sim_dmacs_p=6

```r
library(SimDesign)
```

```
Warning: package 'SimDesign' was built under R version 4.3.3
```

```r
library(lavaan)
```

```
Warning: package 'lavaan' was built under R version 4.3.3
```

```
This is lavaan 0.6-18
lavaan is FREE software! Please report any bugs.
```

```r
library(parallel)
library(pinsearch)

# TODO:
#   - Increase the number of replications to 500
#   - Summarize the pattern of bias

# Define conditions
design <- createDesign(
  n = c(30, 100, 250, 1000)
)

# Fixed objects
set.seed(1855)
# Helper
get_ucov <- function(p, scale = sqrt(0.1), n = 5) {
  W <- matrix(rnorm(p * n), nrow = n)
```

```r
  WtW <- crossprod(W)
  D <- diag(1 / sqrt(diag(WtW))) * scale
  D %*% WtW %*% D
}
fixed <- list(
  p = 6,
  lambda = c(.3, .7, .4, .5, .6, .4),
  dlambda = list(
    c(0, 0, 0, 0, 0, 0),
    c(.3, 0, 0, 0, 0, 0)
  ),
  nu = c(2, 3, 1.5, 3.5, 2, 3),
  alpha = c(0, -0.25),
  psi = c(1, 1.15),
  theta = c(1, 1.2, .8, .9, 1, 1) - .1,
  dtheta = matrix(
    runif(12, min = -0.2, max = 0.2),
    nrow = 2
  ),
  # ucov = replicate(4, get_ucov(6), simplify = FALSE)
  ucov = replicate(2, diag(.1, 6), simplify = FALSE),
  ninv_ind = c(1)
)
# lavaan syntax
fixed$mod <- paste(
  "f =~",
  paste0("y", seq_len(fixed$p), collapse = " + ")
)
# Compute implied means and covariances
fixed <- within(fixed, {
  lambdag <- lapply(dlambda, FUN = \(x) x + lambda)
  Thetag <- lapply(seq_along(ucov),
                   FUN = function(g) {
                     diag(theta + dtheta[g, ]) + ucov[[g]]
                   })
  covy <- mapply(\(lam, psi, th) tcrossprod(lam) * psi + th,
                 lam = lambdag, psi = psi, th = Thetag,
                 SIMPLIFY = FALSE)
  meany <- mapply(\(lam, al, nu) nu + lam * al,
                  lam = lambdag, al = alpha, nu = list(nu),
                  SIMPLIFY = FALSE)
```

```r
})

# Population effect size
fixed$dmacs_pop <- local({
  pooled_sd <- lapply(fixed$covy, FUN = \(x) diag(x)) |>
    do.call(what = rbind) |>
    colMeans() |>
    sqrt()

  dmacs(
    intercepts = matrix(rep(fixed$nu, 2),
                        nrow = 2,
                        byrow = TRUE
    ),
    loadings = sweep(
      do.call(rbind, fixed$dlambda),
      MARGIN = 2,
      STATS = fixed$lambda,
      FUN = "+"
    ),
    latent_mean = 0,
    latent_sd = 1,
    pooled_item_sd = pooled_sd
  )[1]
})

# Function for data generation
# sim_y <- function(n, lambda, nu, alpha, psi, Theta) {
#     covy <- tcrossprod(lambda) * psi + Theta
#     meany <- nu + lambda * alpha
#     MASS::mvrnorm(n, mu = meany, Sigma = covy)
# }
generate <- function(condition, fixed_objects) {
  ylist <- lapply(seq_along(fixed_objects$covy),
                  FUN = function(g) {
                    yg <- MASS::mvrnorm(
                      condition$n,
                      mu = fixed_objects$meany[[g]],
                      Sigma = fixed_objects$covy[[g]]
                    )
                    colnames(yg) <- paste0("y", seq_len(fixed_objects$p))
```

```
                    cbind(yg, group  = g)
                  })
  do.call(rbind, ylist)
}
sim1 <- generate(design[3, ], fixed_objects = fixed)


# Analysis
analyze <- function(condition, dat, fixed_objects) {
  # Define lavaan syntax
  pinv_fit <- cfa(
    fixed_objects$mod,
    data = dat,
    group = "group", std.lv = TRUE,
    group.equal = c("loadings", "intercepts"),
    group.partial = c(
      paste0("f=~y", fixed_objects$ninv_ind),
      paste0("y", fixed_objects$ninv_ind, "~1")
    )
  )
  as.vector(pinsearch::pin_effsize(pinv_fit))
}

analyze_bc <- function(condition, dat, fixed_objects) {
  # Define lavaan syntax
  pinv_fit <- cfa(
    fixed_objects$mod,
    data = dat,
    group = "group", std.lv = TRUE,
    group.equal = c("loadings", "intercepts"),
    group.partial = c(
      paste0("f=~y", fixed_objects$ninv_ind),
      paste0("y", fixed_objects$ninv_ind, "~1")
    )
  )
  f_orig <- as.vector(pinsearch::pin_effsize(pinv_fit))
  f_boot <- lavaan::bootstrapLavaan(pinv_fit,
                                    R = 250,
                                    FUN = pinsearch::pin_effsize,
                                    parallel = "snow",
                                    ncpus = detectCores() - 1
```

```
  )
  pmax(0, 2 * f_orig - colMeans(f_boot, na.rm = TRUE))
}

analyze_bc2 <- function(condition, dat, fixed_objects) {
  # Define lavaan syntax
  pinv_fit <- cfa(
    fixed_objects$mod,
    data = dat,
    group = "group", std.lv = TRUE,
    group.equal = c("loadings", "intercepts"),
    group.partial = c(
      paste0("f=~y", fixed_objects$ninv_ind),
      paste0("y", fixed_objects$ninv_ind, "~1")
    )
  )
  f_orig <- pinsearch::pin_effsize(pinv_fit)
  ns <- lavInspect(pinv_fit, what = "nobs")
  ng <- length(ns)
  f2_bias <- (ng - 1) / ng * sum(1 / ns)
  sqrt(pmax(0, f_orig^2 - f2_bias))
}

# Evaluate/Summarize
evaluate <- function(condition, results, fixed_objects) {
  results <- as.matrix(results)
  c(
    bias = colMeans(results) - fixed_objects$dmacs_pop,
    robust_bias = apply(results, 2, mean, trim = .1) -
      fixed_objects$dmacs_pop,
    emp_sd = apply(results, 2, sd),
    emp_mad = apply(results, 2, mad)
  )
}

# out <- runSimulation(design,
#                      replications = 500,
#                      parallel = TRUE,
#                      ncores = 19,
#                      generate = generate,
#                      analyse = list(naive = analyze,
```

```
#                                     bc_boot = analyze_bc,
#                                     bc_form = analyze_bc2),
#                        summarise = evaluate,
#                        filename = "results-dmacs-trial",
#                        save_results = TRUE,
#                        packages = c("MASS", "lavaan", "pinsearch"),
#                        fixed_objects = fixed
# )
```

```
data <- readRDS("C:/Users/alex/OneDrive/Documents/results-dmacs-trial.rds")

bias_table <- data.frame(
  SampleSize = data$n,
  Bias_Naive1 = data$bias.naive,
  Bias_BC_Boot1 = data$bias.bc_boot,
  Bias_BC_Form1 = data$bias.bc_form
)

robust_bias_table <- data.frame(
  SampleSize = data$n,
  Robust_Bias_Naive1 = data$robust_bias.naive,
  Robust_Bias_BC_Boot1 = data$robust_bias.bc_boot,
  Robust_Bias_BC_Form1 = data$robust_bias.bc_form
)

emp_sd_table <- data.frame(
  SampleSize = data$n,
  Empirical_SD_Naive1 = data$emp_sd.naive,
  Empirical_SD_BC_Boot1 = data$emp_sd.bc_boot,
  Empirical_SD_BC_Form1 = data$emp_sd.bc_form
)

emp_mad_table <- data.frame(
  SampleSize = data$n,
  Empirical_MAD_Naive1 = data$emp_mad.naive,
  Empirical_MAD_BC_Boot1 = data$emp_mad.bc_boot,
  Empirical_MAD_BC_Form1 = data$emp_mad.bc_form
)

# Create and display tables using knitr
knitr::kable(bias_table, caption = "Bias for Naive, BC Bootstrap, and Bias-Corrected acros
```

Table 1: Bias for Naive, BC Bootstrap, and Bias-Corrected across Sample Sizes

SampleSize	Bias_Naive1	Bias_BC_Boot1	Bias_BC_Form1
30	12.7580776	14.9586924	12.7084859
100	0.0556084	-0.0243290	0.0324175
250	0.0263493	0.0019652	0.0177839
1000	0.0082686	0.0033549	0.0063548

```r
knitr::kable(robust_bias_table, caption = "Robust Bias for Naive, BC Bootstrap, and Bias-C
```

Table 2: Robust Bias for Naive, BC Bootstrap, and Bias-Corrected across Sample Sizes

SampleSize	Robust_Bias_Naive1	Robust_Bias_BC_Boot1	Robust_Bias_BC_Form1
30	0.2491791	-0.1952833	0.2023053
100	0.0438257	-0.0391292	0.0236123
250	0.0206758	-0.0015779	0.0131089
1000	0.0085598	0.0037110	0.0067025

```r
knitr::kable(emp_sd_table, caption = "Empirical Standard Deviation for Naive, BC Bootstrap
```

Table 3: Empirical Standard Deviation for Naive, BC Bootstrap, and Bias-Corrected across Sample Sizes

SampleSize	Empirical_SD_Naive1	Empirical_SD_BC_Boot1	Empirical_SD_BC_Form1
30	88.7954956	133.5056894	88.8025930
100	0.1730972	0.1870142	0.1866844
250	0.1123075	0.1216431	0.1165287
1000	0.0569815	0.0577051	0.0574197

```r
knitr::kable(emp_mad_table, caption = "Empirical MAD for Naive, BC Bootstrap, and Bias-Cor
```

Table 4: Empirical MAD for Naive, BC Bootstrap, and Bias-Corrected across Sample Sizes

SampleSize	Empirical_MAD_Naive1	Empirical_MAD_BC_Boot1	Empirical_MAD_BC_Form1
30	0.3194361	0.0000000	0.3698344
100	0.1667974	0.2006873	0.1811686
250	0.1094424	0.1187628	0.1118400

SampleSize	Empirical_MAD_Naive1	Empirical_MAD_BC_Boot1	Empirical_MAD_BC_Form1
1000	0.0567682	0.0578033	0.0572080