Online Supplement Using `SimSolve()` for Structural Equation Models

On-line supplement for the article "Solving Variables with Monte Carlo Simulation Experiments: A Stochastic Root-Solving Approach". This supplement contains *SimDesign* code (Chalmers & Adkins, 2020) relevant for solving sample sizes, confidence interval widths, and expected power using the `SimSolve()` function.

Strategy for Deciding on Sample Size in Structural Equation Models

Code associated with the subsection "Strategy for Deciding on Sample Size in Structural Equation Models" for a confirmatory factor analysis model with two latent variables, where interest is in the latent correlation parameter ϕ.

```
# SimDesign::SimFunctions(SimSolve = TRUE)
library(SimDesign)

Generate <- function(condition, fixed_objects = NULL) {
    Attach(condition)
    population.model <-
      sprintf('
        f1 =~ %s
        f2 =~ %s
        f1 ~~ 1*f1
        f2 ~~ 1*f2
        f1 ~~ %f*f2
        %s', paste0(loadings, "*x", 1:5, collapse=' + '),
             paste0(loadings, "*x", 6:10, collapse=' + '),
             fcor,
             paste0(paste0('x', 1:10, " ~~ "),
                    paste0(residuals, "*x", 1:10, '\n'), collapse=''))
    # cat(population.model)
    dat <- simulateData(population.model, sample.nobs=N)
    dat
```

```
}

Analyse <- function(condition, dat, fixed_objects = NULL) {
    myModel <- '
      f1 =~ NA*x1 + x2 + x3 + x4 + x5
      f2 =~ NA*x6 + x7 + x8 + x9 + x10
      f1 ~~ 1*f1
      f2 ~~ 1*f2'
    mod <- sem(myModel, data=dat)
    ests <- parameterEstimates(mod)
    ret <- with(ests, pvalue[paste0(lhs, op, rhs) == "f1~~f2"])
    ret
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    EDR(results, alpha = .05)
}

#-------------------------------------------------------------------

Design <- createDesign(N=NA, fcor=.25, loadings=.8, residuals=.36)

solved <- SimSolve(design=Design, b=.8, interval=c(100,300), generate=Generate,
                   analyse=Analyse, summarise=Summarise, packages='lavaan',
                   parallel=TRUE)
solved
summary(solved)
plot(solved)
plot(solved, type = 'history')
```

Sample Size Estimates for Target CI Widths and Assurance

Code associated with the subsection "Sample Size Estimates for Target CI Widths and Assurance" for a confirmatory factor analysis model with two latent variables, where interest is in obtaining sample size estimates that would result in $P([CI(\phi)_{97.5\%} - CI(\phi)_{2.5\%}] \leq .10) = .80$. Note that the `Generate()` definition is defined above.

```
Analyse <- function(condition, dat, fixed_objects = NULL) {
    myModel <- '
      f1 =~ NA*x1 + x2 + x3 + x4 + x5
      f2 =~ NA*x6 + x7 + x8 + x9 + x10
      f1 ~~ 1*f1
      f2 ~~ 1*f2'
    mod <- sem(myModel, data=dat)
```

```
    ests <- parameterEstimates(mod)
        ret <- subset(ests, paste0(lhs, op, rhs) == "f1~~f2",
                  select=est:ci.upper)
    ret
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    width <- results[,"ci.upper"] - results[,'ci.lower']
    mean(width <= condition$CI.width)
}

#-----------------------------------------------------------------
# SimSolve()

Design <- createDesign(N=NA, fcor=.25, loadings=.8, residuals=.36, CI.width=.1)

solved <- SimSolve(design=Design, b=.8, interval=c(100,2000), generate=Generate,
                   analyse=Analyse, summarise=Summarise, packages='lavaan',
                   parallel=TRUE)
solved
summary(solved)
plot(solved)
plot(solved, type = 'history')

# obtain additional information on confirmation simulation
csolved <- solved
csolved$N <- ceiling(csolved$N)
SummariseConfirm <- function(condition, results, fixed_objects = NULL) {
    width <- with(results, ci.upper - ci.lower)
    ret <- with(results, data.frame(est=mean(est), se=mean(se),
                                    ci.lower=mean(ci.lower),
                                    ci.upper=mean(ci.upper),
                                    width=mean(ci.upper - ci.lower),
                                    prop=mean(width <= condition$CI.width)))
    ret
}

res2 <- runSimulation(design=csolved, replications=10000, generate=Generate,
                      analyse=Analyse, summarise=SummariseConfirm,
                      packages='lavaan', parallel=TRUE)
res2
```

Solving Sample Size for Expected Power

Code associated with the subsection Solving Sample Size for Expected Power" for a confirmatory factor analysis model with two latent variables, where interest is in the latent correlation parameter ϕ given focal and non-focal effect size uncertainty.

```
library(SimDesign)

Generate <- function(condition, fixed_objects = NULL) {
    Attach(condition)
    if(prior.type %in% c('fcor', 'all'))
        fcor <- rnorm(1, .25, .1)
    if(prior.type %in% c('loadings', 'all'))
        loadings <- rbeta(1, 8, 2)
    residuals <- 1 - loadings^2
    population.model <-
      sprintf('
        f1 =~ %s
        f2 =~ %s
        f1 ~~ 1*f1
        f2 ~~ 1*f2
        f1 ~~ %f*f2
        %s', paste0(loadings, "*x", 1:5, collapse=' + '),
             paste0(loadings, "*x", 6:10, collapse=' + '),
             fcor,
             paste0(paste0('x', 1:10, " ~~ "),
                    paste0(residuals, "*x", 1:10, '\n'), collapse=''))
    # cat(population.model)
    dat <- simulateData(population.model, sample.nobs=N)
    dat
}

Analyse <- function(condition, dat, fixed_objects = NULL) {
    myModel <- '
      f1 =~ NA*x1 + x2 + x3 + x4 + x5
      f2 =~ NA*x6 + x7 + x8 + x9 + x10
      f1 ~~ 1*f1
      f2 ~~ 1*f2'
    mod <- sem(myModel, data=dat)
    ests <- parameterEstimates(mod)
    ret <- with(ests, pvalue[paste0(lhs, op, rhs) == "f1~~f2"])
    ret
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    EDR(results, alpha = .05)
```

```
}

#-----------------------------------------------------------------------

Design <- createDesign(N=NA, fcor=.25, loadings=.8,
                       prior.type = c('fcor', 'loadings', 'all'))

solved <- SimSolve(design=Design, b=.8, interval=c(100,1000), generate=Generate,
                   analyse=Analyse, summarise=Summarise, packages='lavaan',
                   parallel=TRUE)
solved

# Run root-solver for 30 minutes or 500 iterations, whichever occurs first
solved_30 <- SimSolve(design=Design, b=.8, interval=c(100,1000), generate=Generate,
                      analyse=Analyse, summarise=Summarise, packages='lavaan',
                      parallel=TRUE, wait.time=30, maxiter=500)
solved_30
```

References

Chalmers, R. P., & Adkins, M. C. (2020). Writing effective and reliable Monte Carlo simulationswith the SimDesign package. *The Quantitative Methods for Psychology*, *16*(4), 248–280. doi: 10.20982/tqmp.16.4.p248