MCP ANALYTICS

A/B Test Analysis

Statistical analysis report with 8 sections — interactive charts, validated methodology, and AI-generated insights.

Generated **July 18, 2026**

Sections **8**

Platform **mcpanalytics.ai**

Contents

EXECUTIVE SUMMARY

The Verdict

Key Metrics

Observations	**2000**
Variants	**2**
Control Group	**A**
Best Variant	**B**
Absolute Difference	**0.03**
P Value	**0.0421**
Significant	**yes**

Suggested Interpretation

Variant B wins. B lifts the conversion rate by 0.03 percentage points (absolute), or 30% relative to A's baseline. With p = 0.0421, this difference is statistically significant at the 95% confidence level. The evidence is reliable: the true effect lies between 0.11 and 5.89 percentage points with 95% confidence, and zero is excluded from that interval. This is not a marginal call—the test result decisively favors B.

OVERVIEW

Analysis Overview

A/B test of 2,000 observations across 2 variants of Test Group.

Configuration

N Observations	2000
N Variants	2
Outcome Type	binary
Control Group	A

Suggested Interpretation

This A/B test compares 2 variants across 2,000 observations, with Converted as a binary outcome (success = 1). Variant A serves as the control. A two-proportion z-test was selected because the outcome is binary and there are only 2 groups. Each variant receives a conversion rate with a Wilson 95% confidence interval. The 95% confidence interval on the difference (0.0011 to 0.0589) represents the range of plausible true effects; because it excludes zero, the difference is statistically significant at the 95% level. Statistical significance confirms the effect is reliable, though it does not measure business or practical importance.

DATA PREPARATION

Data Quality

Row filtering, variant-group screening, and outcome type detection.

Data Quality

Initial Rows	2000
Final Rows	2000
Rows Removed	0
Variants Kept	2
Variants Dropped	0

Suggested Interpretation

All 2,000 rows were retained; no records were dropped. Both variants (A and B) were retained and usable. The Converted field contains only 0/1 values and was correctly identified as a conversion flag, with success coded as 1. Data quality was clean across all groups of Test Group. No rows or variants were excluded from the analysis, ensuring the full dataset of 2,000 observations was available for statistical testing.

TABLE 4

Variant Summary

Sample size and conversion rate per Test Group with 95% confidence intervals.

Variant	N	Rate Or Mean	CI Low	CI High
A	1000	0.1	0.0829	0.1202
B	1000	0.13	0.1106	0.1523

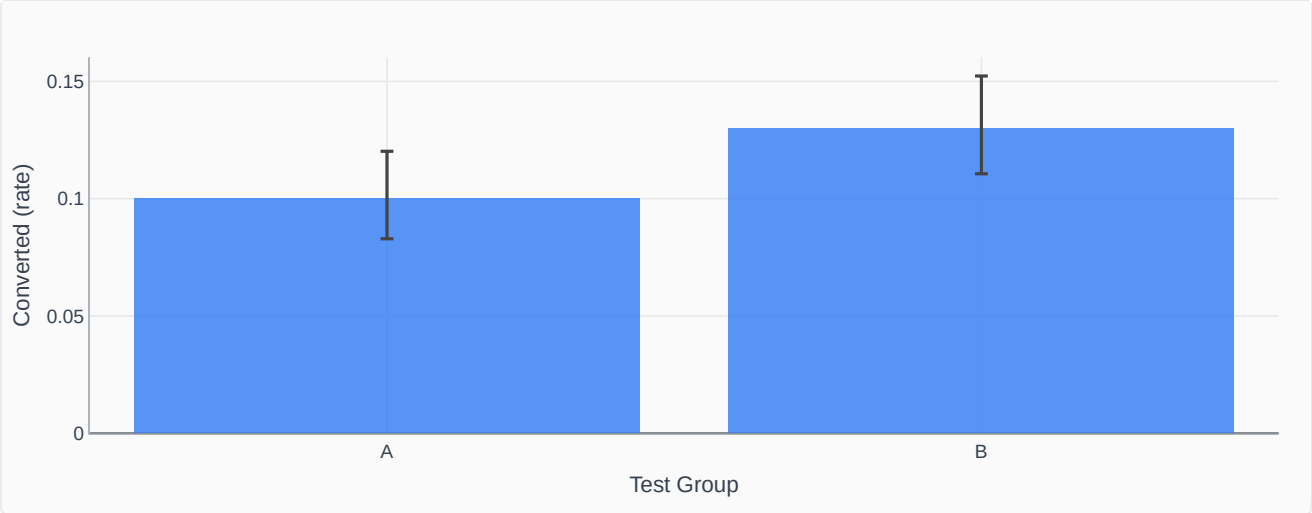
Suggested Interpretation

Variant A (control, n = 1,000) converts at 0.1 with a 95% CI of [0.0829, 0.1202]. Variant B (n = 1,000) converts at 0.13 with a 95% CI of [0.1106, 0.1523]. The groups are evenly sized. B's point estimate (0.13) sits 3 percentage points above A's (0.1). Although the confidence intervals overlap visually, the formal hypothesis test confirms the difference is statistically significant. B's interval is shifted higher, consistent with a genuine uplift in conversion.

FIGURE 5

Outcome by Variant

Converted per Test Group, with 95% confidence intervals as error bars.



Suggested Interpretation

B leads at a conversion rate of 0.13 versus A at 0.1. Visually, the 95% confidence interval error bars overlap: A spans [0.0829, 0.1202] and B spans [0.1106, 0.1523]. Despite this overlap, the formal two-proportion z-test (p = 0.0421) is the deciding metric. Overlapping error bars do not imply statistical equivalence; the test correctly accounts for sample size and variance structure. B's higher point estimate and significant p-value confirm B outperforms A.

TABLE 6

Statistical Test Results

Every comparison vs A with uplift, 95% CI, and p-value.

Comparison	Absolute Diff	Relative Uplift PCT	CI Low	CI High	P Value	Significant
B vs A	0.03	30	0.0011	0.0589	0.0421	yes

Suggested Interpretation

One comparison was tested: B vs A. The absolute difference is 0.03 (B converts 3 percentage points higher). Relative uplift is 30% of A's baseline rate. The 95% confidence interval on the difference is [0.0011, 0.0589], excluding zero. The p-value is 0.0421, meeting the 95% significance threshold ($p < 0.05$). This is the only test conducted, and it is statistically significant. B's conversion advantage over A is reliable and not attributable to random sampling variation.

METHODOLOGY

Methodology

Statistical method, assumptions, and diagnostics

A/B Test Analysis

Standard-library analysis: upload row-level experiment data (one row per user or session), map the variant column and the outcome column, and get a definitive read on your test. The tool detects whether your outcome is a conversion flag or a numeric value, picks the right statistical test (two-proportion, chi-square, Welch t-test, or ANOVA), and returns the uplift with a confidence interval and a plain-English verdict on significance.

> DATA
>
> **N = 2000** observations

KEY RESULTS

Observations	2000
Variants	2
Control Group	A
Best Variant	B
Absolute Difference	0.0300
p-value	0.0421
Significant	yes

ASSUMPTIONS

- One row per independent user or session (no repeated measures of the same user across rows)
- Variants were assigned randomly
- For numeric outcomes, group means are meaningful summaries (Welch tests are robust to unequal variances)

LIMITATIONS

- Significance is not business impact — a tiny but significant lift may not be worth shipping
- Peeking repeatedly at a running test inflates false positives; this is a fixed-horizon test
- Rows are assumed independent — session-level rows from the same user violate this
- Rare variants beyond the top 4, and groups with fewer than 5 rows, are excluded

SOFTWARE & CITATION

MCP Analytics · mcpanalytics.ai

APPENDIX

Analysis Code

Complete R source code for this analysis

A/B Test Analysis — Is the Difference Real?

Takes row-level experiment data (one row per user/session) with a variant column and an outcome column, detects the outcome type, runs the right significance test, and reports the uplift with a 95% confidence interval and a plain-English verdict.

A.1 Why This Method?

The correct test depends on the data: a 0/1 conversion outcome needs a two-proportion test (chi-square + pairwise vs control for 3+ variants); a numeric outcome needs a Welch t-test (ANOVA + Holm-adjusted pairwise Welch for 3+ variants). Automating that choice removes the most common A/B analysis mistakes.

A.2 What This Analysis Covers

- Per-variant conversion rate (Wilson CI) or mean (t-based CI)
- Absolute + relative uplift vs control with a 95% CI on the difference
- Overall + pairwise significance tests, Cohen's d for numeric outcomes
- An honest verdict, including a rough 80%-power sample-size estimate

when the result is not significant

A.3 Standard Library

Platform standard-library module (LAT-1441): runs on ANY dataset via the semantic mapping {variant, outcome}. All narrative is derived from the user's own column names and computed values.

```
suppressPackageStartupMessages(library(DT))
suppressPackageStartupMessages(library(htmlwidgets))
suppressPackageStartupMessages(library(arrow))
suppressPackageStartupMessages(library(knitr))
suppressPackageStartupMessages(library(rmarkdown))
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(tidyr))
suppressPackageStartupMessages(library(ggplot2))
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(lubridate))
suppressPackageStartupMessages(library(broom))
suppressPackageStartupMessages(library(Matrix))
suppressPackageStartupMessages(library(cluster))
suppressPackageStartupMessages(library(data.table))
```

A.4 Core Analysis Pipeline

```
compute_shared <- function(df, params, col_map = list()) {
  # === SHARED EXPORTS ===
  #   initial_rows/final_rows/rows_removed   $ row accounting
  #   variant_h / outcome_h    $ humanized names of the two mapped columns
  #   outcome_type             $ "binary" | "numeric"
  #   success_label/success_how $ how the binary success level was chosen
  #   kept_levels/dropped_levels $ variant groups used / excluded (reported)
  #   n_missing/n_uncoercible/n_dropped_variant_rows $ drop accounting
  #   control / treatments / best_variant $ group roles
  #   gs                       $ per-group stats list (n, est, ci, sd, x)
  #   variant_summary_df       $ variant, n, rate_or_mean, ci_low, ci_high
  #   outcome_by_variant_df    $ variant, value, ci_low, ci_high
  #   test_results_df          $ comparison, absolute_diff, relative_uplift_pct,
  #                              ci_low, ci_high, p_value, significant
  #   primary                  $ headline comparison (best variant vs control)
  #   overall                  $ overall test for 3+ variants (label, p) or NULL
  #   sig / verdict / power_n  $ significance flag, verdict text, 80%-power n
  #   method_name / p_adjust_note
  #   metrics / json_output
  # === /SHARED EXPORTS ===

  variant_h <- humanize_semantic("variant", col_map)
  outcome_h <- humanize_semantic("outcome", col_map)
```

Step 1: Validate mapping + drop rows with missing values

```
initial_rows <- nrow(df)
  if (!"variant" %in% names(df)) {
    stop(sprintf("column_mapping must map &#x27;variant' (your %s column).", variant_h))
  }
  if (!"outcome" %in% names(df)) {
    stop(sprintf("column_mapping must map &#x27;outcome' (your %s column).", outcome_h))
  }
  if (initial_rows < 20) {
    stop(sprintf("Only %d rows — an A/B test needs at least 20 rows(and 5+ per %s).",
                 initial_rows, variant_h))
  }

  v_all <- trimws(as.character(df$variant))
  o_all <- df$outcome
  o_is_logical <- is.logical(o_all)
  o_chr_all <- trimws(as.character(o_all))
  ok <- !is.na(v_all) & v_all != "" & !is.na(o_all) & !is.na(o_chr_all) & o_chr_all != ""
  n_missing <- sum(!ok)
  v <- v_all[ok]
  o_chr <- o_chr_all[ok]
  o_log <- if (o_is_logical) o_all[ok] else NULL
```

Step 2: Variant groups — keep the top 4 by size, drop groups with n<5

```
tab <- sort(table(v), decreasing = TRUE)
  top_levels <- names(tab)[seq_len(min(4L, length(tab)))]
  beyond_top <- setdiff(names(tab), top_levels)
  small <- top_levels[as.integer(tab[top_levels]) < 5]
  kept_levels <- setdiff(top_levels, small)
  dropped_levels <- c(beyond_top, small)
  sel <- v %in% kept_levels
  n_dropped_variant_rows <- sum(!sel)
  v <- v[sel]
  o_chr <- o_chr[sel]
  if (!is.null(o_log)) o_log <- o_log[sel]
  if (length(kept_levels) < 2) {
    stop(sprintf("After cleaning, %s has %d usable group(s) — need at least 2 variants with 5+ rows each.",
                 variant_h, length(kept_levels)))
  }
```

Step 3: Outcome type detection (binary conversion vs numeric value)

```
  outcome_type <- NULL
    success_label <- NA_character_
    success_how <- ""
    n_uncoercible <- 0L
    y <- NULL
    if (o_is_logical) {
      y <- as.numeric(o_log)
      outcome_type <- "binary"
      success_label <- "TRUE"
      success_how <- sprintf("%s is a logical(TRUE/FALSE) column; success = TRUE.", outcome_h)
    } else {
      conv <- suppressWarnings(as.numeric(o_chr))
      if (sum(!is.na(conv)) >= 0.95 * length(o_chr)) {
        bad <- is.na(conv)
        n_uncoercible <- sum(bad)
        if (n_uncoercible > 0) {
          v <- v[!bad]; conv <- conv[!bad]
        }
        uu <- unique(conv)
        if (all(uu %in% c(0, 1))) {
          y <- conv
          outcome_type <- "binary"
          success_label <- "1"
          success_how <- sprintf("%s contains only 0/1 values — treated as a conversion flag; success = 1.",
                                 outcome_h)
        } else {
          y <- conv
          outcome_type <- "numeric"
          success_how <- sprintf("%s is numeric — comparing group means.", outcome_h)
        }
      } else {
        lv <- sort(unique(o_chr))
        if (length(lv) != 2) {
          stop(sprintf("%s has %d distinct non-numeric values — map a 0/1 conversion flag, a two-level yes/no column, or a numeric
                       outcome_h, length(lv)))
        }
        kw <- c("yes", "true", "converted", "1", "success", "clicked", "purchased")
        hit <- lv[tolower(lv) %in% kw]
        if (length(hit) >= 1) {
          success_label <- hit[1]
          success_how <- sprintf("%s has two values(&#x27;%s' / '%s'); success = '%s' (matched a success keyword).",
                                 outcome_h, lv[1], lv[2], success_label)
        } else {
          success_label <- lv[2]
          success_how <- sprintf("%s has two values(&#x27;%s' / '%s'); success = '%s' (the alphabetically-last level — no obvious
                                 outcome_h, lv[1], lv[2], success_label)
        }
        y <- as.numeric(o_chr == success_label)
        outcome_type <- "binary"
      }
    }
```

Re-check group sizes after any outcome-row drops

```
tab2 <- table(v)
  final_small <- names(tab2)[as.integer(tab2) < 5]
  if (length(final_small) > 0) {
    dropped_levels <- c(dropped_levels, final_small)
    sel2 <- !(v %in% final_small)
    n_dropped_variant_rows <- n_dropped_variant_rows + sum(!sel2)
    v <- v[sel2]; y <- y[sel2]
    kept_levels <- setdiff(kept_levels, final_small)
  }
  if (length(kept_levels) < 2) {
    stop(sprintf("After cleaning, %s has fewer than 2 variants with 5+ rows — cannot run a comparison.",
                 variant_h))
  }
  if (outcome_type == "binary") {
    if (all(y == 1) || all(y == 0)) {
      stop(sprintf("%s is constant(every usable row = %s) — there is nothing to compare.",
                   outcome_h, y[1]))
    }
  } else if (is.na(var(y)) || isTRUE(var(y) == 0)) {
    stop(sprintf("%s has zero variance — there is nothing to compare.", outcome_h))
  }

  final_rows <- length(y)
  rows_removed <- initial_rows - final_rows
  if (final_rows < 20) {
    stop(sprintf("Only %d usable rows after cleaning — need at least 20.", final_rows))
  }
```

Step 4: Per-variant estimates (Wilson CI for rates, t-based CI for means)

```
wilson_ci <- function(x, n, conf = 0.95) {
    z <- qnorm(1 - (1 - conf) / 2)
    p <- x / n
    denom <- 1 + z^2 / n
    centre <- (p + z^2 / (2 * n)) / denom
    half <- z * sqrt(p * (1 - p) / n + z^2 / (4 * n^2)) / denom
    c(max(0, centre - half), min(1, centre + half))
  }
```

Order groups by size (largest first)

```
grp_n <- as.integer(table(v)[kept_levels])
  names(grp_n) <- kept_levels
  kept_levels <- kept_levels[order(-grp_n)]
  grp_n <- grp_n[kept_levels]

  gs <- lapply(kept_levels, function(g) {
    yy <- y[v == g]
    n <- length(yy)
    if (outcome_type == "binary") {
      x <- sum(yy)
      ci <- wilson_ci(x, n)
      list(variant = g, n = n, est = x / n, ci_low = ci[1], ci_high = ci[2],
           x = x, sd = NA_real_)
    } else {
      m <- mean(yy)
      s <- sd(yy)
      half <- if (n > 1 && is.finite(s)) qt(0.975, n - 1) * s / sqrt(n) else NA_real_
      list(variant = g, n = n, est = m, ci_low = m - half, ci_high = m + half,
           x = NA_real_, sd = s)
    }
  })
  names(gs) <- kept_levels
```

Step 5: Control group — a level literally named control/a/baseline, else largest n

```
named_ctrl <- kept_levels[tolower(kept_levels) %in% c("control", "a", "baseline")]
  control <- if (length(named_ctrl) > 0) named_ctrl[1] else kept_levels[1]
  control_how <- if (length(named_ctrl) > 0) {
    sprintf("&#x27;%s' (named like a control group)", control)
  } else {
    sprintf("&#x27;%s' (the largest group)", control)
  }
  treatments <- setdiff(kept_levels, control)
  cg <- gs[[control]]
```

Step 6: Significance tests — pairwise vs control (+ overall for 3+ variants)

```
comp_rows <- list()
  for (g in treatments) {
    tg <- gs[[g]]
    d <- tg$est - cg$est
    eff <- NA_real_
    if (outcome_type == "binary") {
      pt <- tryCatch(suppressWarnings(prop.test(c(tg$x, cg$x), c(tg$n, cg$n))),
                     error = function(e) NULL)
      p <- if (!is.null(pt)) pt$p.value else NA_real_
      ci <- if (!is.null(pt)) as.numeric(pt$conf.int) else c(NA_real_, NA_real_)
    } else {
      tt <- tryCatch(t.test(y[v == g], y[v == control]), error = function(e) NULL)
      p <- if (!is.null(tt)) tt$p.value else NA_real_
      ci <- if (!is.null(tt)) as.numeric(tt$conf.int) else c(NA_real_, NA_real_)
      sp <- sqrt(((tg$n - 1) * tg$sd^2 + (cg$n - 1) * cg$sd^2) / (tg$n + cg$n - 2))
      eff <- if (is.finite(sp) && sp > 0) d / sp else NA_real_
    }
    rel <- if (is.finite(cg$est) && cg$est != 0) 100 * d / cg$est else NA_real_
    comp_rows[[g]] <- list(comparison = paste(g, "vs", control), variant = g,
                           d = d, rel = rel, ci_low = ci[1], ci_high = ci[2],
                           p = p, effect = eff)
  }

  p_adjust_note <- ""
  if (outcome_type == "numeric" && length(treatments) > 1) {
    ps <- sapply(comp_rows, function(r) r$p)
    adj <- p.adjust(ps, method = "holm")
    for (i in seq_along(comp_rows)) comp_rows[[i]]$p <- as.numeric(adj[i])
    p_adjust_note <- "Pairwise p-values are Holm-adjusted for multiple comparisons."
  }

  overall <- NULL
  if (length(kept_levels) > 2) {
    if (outcome_type == "binary") {
      ch <- tryCatch(suppressWarnings(chisq.test(table(v, y))), error = function(e) NULL)
      if (!is.null(ch)) overall <- list(label = "Overall(chi-square, all variants)",
                                        p = as.numeric(ch$p.value))
    } else {
      av <- tryCatch(summary(aov(y ~ factor(v))), error = function(e) NULL)
      p_over <- tryCatch(av[[1]][["Pr(>F)"]][1], error = function(e) NA_real_)
      if (!is.na(p_over)) overall <- list(label = "Overall(one-way ANOVA, all variants)",
                                          p = as.numeric(p_over))
    }
  }

  method_name <- if (outcome_type == "binary") {
    if (length(kept_levels) == 2) "two-proportion z-test(prop.test)"
    else "chi-square across all variants + pairwise two-proportion tests vs control"
  } else {
    if (length(kept_levels) == 2) "Welch two-sample t-test with Cohen&#x27;s d"
    else "one-way ANOVA + pairwise Welch t-tests vs control(Holm-adjusted)"
  }
```

Step 7: Headline comparison — the best-performing treatment vs control

```
ests <- sapply(treatments, function(g) gs[[g]]$est)
  ests <- ests[!is.na(ests)]          # never index which.max over possible NAs
  best_variant <- if (length(ests) > 0) names(ests)[which.max(ests)] else treatments[1]
  primary <- comp_rows[[best_variant]]
  sig <- isTRUE(!is.na(primary$p) && primary$p < 0.05)
```

80%-power sample-size estimate for the observed effect (only quoted when the result is not significant); tryCatch — skipped entirely on error.

```
power_n <- NULL
  if (!sig) {
    power_n <- tryCatch({
      n_est <- if (outcome_type == "binary") {
        ceiling(power.prop.test(p1 = cg$est, p2 = gs[[best_variant]]$est,
                                power = 0.80, sig.level = 0.05)$n)
      } else {
        tg <- gs[[best_variant]]
        sp <- sqrt(((tg$n - 1) * tg$sd^2 + (cg$n - 1) * cg$sd^2) / (tg$n + cg$n - 2))
        ceiling(power.t.test(delta = abs(primary$d), sd = sp,
                             power = 0.80, sig.level = 0.05)$n)
      }
      if (is.finite(n_est) && n_est > 0) n_est else NULL
    }, error = function(e) NULL)
  }
```

Step 8: The verdict (plain English, fully computed)

```
fmt_p <- function(p) {
    if (is.na(p)) "p unavailable"
    else if (p < 0.001) "p < 0.001"
    else paste0("p = ", signif(p, 2))
  }
  verb <- if (primary$d >= 0) "lifts" else "lowers"
  if (outcome_type == "binary") {
    gap_txt <- sprintf("%.1f pts", abs(100 * primary$d))
    rel_txt <- if (is.finite(primary$rel)) sprintf(" (rel %+.1f%%)", primary$rel) else ""
  } else {
    gap_txt <- sprintf("%.4g", abs(primary$d))
    rel_txt <- if (is.finite(primary$rel)) sprintf(" (rel %+.1f%%)", primary$rel) else ""
  }
  verdict <- sprintf("%s %s %s by %s%s%s, %s — %s at the 95%% confidence level.",
                     best_variant, verb, outcome_h,
                     if (primary$d >= 0) "+" else "-", gap_txt, rel_txt,
                     fmt_p(primary$p),
                     if (sig) "statistically significant" else "NOT statistically significant")
  if (!sig) {
    verdict <- paste0(verdict,
      " In practice: the data cannot distinguish this gap from random chance —",
      " which is not the same as proving the variants are equal.")
    if (!is.null(power_n)) {
      verdict <- paste0(verdict, sprintf(
        " At the observed effect size, roughly %s rows per variant(~%s total) would give an 80%% chance of detecting a real diff
        format(power_n, big.mark = ","), format(2 * power_n, big.mark = ",")))
    }
  }
```

Step 10: Metrics + machine answer

```
    metrics <- list(
        `Observations`        = final_rows,
        `Variants`            = length(kept_levels),
        `Control Group`       = control,
        `Best Variant`        = best_variant,
        `Absolute Difference` = round(primary$d, 4),
        `P Value`             = signif(primary$p, 3),
        `Significant`         = if (sig) "yes" else "no"
      )

      json_output <- list(
        answer = paste0(
          "A/B test on ", format(final_rows, big.mark = ","), " rows across ",
          length(kept_levels), " variants of ", variant_h, ", measuring ", outcome_h,
          " (", if (outcome_type == "binary") "conversion rate" else "numeric mean", "). ",
          "Method: ", method_name, "; control = ", control_how, ". ", verdict,
          if (!is.null(overall)) paste0(" ", overall$label, ": ", fmt_p(overall$p), ".") else ""
        ),
        cards = lapply(
          c("tldr", "overview", "preprocessing", "variant_summary",
            "outcome_by_variant", "test_results"),
          function(cid) list(id = cid, metrics = metrics)
        )
      )

      list(
        initial_rows = initial_rows, final_rows = final_rows, rows_removed = rows_removed,
        variant_h = variant_h, outcome_h = outcome_h,
        outcome_type = outcome_type, success_label = success_label, success_how = success_how,
        kept_levels = kept_levels, dropped_levels = dropped_levels,
        n_missing = n_missing, n_uncoercible = n_uncoercible,
        n_dropped_variant_rows = n_dropped_variant_rows,
        control = control, control_how = control_how,
        treatments = treatments, best_variant = best_variant,
        gs = gs, comp_rows = comp_rows, primary = primary, overall = overall,
        sig = sig, verdict = verdict, power_n = power_n,
        method_name = method_name, p_adjust_note = p_adjust_note,
        variant_summary_df = variant_summary_df,
        outcome_by_variant_df = outcome_by_variant_df,
        test_results_df = test_results_df,
        metrics = metrics, json_output = json_output
      )
    }
```

This is the exact R code executed to produce the analysis above. Run it with the same dataset to reproduce these results.