Simulation Studies for $eJAB_{01}$

Evan Strasdin, …

2025-10-24

Overview

The simulation experiments reported in Velidi et al. (2025) provide preliminary validation of the $eJAB_{01}$ statistic. The present study extends that framework through a comprehensive and fully reproducible simulation program designed to characterize the finite-sample and asymptotic behavior of $eJAB_{01}$ across a broad class of models. Specifically, we quantify its operating characteristics—type I error control, power, bias, and root-mean-square error—relative to exact and approximate Bayes factors, and we examine its robustness under distributional misspecification, heteroskedasticity, and dependence. All simulations follow the notation, model conventions, and effective-sample-size definitions established in the original paper, ensuring methodological continuity while providing a substantially more rigorous empirical evaluation.

---

0. Reproducibility & project setup

One-time package bootstrap script located at `scripts/setup_env.R`, e.g.

```
cd Simulation
R -q -e 'source("scripts/setup_env.R")'
```

See raw `.rmd` for setup chunks.

---

1. Definition & Diagnostics

We consider hypothesis tests of the form $H_0 : \theta = \theta_0$ versus $H_1 : \theta \neq \theta_0$.
The **generalized Jeffreys' approximate Bayes factor** (Type I, q-dimensional) is defined as

$$eJAB_{01} \;=\; \sqrt{n}\,\exp\!\left[-\tfrac{1}{2}\,\frac{n^{1/q}-1}{n^{1/q}}\,Q_{\chi^2_q}(1-p)\right],$$

where n is the effective sample size, q the parameter dimension, p the p-value from a test statistic asymptotically following χ^2_q, and $Q_{\chi^2_q}(1-p)$ the $(1-p)$ quantile of the χ^2_q distribution.
The reciprocal quantity is

$$eJAB_{10} \;=\; \frac{1}{eJAB_{01}}.$$

Regularity Conditions

Following *Velidi et al.* (2025), two asymptotic diagnostics underpin the validity of $eJAB_{01}$:

- R_1 **(Null calibration):** under H_0, $p \Rightarrow \mathrm{Unif}(0,1)$.

  - Ensures correct null-model behavior and nominal Type I error control.
- R_2 **(Consistency under H_1):** under H_1, $D_n = -\sqrt{n}\,p\ln p \xrightarrow{P} 0$.
  - This condition checks that p-values shrink toward 0 at the rate assumed in the asymptotic derivation; In the original paper this was implicit.

```r
ejab01 <- function(p, n, q) {
  stopifnot(all(is.finite(p)), all(p > 0 & p < 1), n > 0, q >= 1)
  if (length(q) > 1L && length(p) > 1L)
    stop("Vectorized 'p' with vectorized 'q' not supported")
  term <- qchisq(1 - p, df = q)
  sqrt(n) * exp(-0.5 * ((n^(1/q) - 1)/n^(1/q)) * term)
}

Dn <- function(p, n) {
  stopifnot(all(is.finite(p)), all(p > 0 & p < 1), n > 0)
  -sqrt(n) * p * log(p)
}
```

---

2. Program structure

This notebook has three **sections**. Reproduction is optional and done as a sanity check; rigor happens in Sections II–III.

- **Section I — Baseline replication:** reproduce two figures for a subset of tests to verify code correctness.
- **Section II — Rigorous evaluation:** dense grids over n and effect sizes; compute size, power, bias, RMSE, calibration, and accuracy vs. Bayes factors.
- **Section III — Robustness & ablations:** heavy tails, skew, variance misspecification, dependence violations; component ablations (e.g., using n vs. effective-n variants).

---

3. Designs (tests & effective-n rules)

We include the 11 designs from the paper and extend their grids:

1. **Two-sample t** (equal variance; optional unequal-σ^2 stress).
2. **Simple linear regression** ($Y = \beta_0 + \beta_1 X + \varepsilon$, $\varepsilon \sim N(0,1)$).
3. **Simple logistic regression** ($\operatorname{logit} P(Y=1|X) = \beta_0 + \beta_1 X$).
4. **One-way ANOVA** ($K = 3$ groups, $q = K - 1 = 2$).
5. **Repeated-measures ANOVA** (ICC $\rho \in \{0.2, 0.9\}$; effective $n = \text{subjects} \times (\text{conditions} - 1)$).
6. **Chi-squared independence** (3×3; multinomial & product-multinomial; $q = 4$).
7. **Cox PH** (exponential baseline; effective n = number of uncensored).
8. **Conditional logistic regression** (matched pairs; effective n = pairs).

9. **Wilcoxon signed-rank** (paired; Gaussian & log-normal).

10. **Mann–Whitney** U (two-sample; heavy tails, e.g., t_ν).

11. **Kruskal–Wallis** ($K = 4$, $q = 3$; mean rank shift).

Note. All p-values are obtained from the same tests used in the paper (t/F/Wald/asymptotic nulls). We adhere to the effective-n definitions above in all calculations of $eJAB_{01}$.

---

4. Factor grids

We extend the sample-size ladders beyond those in Velidi et al. (2025) to probe both finite-sample and asymptotic regimes. Effect-size ladders are chosen to mirror conventional benchmarks (Cohen's d, log-odds, hazard ratios) and to allow direct comparison across test families.

```r
# Example override: R -e "options(ejab.reps = 200L); rmarkdown::render('Simulation.Rmd')"
R_reps <- getOption("ejab.reps", 1000L)

# Sample size ladders per design (extendable)
N_t      <- c(20, 50, 100, 200, 500)
N_lm     <- c(50, 100, 200, 500, 1000)
N_glm    <- c(200, 500, 1000, 2000)
N_anova  <- c(60, 120, 240, 480)
N_ranova <- c(30, 60, 120, 240)          # subjects; C=4 within -> eff n = subjects*(C-1)
N_chi    <- c(200, 500, 1500, 3000)      # total count
N_cox    <- c(200, 500, 1000, 2000)      # total subjects (we will record uncensored)
N_clogit <- c(50, 100, 200, 400)         # pairs
N_np     <- c(30, 60, 120, 240)          # generic for Wilcoxon/Mann/KW

# Effect ladders (paper-like small/medium/large + denser)
d_vals        <- c(0, 0.2, 0.4, 0.6, 0.8)   # Cohen's d or mean shift scales
beta1_lm      <- c(0, 0.2, 0.4, 0.6, 0.8)
beta1_glm     <- c(0, 0.4, 0.8, 1.2, 1.6)   # log-odds
anova_delta   <- c(0, 0.3, 0.6, 0.9)        # group mean contrasts
rho_ranova    <- c(0.2, 0.9)
hr_vals       <- c(1.0, 1.2, 1.5, 2.0)      # hazard ratios
chisq_phi     <- c(0, 0.05, 0.1, 0.2)       # generic association strengths
np_shift      <- c(0, 0.2, 0.4, 0.6)        # nonparametric shift
```

---

5. Data generation utilities

We implement simplified, canonical data-generating mechanisms (DGMs) that preserve the core structure and parameterization of the tests in *Velidi et al.* (2025) but avoid unnecessary complexity during baseline tests. Each generator produces p-values under standard parametric assumptions—balanced designs, homoskedastic errors, and independent observations— while enforcing the same (q, n_{eff}) definitions used in the paper.

This deliberate simplification isolates the behaviour of $eJAB_{01}$ itself rather than artifacts of complex data models. Once baseline operating characteristics (size, power, calibration, and agreement with true Bayes factors) are established, we will extend these DGMs with more realistic features (heteroscedasticity, dependence, skew, and heavy tails) in the robustness phase (Section III).

```
Contract: Each function returns a list with at least
`p` (two-sided), `q` (parameter dimension), and `n_eff` (effective sample size used in $eJAB_{01}$).
```

Additional fields are included only for debugging.

```r
## 1) Two-sample t (equal variances; optional Welch control elsewhere)
dgm_t_equalvar <- function(n_per_group, delta, sigma = 1) {
  n1 <- n2 <- as.integer(n_per_group)
  y1 <- rnorm(n1, mean =  delta/2, sd = sigma)
  y2 <- rnorm(n2, mean = -delta/2, sd = sigma)
  tt <- t.test(y1, y2, var.equal = TRUE, alternative = "two.sided")
  list(p = tt$p.value, q = 1L, n_eff = n1 + n2,
       meta = list(stat = unname(tt$statistic), df = unname(tt$parameter)))
}

## 2) Simple linear regression (Gaussian)
dgm_lm <- function(n, beta1, beta0 = 0, sigma = 1) {
  x <- rnorm(n)
  y <- beta0 + beta1 * x + rnorm(n, sd = sigma)
  fit <- stats::lm(y ~ x)
  s <- summary(fit)$coefficients
  p <- s["x", "Pr(>|t|)"]
  list(p = p, q = 1L, n_eff = n,
       meta = list(t_x = s["x", "t value"]))
}

## 3) Simple logistic regression
dgm_glm_logit <- function(n, beta1, beta0 = 0) {
  x <- rnorm(n)
  eta <- beta0 + beta1 * x
  pY <- 1/(1 + exp(-eta))
  y  <- rbinom(n, 1, pY)
  fit <- stats::glm(y ~ x, family = binomial())
  s <- summary(fit)$coefficients
  p <- s["x", "Pr(>|z|)"]
  list(p = p, q = 1L, n_eff = n,
       meta = list(z_x = s["x", "z value"]))
}

## 4) One-way ANOVA (K=3), equal group sizes; means (0, 0, delta)
dgm_anova <- function(n_total, delta) {
  K <- 3L
  n_total <- as.integer(n_total)
  ng <- rep(n_total %/% K, K); ng[seq_len(n_total %% K)] <- ng[seq_len(n_total %% K)] + 1L
  mu <- c(0, 0, delta)
  y  <- unlist(Map(function(m, n) rnorm(n, m, 1), mu, ng))
  g  <- factor(rep(seq_len(K), times = ng))
  fit <- stats::aov(y ~ g)
  p <- summary(fit)[[1]][["Pr(>F)"]][1L]
  list(p = p, q = K - 1L, n_eff = n_total,
       meta = list(F = summary(fit)[[1]][["F value"]][1L]))
}

## 5) Repeated-measures ANOVA (C = 4 within), ICC = rho; eff n = S*(C-1)
##    Subject random intercept: Var(a_s)=rho, residual Var=1-rho   ICC rho
dgm_ranova <- function(n_subjects, rho, delta, C = 4L) {
  S <- as.integer(n_subjects)
  C <- as.integer(C)
  a_s  <- rnorm(S, mean = 0, sd = sqrt(max(rho, 0)))
  eff  <- c(0, 0, 0, delta)              # shift in last condition
  eps_sd <- sqrt(max(1 - rho, 0))
  y <- vector("numeric", S * C)
  subj <- rep(seq_len(S), each = C)
  cond <- rep(seq_len(C), times = S)
  for (s in seq_len(S)) {
    idx <- which(subj == s)
    y[idx] <- a_s[s] + eff + rnorm(C, sd = eps_sd)
  }
  dat <- data.frame(y = y, subj = factor(subj), cond = factor(cond))
  ## Classical RM-ANOVA using Error(strata). Extract within-subject p for cond.
  fit <- stats::aov(y ~ cond + Error(subj/cond), data = dat)
  aov_tabs <- summary(fit)
  ## Within-subject effect is in the 2nd list element, "Within" / "cond"
  p <- tryCatch(aov_tabs[[2]][[1]]["cond", "Pr(>F)"], error = function(e) NA_real_)
  list(p = p, q = C - 1L, n_eff = S * (C - 1L),
       meta = list(S = S, C = C))
}

## 6) Chi-squared independence (3×3)
##    Assoc=0  independence with uniform margins; Assoc>0 increases diagonal mass.
dgm_chisq <- function(N_total, assoc) {
  N_total <- as.integer(N_total)
  assoc <- max(min(assoc, 0.9), 0)  # clamp to [0,0.9)
  P_indep <- matrix(1/9, 3, 3)
  P_diag  <- diag(1/3, 3)
  P <- (1 - assoc) * P_indep + assoc * P_diag
  P <- P / sum(P)
  counts <- as.vector(rmultinom(1, N_total, prob = as.vector(P)))
  tbl <- matrix(counts, 3, 3, byrow = TRUE)
  xt <- suppressWarnings(stats::chisq.test(tbl, correct = FALSE))
  list(p = xt$p.value, q = (3 - 1) * (3 - 1), n_eff = N_total,
       meta = list(stat = unname(xt$statistic)))
}

## 7) Cox proportional hazards - n_eff = number of uncensored (events)
##    Exponential baseline with hazard ratio hr for x=1; ~20-30% censoring.
dgm_cox <- function(N_total, hr) {
  stopifnot(hr > 0)
  N_total <- as.integer(N_total)
  x <- rbinom(N_total, 1, 0.5)
  lambda0 <- 1
  T_event <- rexp(N_total, rate = lambda0 * (hr ^ x))
  C_cens  <- rexp(N_total, rate = 0.2)  # tweak for ~20-30% censoring
  time  <- pmin(T_event, C_cens)
  status <- as.integer(T_event <= C_cens)
  n_events <- sum(status)
  if (n_events < 3L) {
    return(list(p = NA_real_, q = 1L, n_eff = n_events,
                meta = list(warning = "too few events")))
  }
```

```r
  fit <- survival::coxph(survival::Surv(time, status) ~ x)
  p <- summary(fit)$coefficients["x", "Pr(>|z|)"]
  list(p = p, q = 1L, n_eff = n_events,
       meta = list(events = n_events))
}

## 8) Conditional logistic regression (matched pairs) - n_eff = #pairs
##    Simulate strata with exactly 1 case and 1 control per pair.
dgm_clogit <- function(n_pairs, log_odds) {
  S <- as.integer(n_pairs)
  ## Covariate z differs within each pair; probability(case has z=1) = logit^{-1}(log_odds)
  ## Build two rows per pair: one case (y=1) and one control (y=0)
  pr <- 1/(1 + exp(-log_odds))
  ## For each pair, assign z_case   {0,1} with prob pr; z_control = 1 - z_case
  z_case <- rbinom(S, 1, pr)
  z_ctrl <- 1 - z_case
  y  <- c(rep(1L, S), rep(0L, S))
  z  <- c(z_case, z_ctrl)
  stratum <- rep(seq_len(S), times = 2L)
  dat <- data.frame(y = y, z = z, pair = factor(stratum))
  fit <- survival::clogit(y ~ z + strata(pair), data = dat)
  p <- summary(fit)$coefficients["z", "Pr(>|z|)"]
  list(p = p, q = 1L, n_eff = S,
       meta = list(beta_hat = summary(fit)$coefficients["z", "coef"]))
}

## 9) Wilcoxon signed-rank (paired), Normal or Log-Normal shift
dgm_wilcoxon <- function(n, shift, dist = c("normal", "lognormal")) {
  dist <- match.arg(dist)
  if (dist == "normal") {
    base <- rnorm(n)
    y1 <- base + shift
    y2 <- base
  } else {
    base <- rlnorm(n, meanlog = 0, sdlog = 1)
    y1 <- log(base + shift + 1e-8)   # small offset to keep positivity
    y2 <- log(base)
  }
  wt <- suppressWarnings(stats::wilcox.test(y1, y2, paired = TRUE, alternative = "two.sided"))
  list(p = wt$p.value, q = 1L, n_eff = n,
       meta = list(W = unname(wt$statistic)))
}

## 10) Mann-Whitney U (two-sample), heavy tails / normal
dgm_mann <- function(n_per_group, shift,
                     dist1 = c("t3", "normal"),
                     dist2 = c("t3", "normal")) {
  dist1 <- match.arg(dist1); dist2 <- match.arg(dist2)
  n1 <- n2 <- as.integer(n_per_group)
  rgen <- function(n, kind) {
    if (kind == "t3") return(shift + rt(n, df = 3))
    if (kind == "normal") return(shift + rnorm(n))
  }
  x <- rgen(n1, dist1)
  y <- rgen(n2, dist2) - shift              # net location difference = shift
  wt <- suppressWarnings(stats::wilcox.test(x, y, paired = FALSE, alternative = "two.sided"))
  list(p = wt$p.value, q = 1L, n_eff = n1 + n2,
       meta = list(W = unname(wt$statistic)))
}

## 11) Kruskal-Wallis (K=4), mean-rank shift via location shift in one group
dgm_kw <- function(n_total, shift) {
  K <- 4L
  n_total <- as.integer(n_total)
  ng <- rep(n_total %/% K, K); ng[seq_len(n_total %% K)] <- ng[seq_len(n_total %% K)] + 1L
  mu <- c(0, 0, 0, shift)
  y  <- unlist(Map(function(m, n) rnorm(n, m, 1), mu, ng))
  g  <- factor(rep(seq_len(K), times = ng))
  kt <- stats::kruskal.test(y ~ g)
  list(p = kt$p.value, q = K - 1L, n_eff = n_total,
       meta = list(stat = unname(kt$statistic)))
}
```

---

6. Reference Bayes Factors

To assess the calibration of $eJAB_{01}$, we compare its log form $\ln eJAB_{10}$ to reference Bayes factors obtained from fully Bayesian or analytic constructions. We implement three classes:

1. **Parametric (MCMC; Savage–Dickey).**
   Using `rstan` and `logspline`, compute

   $$\mathrm{BF}_{10} = \frac{p(\theta_0)}{p(\theta_0 \mid y)},$$

   by evaluating prior and posterior densities at θ_0 for the focal parameter. This is used sparingly (runtime) for small verification subsets.

2. **Chi-squared (categorical models).**
   (i) A test-statistic Bayes factor (Johnson-style) as a function of the χ^2 statistic and its df;

   (ii) Dirichlet–multinomial Bayes factors via `BayesFactor::contingencyTableBF`.

3. **Nonparametric tests.**
   Closed-form BFs for rank-test z statistics using a working-normal model with a $N(0, g)$ prior for the mean shift (default $g = n_{\text{eff}}$).

Flags for runtime control (toggle expensive references as needed; run-time level flags - the flags in 6.4 are for demo purposes only):

```
RUN_MCMC_BF      <- getOption("ejab.run_mcmc_bf",    TRUE)  # set TRUE to enable MCMC BF
RUN_DIRICHLET_BF <- getOption("ejab.run_dirichlet",  TRUE)   # Dirichlet BF for contingency tables
RUN_BF_NP        <- getOption("ejab.run_np_bf",      TRUE)   # nonparametric references
```

6.1 MCMC Bayes factor (Savage–Dickey)

```
have <- function(pkg) requireNamespace(pkg, quietly = TRUE)
```

```r
if (RUN_MCMC_BF && !(have("rstan") && have("logspline"))) {
  warning("MCMC BF requested but 'rstan' and/or 'logspline' are not available; disabling.")
  RUN_MCMC_BF <- FALSE
}

# Safe logspline density at a point
safe_logspline_density <- function(x, at) {
  fit <- logspline::logspline(x)
  as.numeric(logspline::dlogspline(at, fit))
}

# Generic Savage-Dickey wrapper:
#  - stanmodel: compiled Stan model
#  - data: list for Stan
#  - par_name: scalar parameter name whose value at H0 is theta0
#  - theta0: null value
#  - prior_density: function(theta) giving prior density at theta under the SAME prior as used for eJAB
# Returns: list with ln_BF10 and prior/posterior densities at theta0
compute_bf_mcmc <- function(stanmodel, data, par_name, theta0, prior_density,
                            chains = 4, iter = 2000, warmup = iter %/% 2, seed = 1234) {
  stopifnot(RUN_MCMC_BF)
  fit <- rstan::sampling(stanmodel, data = data, chains = chains, iter = iter,
                         warmup = warmup, seed = seed, refresh = 0)
  dr <- as.data.frame(fit)[[par_name]]
  if (!is.numeric(dr)) stop("Parameter '", par_name, "' not found or not numeric in Stan fit.")
  f_post  <- safe_logspline_density(dr, theta0)
  f_prior <- prior_density(theta0)
  list(
    ln_BF10 = log(f_prior) - log(f_post),
    post_density_at_theta0  = f_post,
    prior_density_at_theta0 = f_prior
  )
}

# Example usage (disabled by default):
# if (RUN_MCMC_BF) {
#   sm <- rstan::stan_model(file = "stan/your_model.stan")
#   data_stan <- list(...)
#   prior_density <- function(theta) dnorm(theta, mean = 0, sd = 1)
#   out <- compute_bf_mcmc(sm, data_stan, par_name = "beta1", theta0 = 0, prior_density = prior_density
#   cat(sprintf("MCMC BF (Savage-Dickey): ln BF10 = %.3f\n", out$ln_BF10))
# }
```

6.2 Chi-squared references (test-statistic & Dirichlet)

```r
if (RUN_DIRICHLET_BF && !have("BayesFactor")) {
  warning("Dirichlet BF requested but 'BayesFactor' is not available; disabling.")
  RUN_DIRICHLET_BF <- FALSE
}

# (A) Johnson-style test-statistic BF for chi-square:
# For chi-square statistic T with df=q:
#   ln BF10 = T/2 - (q/2) * log(1 + T/q)
```

```r
compute_bf_chisq_ts <- function(chisq_stat, df) {
  stopifnot(chisq_stat >= 0, df >= 1)
  as.numeric( (chisq_stat / 2) - (df / 2) * log1p(chisq_stat / df) )
}

# (B) Dirichlet-multinomial BF via BayesFactor::contingencyTableBF
compute_bf_chisq_dirichlet <- function(tbl, prior_conc = 1,
                                       sampleType = c("jointMulti","indepMulti"),
                                       fixedMargin = c("rows","cols","none")) {
  stopifnot(RUN_DIRICHLET_BF)
  sampleType  <- match.arg(sampleType)
  fixedMargin <- match.arg(fixedMargin)
  bf <- BayesFactor::contingencyTableBF(as.table(tbl),
                                        sampleType = sampleType,
                                        fixedMargin = fixedMargin,
                                        priorConcentration = prior_conc)
  as.numeric(log(BayesFactor::extractBF(bf, onlybf = TRUE)))
}

# Helper to compute chi-square test pieces from a table
chisq_test_tbl <- function(tbl) {
  tst <- suppressWarnings(chisq.test(tbl, correct = FALSE))
  list(stat = unname(tst$statistic), df = unname(tst$parameter), p = unname(tst$p.value))
}
```

6.3 Nonparametric references (rank-test z working model)

We map a two-sided rank-test p (Wilcoxon/Mann–Whitney/Kruskal–Wallis) to $Z = \Phi^{-1}(1 - p/2)$ and use a working-normal model with prior $\delta \sim N(0, g)$, yielding

$$\ln \mathrm{BF}_{10} \;=\; -\tfrac{1}{2}\ln(1+g) + \frac{Z^2 g}{2(1+g)}.$$

We default to $g = n_{\mathrm{eff}}$ (unit-information style), but allow overrides.

```r
compute_bf_np_from_p <- function(p_two_sided, n_eff, g = n_eff) {
  stopifnot(p_two_sided > 0, p_two_sided < 1, n_eff > 0, g > 0)
  Z <- qnorm(1 - p_two_sided / 2)   # |Z| from two-sided p
  as.numeric( -0.5 * log1p(g) + (Z * Z) * g / (2 * (1 + g)) )
}
```

6.4 Demonstration

Set flags to TRUE in raw `.rmd` for demo.

Table 1: Summary of log- and standard Bayes factors from Section 6.4 demonstration.

Test	ln_BF10	BF10	Evidence
Chi-squared (TS)	1.318	3.736	Moderate H1
Chi-squared (Dirichlet)	-2.900	0.055	Supports H0
Nonparametric (Wilcoxon-like)	0.979	2.662	Anecdotal H1
MCMC (Savage–Dickey)	2.252	9.507	Moderate H1

7. Section I — Baseline replication (Incomplete)

Replicate two reference panels to confirm alignment.

```r
# Example: t-test at selected n/effects; scatter ln eJAB10 vs ln BF10
# results_s1 <- ...
# ggplot(results_s1, aes(x = ln_ejab10, y = ln_bf10_ref, colour = p)) +
#   geom_point(alpha=.6) + geom_abline(slope=1, intercept=0, linetype="dashed") +
#   facet_wrap(~ panel, scales="free") + theme_bw()
```

---

8. Section II — Rigorous evaluation (Incomplete)

8.1 Grid definition

```r
grid_s2 <- tibble::tibble(
  design = c(
    rep("t",      length(N_t)      * length(d_vals)),
    rep("lm",     length(N_lm)     * length(beta1_lm)),
    rep("glm",    length(N_glm)    * length(beta1_glm)),
    rep("anova",  length(N_anova)  * length(anova_delta)),
    rep("ranova", length(N_ranova) * length(anova_delta) * length(rho_ranova)),
    rep("chisq",  length(N_chi)    * length(chisq_phi)),
    rep("cox",    length(N_cox)    * length(hr_vals)),
    rep("clogit", length(N_clogit) * length(d_vals)),
    rep("wilcox", length(N_np)     * length(np_shift)),
    rep("mann",   length(N_np)     * length(np_shift)),
    rep("kw",     length(N_np)     * length(np_shift))
  )
) %>% dplyr::mutate(row_id = dplyr::row_number())

# quick peek
head(grid_s2)
#> # A tibble: 6 x 2
#>   design row_id
#>   <chr>   <int>
#> 1 t           1
#> 2 t           2
#> 3 t           3
#> 4 t           4
#> 5 t           5
#> 6 t           6
```

8.2 Metrics

For each (design, n, effect):

- **Size** (under H_0): $\hat{\alpha} = \Pr(eJAB_{01} > 1)$, MCSE via binomial formula.
- **Power** (under H_1): $\Pr(eJAB_{01} < 1)$ across effect ladder.
- **Bias / RMSE** of $\ln eJAB_{10}$ vs. reference $\ln BF_{10}$.
- **Calibration**: $\ln eJAB_{10}$ vs. $\ln BF_{10}$ scatter + smooth; slope near 1 desirable.

- R_2 **diagnostic**: distribution of D_n contracting to 0 with n.

Skeleton aggregation:

```
# results_s2 <- purrr::map_dfr(1:nrow(grid_s2), function(i) { ... })
# summary_s2 <- results_s2 %>%
#   dplyr::group_by(design, n, effect) %>%
#   dplyr::summarise(
#     size_hat  = mean(ejab01 > 1),
#     size_mcse = mcse_prop(size_hat, R_reps),
#     power_hat = mean(ejab01 < 1),
#     power_mcse= mcse_prop(power_hat, R_reps),
#     bias_ln   = mean(ln_ejab10 - ln_bf10_ref, na.rm=TRUE),
#     rmse_ln   = sqrt(mean((ln_ejab10 - ln_bf10_ref)^2, na.rm=TRUE)),
#     Dn_mean   = mean(Dn, na.rm=TRUE),
#     .groups = "drop"
#   )
```

8.3 Figures

- **Size/power curves** vs n for each design.

- $\ln eJAB_{10}$ **vs** $\ln BF_{10}$ (per design, facets).

- D_n **vs** n (expect contraction to 0).

```
# ggplot(summary_s2 %>% dplyr::filter(effect==0), aes(n, size_hat)) + ...
# ggplot(summary_s2 %>% dplyr::filter(effect>0), aes(n, power_hat, colour=as.factor(effect))) + ...
# ggplot(results_s2, aes(ln_ejab10, ln_bf10_ref, colour=p)) + ...
# ggplot(results_s2 %>% dplyr::filter(effect>0), aes(n_eff, Dn)) + ...
```

---

9. Section III — Robustness & ablations (Incomplete)

We probe violations and components:

- **Heavy tails & skew:** replace Gaussian errors with t_ν ($\nu \in \{3, 5\}$), log-normal, skew-normal; rerun size/power.

- **Variance misspecification:** two-sample t with unequal σ^2 (use Welch test p-values accordingly; check impact on $eJAB_{01}$ classification).

- **Dependence:** rANOVA with mis-specified ρ at analysis; quantify degradation.

- **Outliers:** contamination mixtures (e.g., $0.95 \cdot N(0, 1) + 0.05 \cdot N(0, 10^2)$).

- **Ablations:** swap effective-n with naive n; compare BIC approximation vs $eJAB_{01}$ for $q > 1$.

```
# results_s3 <- run_robustness_scenarios(...)
# figs_s3 <- plot_robustness_panels(results_s3)
```

---

10. Tables & reporting (Incomplete)

For each design and grid point, report **estimate ± MCSE**:

- Size and power (α, β)
- Bias and RMSE (vs reference BF)
- Proportion classified into Jeffreys evidence categories by $eJAB_{10}$ (optional)

```
# knitr::kable(summary_s2 %>% dplyr::arrange(design, n, effect), digits=3)
```

---

11. Practical guidance (to be filled after results) (Incomplete)

- Regions (in n–effect space) where $eJAB_{01}$ is conservative/liberal vs BF.
- Effect sizes where R_2 contraction becomes practically visible.
- Recommendations for effective-n choices in survival / matched designs.
- When to prefer $eJAB_{01}$ vs BIC approximation (especially $q > 1$).

---

12. Session info

```
sessionInfo()
#> R version 4.5.1 (2025-06-13)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Arch Linux
#>
#> Matrix products: default
#> BLAS:   /usr/lib/libblas.so.3.12.0
#> LAPACK: /usr/lib/liblapack.so.3.12.0  LAPACK version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: Canada/Pacific
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] splines   stats4    stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] rstan_2.32.7           StanHeaders_2.32.10    logspline_2.1.22
#>  [4] BayesFactor_0.9.12-4.7 Matrix_1.7-3           coda_0.19-4.1
#>  [7] survival_3.8-3         VGAM_1.1-13            MASS_7.3-65
#> [10] cowplot_1.2.0          patchwork_1.3.2        ggplot2_4.0.0
#> [13] glue_1.8.0             stringr_1.5.2          broom_1.0.10
#> [16] data.table_1.17.8      purrr_1.1.0            tibble_3.3.0
#> [19] magrittr_2.0.4        dplyr_1.1.4
#>
#> loaded via a namespace (and not attached):
#>  [1] utf8_1.2.6           generics_0.1.4       tidyr_1.3.1
#>  [4] renv_1.1.4           stringi_1.8.7        lattice_0.22-7
#>  [7] digest_0.6.37        evaluate_1.0.5       grid_4.5.1
#> [10] RColorBrewer_1.1-3   mvtnorm_1.3-3        fastmap_1.2.0
#> [13] pkgbuild_1.4.8       backports_1.5.0      gridExtra_2.3
#> [16] QuickJSR_1.8.1       scales_1.4.0         pbapply_1.7-4
#> [19] codetools_0.2-20     cli_3.6.5            rlang_1.1.6
#> [22] withr_3.0.2          yaml_2.3.10          inline_0.3.21
#> [25] tools_4.5.1          parallel_4.5.1       MatrixModels_0.5-4
#> [28] vctrs_0.6.5          R6_2.6.1             matrixStats_1.5.0
#> [31] lifecycle_1.0.4      pkgconfig_2.0.3      RcppParallel_5.1.11-1
#> [34] pillar_1.11.1        gtable_0.3.6         loo_2.8.0
#> [37] Rcpp_1.1.0           xfun_0.53            tidyselect_1.2.1
#> [40] knitr_1.50           farver_2.1.2         htmltools_0.5.8.1
#> [43] rmarkdown_2.29       compiler_4.5.1       S7_0.2.0
```