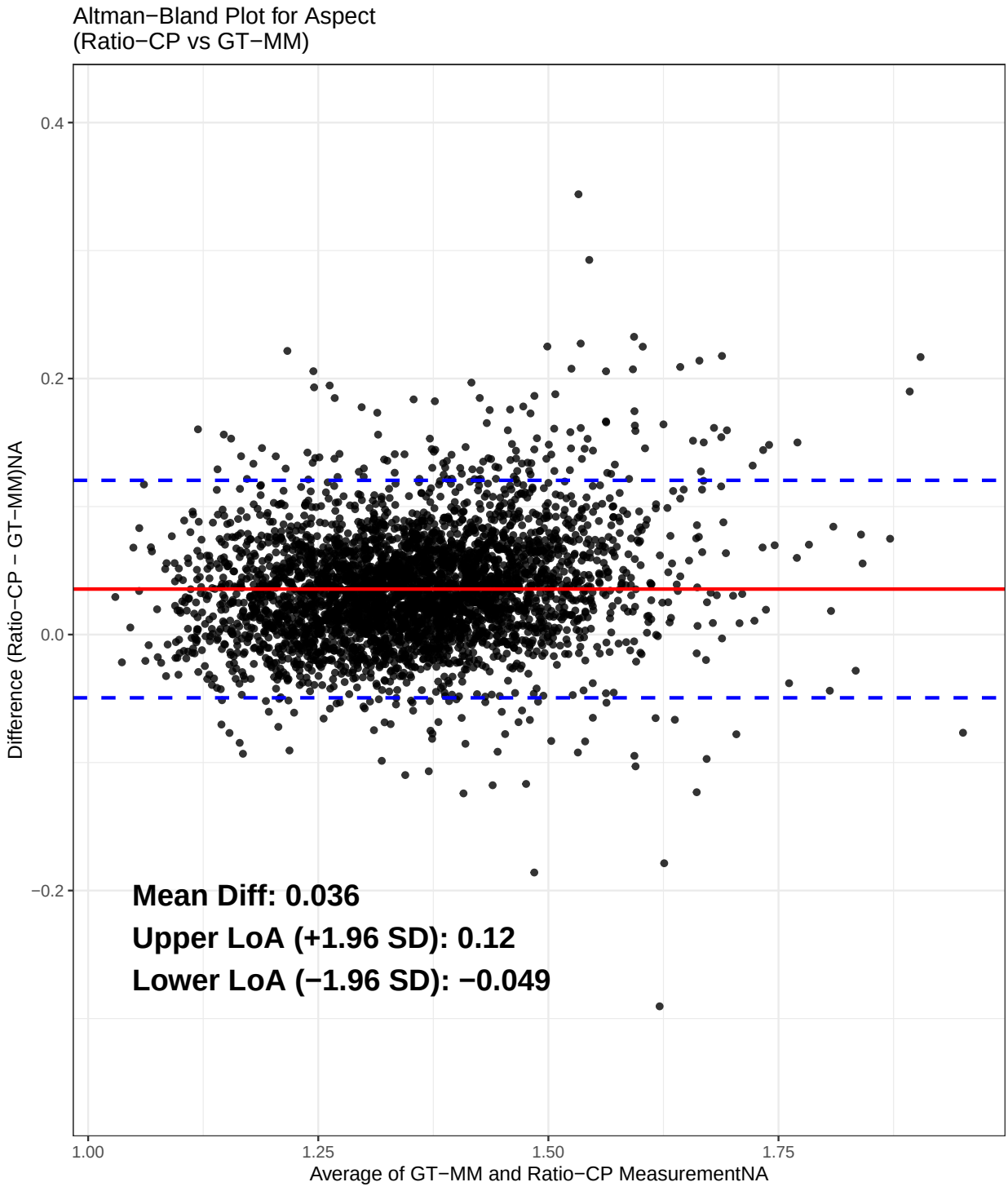

Circularity − CP vs GT−MM
N=3923
R²=0.75 (Boot 95% CI: [0.715, 0.782])
Bias=−0.004 (unitless)
RMSE=0.015 (unitless)
Slope=0.875
95% CI: [0.859, 0.891]
Boot 95% CI: [0.83, 0.917]
Intercept=0.143 (unitless)
95% CI: [0.124, 0.161]
Boot 95% CI: [0.094, 0.195]
CP Measurement (unitless)
GT−MM (unitless)
1.1
1.2
1.3
1.4
1.0
1.1
1.2
1.3
1.4

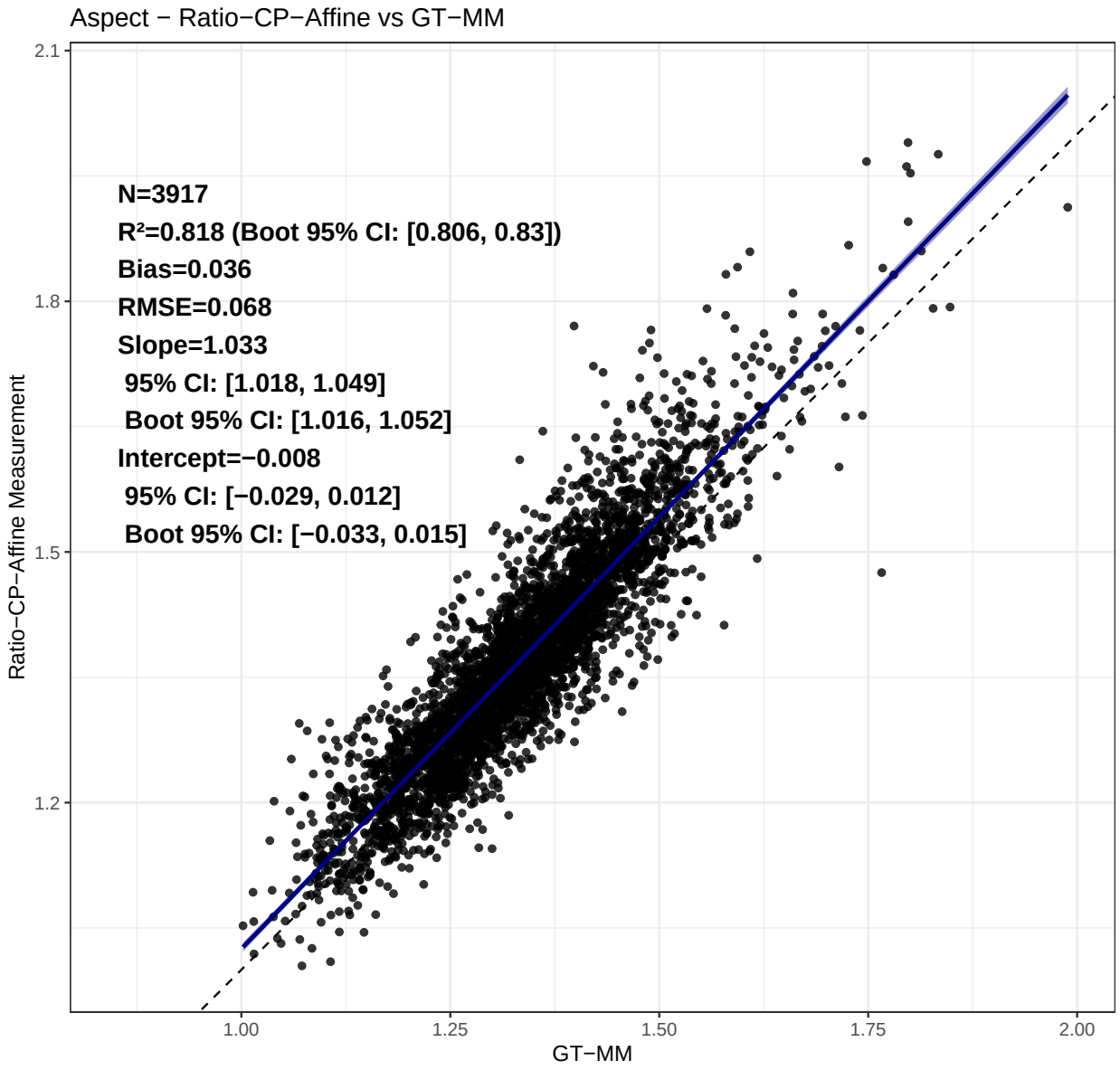

Altman−Bland Plot for Circularity
(CP vs GT−MM)
Difference (CP − GT−MM) (unitless)
0.2
0.1
0.0
−0.1
−0.2
Mean Diff: −0.004
Upper LoA (+1.96 SD): 0.024
Lower LoA (−1.96 SD): −0.032
1.2
1.3
Average of GT−MM and CP Measurement (unitless)

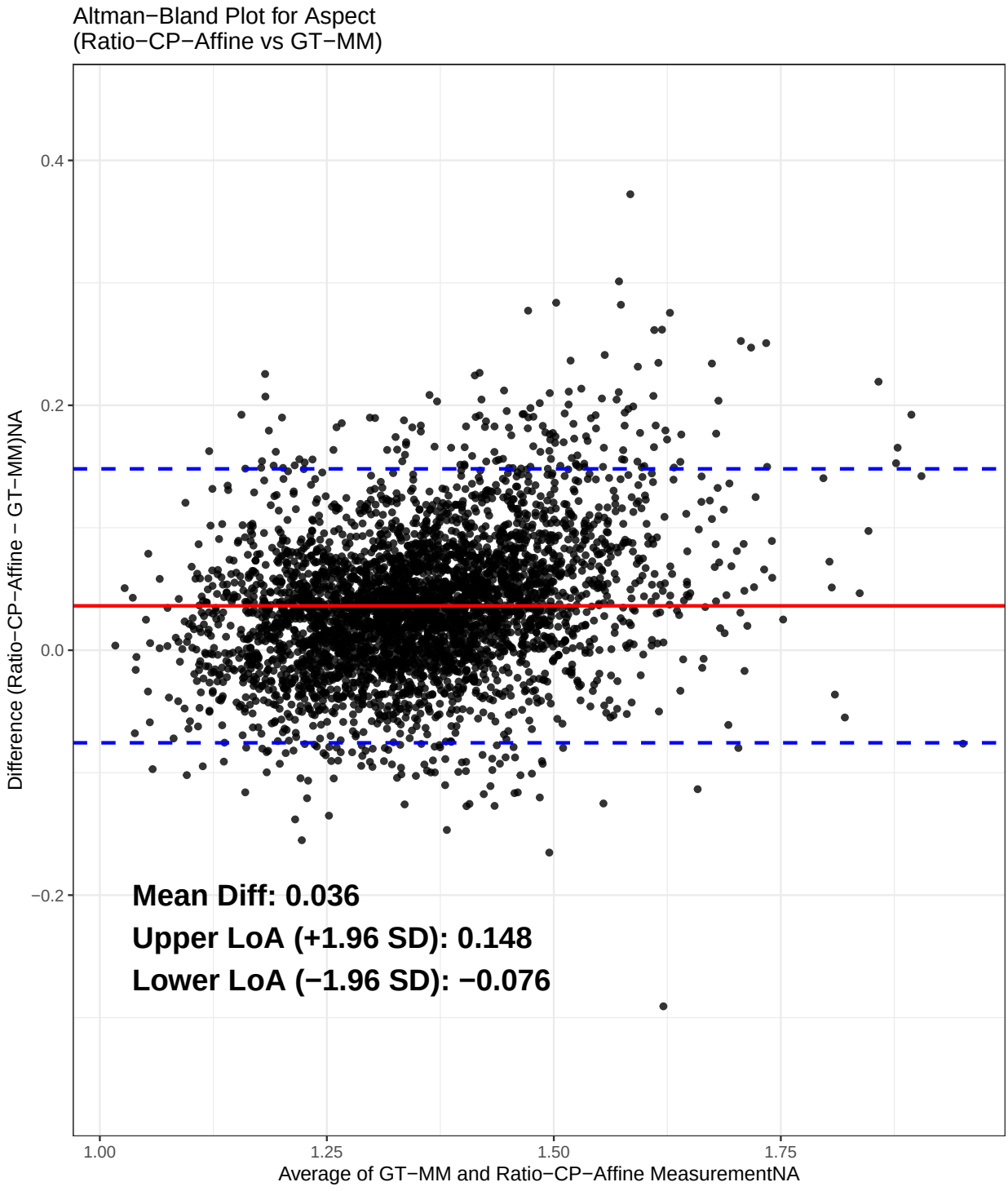

Circularity − CP−Affine vs GT−MM
N=3917
R²=0.724 (Boot 95% CI: [0.689, 0.755])
Bias=−0.005 (unitless)
RMSE=0.016 (unitless)
Slope=0.912
95% CI: [0.894, 0.93]
Boot 95% CI: [0.869, 0.957]
Intercept=0.098 (unitless)
95% CI: [0.078, 0.119]
Boot 95% CI: [0.046, 0.149]
CP−Affine Measurement (unitless)
GT−MM (unitless)
1.4
1.3
1.2
1.1
1.0
0.9
1.0
1.1
1.2
1.3
1.4


Altman−Bland Plot for Circularity
(CP−Affine vs GT−MM)
Difference (CP−Affine − GT−MM) (unitless)
Average of GT−MM and CP−Affine Measurement (unitless)
0.1
0.0
−0.1
−0.2
−0.3
1.0
1.1
1.2
1.3
Mean Diff: −0.005
Upper LoA (+1.96 SD): 0.026
Lower LoA (−1.96 SD): −0.035


Aspect − Ratio−CP vs GT−MM
N=3923
R²=0.879 (Boot 95% CI: [0.87, 0.888])
Bias=0.036
RMSE=0.056
Slope=0.999
95% CI: [0.987, 1.01]
Boot 95% CI: [0.983, 1.014]
Intercept=0.037
95% CI: [0.022, 0.053]
Boot 95% CI: [0.018, 0.058]
Ratio−CP Measurement
1.8
1.5
1.2
1.00
1.25
1.50
1.75
2.00
GT−MM


Altman−Bland Plot for Aspect
(Ratio−CP vs GT−MM)
Difference (Ratio−CP − GT−MM)NA
0.4
0.2
0.0
−0.2
Mean Diff: 0.036
Upper LoA (+1.96 SD): 0.12
Lower LoA (−1.96 SD): −0.049
1.00
1.25
1.50
1.75
Average of GT−MM and Ratio−CP MeasurementNA


Aspect − Ratio−CP−Affine vs GT−MM
N=3917
R²=0.818 (Boot 95% CI: [0.806, 0.83])
Bias=0.036
RMSE=0.068
Slope=1.033
95% CI: [1.018, 1.049]
Boot 95% CI: [1.016, 1.052]
Intercept=−0.008
95% CI: [−0.029, 0.012]
Boot 95% CI: [−0.033, 0.015]
Ratio−CP−Affine Measurement
GT−MM
2.1
1.8
1.5
1.2
1.00
1.25
1.50
1.75
2.00


Altman−Bland Plot for Aspect
(Ratio−CP−Affine vs GT−MM)
Difference (Ratio−CP−Affine − GT−MM)NA
0.4
0.2
0.0
−0.2
Mean Diff: 0.036
Upper LoA (+1.96 SD): 0.148
Lower LoA (−1.96 SD): −0.076
1.00
1.25
1.50
1.75
Average of GT−MM and Ratio−CP−Affine MeasurementNA