Bootstrapping

Nhi-Ha Le

#Full Code and Notes

```
# NOTES:
# Requires CRAN packages: Rcapture, dga, LCMCR, MCMCpack, boot, MASS
# Some routines (Madigan-York) are limited to <= 5 lists; guard against that.

## 0. Packages/install ---------------------------------------------------------
pkgs <- c("Rcapture","dga","LCMCR","MCMCpack","boot","MASS")
for(p in pkgs) if(!requireNamespace(p, quietly=TRUE)) install.packages(p)
library(MASS)
library(Rcapture)
library(dga)
library(LCMCR)
library(MCMCpack)
library(boot)

## 1. Input contingency table (your UK six-list table) -------------------------------
## Silverman's data is not publicly avaliable or easily accessible. I base the replication o
ctab_raw <- data.frame(
  LA = c(
    # singles
    1,0,0,0,0,0,
    # middle block (13 rows)
    1,1,1,0,0,0,0,0,0,0,0,0,0,
    # 5 triples
    1,1,1,0,0,
    # quadruple
    1
  ),
  NG = c(
    # singles
    0,1,0,0,0,0,
```

```
  # middle block
  0,0,0,1,1,1,1,0,0,0,0,0,0,
  # triples
  1,1,0,1,0,
  # quadruple
  1
),
PF = c(
  # singles
  0,0,1,0,0,0,
  # middle block
  1,0,0,1,0,0,0,1,1,1,0,0,0,
  # triples
  1,0,1,0,0,
  # quadruple
  1
),
GO = c(
  # singles
  0,0,0,1,0,0,
  # middle block
  0,1,0,0,1,0,0,1,0,0,1,1,0,
  # triples
  0,1,0,1,0,
  # quadruple
  1
),
GP = c(
  # singles
  0,0,0,0,1,0,
  # middle block
  0,0,1,0,0,1,0,0,1,0,1,0,1,
  # triples
  0,0,1,0,1,
  # quadruple
  1
),
NCA = c(
  # singles
  0,0,0,0,0,1,
  # middle block
  0,0,0,0,0,0,1,0,0,1,0,1,1,
```

```
    # triples
    0,0,0,1,1,
    # quadruple
    1
  ),
  Count = c(
    # singles
    54,463,907,695,316,57,
    # middle block counts (13)
    15,19,3,56,19,1,3,69,10,31,8,6,1,
    # triples (5)
    1,1,4,3,1,
    # quadruple (1)
    1
  )
)
# Standardize column name to freq (script expects 'freq')
colnames(ctab_raw)[ncol(ctab_raw)] <- "freq"
# Ensure columns are numeric
ctab_raw[] <- lapply(ctab_raw, function(x) as.numeric(as.character(x)))

# Quick sanity check
cat("Number of observable rows:", nrow(ctab_raw), "\n")
```

```
Number of observable rows: 25
```

```
cat("Total observed:", sum(ctab_raw$freq), "\n\n")
```

```
Total observed: 2744
```

```
## 2. Utility helpers ----------------------------------------------------------
# Expand contingency rows to individual-level matrix (rows = individuals, cols = lists)
expand_table_to_individuals <- function(tab) {
  if(!"freq" %in% colnames(tab)) stop("tab must have 'freq' column")
  K <- ncol(tab) - 1
  rows_list <- lapply(1:nrow(tab), function(i) {
    rfreq <- tab$freq[i]
    if(rfreq <= 0) return(NULL)
    vec <- as.numeric(tab[i, 1:K])
    mat <- matrix(rep(vec, each=rfreq), nrow=rfreq, byrow=TRUE)
    as.data.frame(mat)
```

```
  })
  rows <- do.call(rbind, rows_list)
  if(is.null(rows)) return(NULL)
  colnames(rows) <- colnames(tab)[1:(ncol(tab)-1)]
  # ensure numeric 0/1
  rows[] <- lapply(rows, function(x) as.numeric(as.character(x)))
  return(rows)
}

# Collapse individual-level back to contingency table (same column names + freq)
individuals_to_ctable <- function(indiv_df, template_cols=NULL) {
  if(is.null(indiv_df) || nrow(indiv_df)==0) return(NULL)
  pat <- apply(indiv_df, 1, paste, collapse="")
  tbl <- as.data.frame(table(pat), stringsAsFactors=FALSE)
  S <- nchar(tbl$pat[1])
  mat <- do.call(rbind, strsplit(tbl$pat, ""))
  mat <- apply(mat, 2, as.numeric)
  df <- as.data.frame(mat)
  if(!is.null(template_cols)) {
    # set names consistent with template
    colnames(df) <- template_cols
  } else {
    colnames(df) <- paste0("L",1:ncol(df))
  }
  df$freq <- as.numeric(tbl$Freq)
  return(df)
}

## 3. Estimators ---------------------------------------------------------------

# A: main-effects Poisson GLM (fast & stable)
est_main_effects <- function(ctab) {
  # ctab: first K columns are lists (0/1), last col named freq
  if(!"freq" %in% colnames(ctab)) stop("ctab must contain 'freq'")
  K <- ncol(ctab) - 1
  listvars <- colnames(ctab)[1:K]
  f <- as.formula(paste("freq ~", paste(listvars, collapse = " + ")))
  fit <- glm(f, family=poisson(), data=ctab)
  mu_hat <- coef(fit)["(Intercept)"]
  dark <- exp(mu_hat)
  observed <- sum(ctab$freq)
  total_est <- observed + dark
```

```r
  return(list(fit=fit, dark=dark, total=total_est))
}

# B: stepwise AIC using stepAIC (from MASS) - stable and reproducible
est_stepwise_AIC <- function(ctab, direction="both", trace=FALSE) {
  K <- ncol(ctab) - 1
  listvars <- colnames(ctab)[1:K]
  base_f <- as.formula(paste("freq ~", paste(listvars, collapse = " + ")))
  # Full scope = all pairwise interactions
  if(K >= 2) {
    interaction_terms <- c()
    for(i in 1:(K-1)) for(j in (i+1):K) interaction_terms <- c(interaction_terms, paste0(list
    scope_formula <- as.formula(paste("~ . +", paste(interaction_terms, collapse=" + ")))
  } else {
    scope_formula <- base_f
  }
  start_fit <- glm(base_f, family=poisson(), data=ctab)
  # Use stepAIC to search
  step_fit <- tryCatch({
    stepAIC(start_fit, scope=scope_formula, direction=direction, trace=trace)
  }, error=function(e){ warning("stepAIC failed: ", e$message); return(NULL) })
  if(is.null(step_fit)) return(NULL)
  mu_hat <- coef(step_fit)["(Intercept)"]
  dark <- exp(mu_hat)
  observed <- sum(ctab$freq)
  return(list(fit=step_fit, dark=dark, total=observed + dark))
}

# C: Madigan-York BMA via dga - only implemented for <=5 lists
est_madigan_york <- function(ctab, maxN=NULL) {
  K <- ncol(ctab) - 1
  if(K > 5) {
    warning("Madigan-York (dga::bma.cr) is only for <=5 lists; skipping.")
    return(NULL)
  }
  patterns <- apply(ctab[,1:K], 1, paste, collapse="")
  counts <- setNames(ctab$freq, patterns)
  if(is.null(maxN)) maxN <- 10 * sum(ctab$freq)
  res <- tryCatch({
    bma.cr(counts, N.range = 0:maxN)
  }, error = function(e) { warning("bma.cr failed: ", e$message); return(NULL) })
  return(res)
```

```r
}

# D: Dirichlet-process latent class via LCMCR (wrap safely)
est_lcmcr <- function(ctab, its=20000, burnin=5000, thin=10, seed=12345) {
  K <- ncol(ctab) - 1
  hist <- as.matrix(ctab[,1:(ncol(ctab)-1)])
  hist <- cbind(hist, ct=ctab$freq)
  set.seed(seed)
  res <- tryCatch({
    lcmcr.fit(hist, its=its, burnin=burnin, thin=thin)
  }, error=function(e){ warning("LCMCR failed: ", e$message); return(NULL) })
  return(res)
}

# E: Silverman Bayesian-thresholding using MCMCpoisson (robustified)
est_bayesian_threshold <- function(ctab, prior_var=1, thresh=2,
                                   niter=5000, burnin=1000, thin=2, seed=12345) {

  if(!"freq" %in% colnames(ctab)) stop("ctab must have 'freq' column")
  # drop any rows with zero freq (not informative for MCMC)
  ctab2 <- ctab[ctab$freq > 0, , drop=FALSE]
  K <- ncol(ctab2) - 1
  listvars <- colnames(ctab2)[1:K]

  # identify interactions that are possible (i.e., observed overlap exists)
  possible_interactions <- c()
  for(i in 1:(K-1)) for(j in (i+1):K) {
    if(sum(ctab2[[listvars[i]]] & ctab2[[listvars[j]]]) > 0) {
      possible_interactions <- c(possible_interactions, paste0(listvars[i], ":", listvars[j]))
    }
  }

  # build full formula (main + possible interactions)
  all_terms <- c(listvars, possible_interactions)
  if(length(all_terms)==0) stop("No predictors found")
  full_formula <- as.formula(paste("freq ~", paste(all_terms, collapse=" + ")))

  # design & remove zero-variance columns (constant columns)
  design <- model.matrix(full_formula, data=ctab2)
  keep_cols <- apply(design, 2, function(col) length(unique(col)) > 1)
  design <- design[, keep_cols, drop=FALSE]
  # if only intercept remains, set a formula accordingly
```

```
if(ncol(design)==1) {
  final_formula <- as.formula("freq ~ 1")
} else {
  final_terms <- colnames(design)[-1]  # drop intercept
  final_formula <- as.formula(paste("freq ~", paste(final_terms, collapse=" + ")))
}

# build prior mean/precision matching number of columns in design
p <- ncol(design)
b0 <- rep(0, p)
# stable default: small precision (large variance) for intercept/main effects,
# moderate precision for interaction columns
B0 <- diag(1e-3, p)   # default weak precision
int_idx <- grep(":", colnames(design))
if(length(int_idx) > 0) diag(B0)[int_idx] <- 1 / prior_var

# run MCMC with tryCatch to capture errors
set.seed(seed)
mcmc_out <- tryCatch({
  MCMCpoisson(final_formula, data=ctab2, b0=b0, B0=B0, mcmc=niter, burnin=burnin, thin=thi
}, error=function(e){
  stop("MCMCpoisson failed even after robustification: ", e$message)
})

# posterior summaries and thresholding
post_mean <- colMeans(mcmc_out)
post_sd   <- apply(mcmc_out, 2, sd)
int_idx2 <- grep(":", colnames(mcmc_out))
kept <- character(0)
if(length(int_idx2)>0) {
  zratios <- abs(post_mean[int_idx2] / post_sd[int_idx2])
  kept <- names(zratios)[which(zratios > thresh)]
}

intercept_draws <- mcmc_out[, "(Intercept)"]
dark_draws <- exp(intercept_draws)
observed <- sum(ctab$freq)

return(list(
  mcmc = mcmc_out,
  kept_interactions = kept,
  dark_mean = mean(dark_draws),
```

```
    dark_median = median(dark_draws),
    dark_ci = quantile(dark_draws, c(0.025, 0.5, 0.975)),
    total_mean = observed + mean(dark_draws),
    total_ci = observed + quantile(dark_draws, c(0.025, 0.5, 0.975))
  ))
}

## 4. Bootstrap pipeline (nonparametric individual-level) ---------------------------------
# We do bootstrap for main-effects by default ( for bayes, we can call with small B to test.)

bootstrap_pipeline <- function(ctab, B=200, estimator=c("main","bayes_thresh"), bayes_args=li
  estimator <- match.arg(estimator)
  indiv <- expand_table_to_individuals(ctab)
  if(is.null(indiv) || nrow(indiv)==0) stop("No individual-level rows generated from continge
  set.seed(seed)

  # statistic function for boot
  if(estimator=="main") {
    stat_fn <- function(data, indices) {
      sampled <- data[indices, , drop=FALSE]
      ctab_rep <- individuals_to_ctable(sampled, template_cols=colnames(ctab)[1:(ncol(ctab)-1
      # ensure col order
      ctab_rep <- ctab_rep[, c(colnames(ctab)[1:(ncol(ctab)-1)], "freq")]
      res <- tryCatch(est_main_effects(ctab_rep)$total, error=function(e) NA)
      return(as.numeric(res))
    }
    boot_out <- boot::boot(indiv, statistic = stat_fn, R = B)
    return(boot_out)
  } else {
    # bayes_thresh bootstrap is expensive - each replicate runs MCMC
    stat_fn <- function(data, indices) {
      sampled <- data[indices, , drop=FALSE]
      ctab_rep <- individuals_to_ctable(sampled, template_cols=colnames(ctab)[1:(ncol(ctab)-1
      ctab_rep <- ctab_rep[, c(colnames(ctab)[1:(ncol(ctab)-1)], "freq")]
      res <- tryCatch({
        out <- do.call(est_bayesian_threshold, c(list(ctab=ctab_rep), bayes_args))
        out$total_mean
      }, error=function(e) NA)
      return(as.numeric(res))
    }
    boot_out <- boot::boot(indiv, statistic = stat_fn, R = B)
    return(boot_out)
```

```
  }
}

## 5. Run analyses on the UK table and produce results ----------------------------------
ctab <- ctab_raw  # standardized name used throughout

# Print basic counts
cat("Observed total cases (sum of frequencies):", sum(ctab$freq), "\n")
```

```
Observed total cases (sum of frequencies): 2744
```

```
cat("Number of observed patterns (rows):", nrow(ctab), "\n\n")
```

```
Number of observed patterns (rows): 25
```

```
# 5.1 Main-effects GLM
cat("Running main-effects GLM ...\n")
```

```
Running main-effects GLM ...
```

```
main_res <- tryCatch(est_main_effects(ctab), error=function(e) { warning(e$message); NULL })
if(!is.null(main_res)) {
  cat("Main-effects estimate (dark figure):", round(main_res$dark,1), "\n")
  cat("Main-effects estimate (total):", round(main_res$total,1), "\n\n")
}
```

```
Main-effects estimate (dark figure): 8596.6
Main-effects estimate (total): 11340.6
```

```
# 5.2 Stepwise AIC (MASS::stepAIC)
cat("Running stepwise AIC (pairwise interactions allowed) ...\n")
```

```
Running stepwise AIC (pairwise interactions allowed) ...
```

```r
step_res <- tryCatch(est_stepwise_AIC(ctab, direction="both", trace=FALSE), error=function(e)
if(!is.null(step_res)) {
  cat("Stepwise-AIC estimate (dark):", round(step_res$dark,1), "\n")
  cat("Stepwise-AIC estimate (total):", round(step_res$total,1), "\n")
  cat("Selected model coefficients:\n")
  print(coef(step_res$fit))
  cat("\n")
}
```

```
Stepwise-AIC estimate (dark): 145986.6
Stepwise-AIC estimate (total): 148730.6
Selected model coefficients:
(Intercept)          LA          NG          PF          GO          GP
  11.891270   -7.900751   -5.753722   -5.081117   -5.348126   -6.135272
         NCA       PF:NCA       LA:GO       LA:PF       LA:NCA       GO:GP
  -7.849635    4.473469    4.325600    3.798001   -8.893486    1.671570
       PF:GP       LA:GP       NG:GO       GO:NCA       NG:NCA       NG:PF
   1.619373    3.215320    2.182711    3.111609    2.837241    2.970228
       PF:GO       GP:NCA
   2.772079    2.093638
```

```r
# 5.3 Madigan-York (only if K <= 5)
cat("Madigan-York (dga) attempt (only for <=5 lists) ...\n")
```

```
Madigan-York (dga) attempt (only for <=5 lists) ...
```

```r
my_res <- tryCatch(est_madigan_york(ctab), error=function(e){ warning(e$message); NULL })
if(is.null(my_res)) cat("Madigan-York not run (K>5 or error).\n\n") else cat("Madigan-York ra
```

```
Madigan-York not run (K>5 or error).
```

```r
# 5.4 LCMCR (Dirichlet-process latent class)
cat("Running LCMCR (may be slow) ...\n")
```

```
Running LCMCR (may be slow) ...
```

```r
lcm_res <- tryCatch(est_lcmcr(ctab, its=20000, burnin=5000, thin=10), error=function(e){ war
if(!is.null(lcm_res)) {
  cat("LCMCR result object present. Inspect lcm_res for posterior draws and diagnostics.\n\n
} else {
  cat("LCMCR not available or failed - skipping detailed output.\n\n")
}
```

```
LCMCR not available or failed - skipping detailed output.
```

```r
# 5.5 Silverman Bayesian-threshold method (MCMC) - modest MCMC to be feasible
cat("Running Bayesian-threshold MCMC (modest iterations) ...\n")
```

```
Running Bayesian-threshold MCMC (modest iterations) ...
```

```r
bt_res <- tryCatch(est_bayesian_threshold(ctab, prior_var=1, thresh=2, niter=6000, burnin=150
                   error=function(e){ warning(e$message); NULL })
if(!is.null(bt_res)) {
  cat("Bayes-threshold total mean:", round(bt_res$total_mean,1), "\n")
  cat("Bayes-threshold total 95% CI:", round(bt_res$total_ci[1],1), "to", round(bt_res$total_
  cat("Kept interactions:", if(length(bt_res$kept_interactions)==0) "None" else paste(bt_res$
}
```

```
Bayes-threshold total mean: 2849.9
Bayes-threshold total 95% CI: 2836.7 to 2860.1
Kept interactions: PF:LA, GO:LA, GP:LA, NCA:LA, NG:PF, NG:GO, NG:GP, NG:NCA, PF:GO, PF:GP, PI
```

```r
## 6. Bootstrap results (fast) ----------------------------------------------------
cat("Running bootstrap for main-effects (B=200; fast demo)...\n")
```

```
Running bootstrap for main-effects (B=200; fast demo)...
```

```r
set.seed(2025)
boot_main <- tryCatch(bootstrap_pipeline(ctab, B=200, estimator="main"), error=function(e){ w
if(!is.null(boot_main)) {
  print(boot_main)
  ci_main <- tryCatch(boot.ci(boot_main, type="bca"), error=function(e) NULL)
  if(!is.null(ci_main)) {
    cat("Main-effects bootstrap BCa 95% CI for total:\n")
    print(ci_main$bca)
  }
} else cat("Bootstrap (main) failed.\n")
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot::boot(data = indiv, statistic = stat_fn, R = B)


Bootstrap Statistics :
    original   bias    std. error
t1* 4358.604 101.7273    72.22375

Main-effects bootstrap BCa 95% CI for total:
     conf
[1,] 0.95 0 73.56 4269.306 4453.364
```

```r
# Bootstrap Bayesian-threshold (WARNING: slow because it reruns MCMC each replicate)
do_bayes_bootstrap <- FALSE
if(do_bayes_bootstrap) {
  cat("Running bootstrap for Bayes-threshold (B=100; WARNING: slow) ...\n")
  set.seed(2026)
  boot_bayes <- tryCatch(bootstrap_pipeline(ctab, B=100, estimator="bayes_thresh",
                                            bayes_args=list(prior_var=1, thresh=2, niter=2000
                         error=function(e) { warning(e$message); NULL })
  if(!is.null(boot_bayes)) {
    ci_bayes <- tryCatch(boot.ci(boot_bayes, type="bca"), error=function(e) NULL)
    if(!is.null(ci_bayes)) {
      cat("Bayes-threshold bootstrap BCa 95% CI for total (approx):\n"); print(ci_bayes$bca)
    }
  }
}

## 7. Results & interpretation ---------------------------------------
cat("\n======================== Summary of results ========================\n")
```

```
======================== Summary of results ========================
```

```r
obs <- sum(ctab$freq)
cat("Observed cases (sum of freq):", obs, "\n\n")
```

```
Observed cases (sum of freq): 2744
```

```
if(!is.null(main_res)) {
  cat("Main-effects model:\n")
  cat("  Dark figure estimate:", round(main_res$dark,1), "\n")
  cat("  Total estimate:", round(main_res$total,1), "\n")
  cat("  Interpretation: assumes independence between lists (no interactions).\n\n")
}
```

```
Main-effects model:
  Dark figure estimate: 8596.6
  Total estimate: 11340.6
  Interpretation: assumes independence between lists (no interactions).
```

```
if(!is.null(step_res)) {
  cat("Stepwise AIC model:\n")
  cat("  Total estimate:", round(step_res$total,1), "\n")
  cat("  Interpretation: model chosen by AIC among pairwise-interaction models; CI conditiona
}
```

```
Stepwise AIC model:
  Total estimate: 148730.6
  Interpretation: model chosen by AIC among pairwise-interaction models; CI conditional on cl
```

```
if(!is.null(bt_res)) {
  cat("Bayesian-threshold (Silverman) model:\n")
  cat("  Total mean:", round(bt_res$total_mean,1), "\n")
  cat("  95% credible interval:", round(bt_res$total_ci[1],1), "to", round(bt_res$total_ci[3]
  cat("  Kept interactions:", if(length(bt_res$kept_interactions)==0) "None" else paste(bt_re
  cat("  Interpretation: posterior-based, shrinks weak interactions, thresholded to avoid ove
}
```

```
Bayesian-threshold (Silverman) model:
  Total mean: 2849.9
  95% credible interval: 2836.7 to 2860.1
  Kept interactions: PF:LA, GO:LA, GP:LA, NCA:LA, NG:PF, NG:GO, NG:GP, NG:NCA, PF:GO, PF:GP,
  Interpretation: posterior-based, shrinks weak interactions, thresholded to avoid overfittin
```

```r
if(!is.null(boot_main)) {
  cat("Bootstrap (main-effects) summary:\n")
  cat("  Bootstrap mean (total):", round(mean(boot_main$t, na.rm=TRUE),1), "\n")
  cat("  Bootstrap std. error:", round(sd(boot_main$t, na.rm=TRUE),1), "\n")
  cat("  BCa 95% CI (if computed above): see boot.ci output above.\n\n")
}
```

```
Bootstrap (main-effects) summary:
  Bootstrap mean (total): 4460.3
  Bootstrap std. error: 72.2
  BCa 95% CI (if computed above): see boot.ci output above.
```

```r
cat("=====================================================================\n")
```

```
=====================================================================
```

```r
cat("Notes:\n - 'Total' = observed + estimated dark figure.\n - Stepwise/AIC CIs are conditi
```

```
Notes:
 - 'Total' = observed + estimated dark figure.
 - Stepwise/AIC CIs are conditional on the selected model and may under-
represent model-selection uncertainty.
 - Bootstrapping by re-fitting the model inside each replicate (especially MCMC models) is t
selection uncertainty but is computationally expensive.
```