checkDataSimulation.R

kim

2024-03-14

```r
# simulate one large data set for each of the simulated conditions
# confirm simulation by using SEM to recover simulated parameters
# compare recovered parameters to true, simulated parameters
# plot?

# load packages
library(mvtnorm)
library(truncnorm)
library(parallel)

# load parameters & custom functions
source("setParameters.R") # parameter values
source("simTools.R") # functions for data simulation

# grid to simulate data with mapply later
# simulation via mapply to easily simulate subsets of parameter combinations
# create factor structure for predictor variables to increase reliability
gridFull <- expand.grid(pTrash = setParam$dgp$pTrash,
                        reliability = setParam$dgp$reliability)
# ! factors are interpreted as level numbers!; only character variables are interpreted by their name!
str(gridFull)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ pTrash     : num  10 50 10 50 10 50
##  $ reliability: num  0.6 0.6 0.8 0.8 1 1
##  - attr(*, "out.attrs")=List of 2
##   ..$ dim     : Named int [1:2] 2 3
##   .. ..- attr(*, "names")= chr [1:2] "pTrash" "reliability"
##   ..$ dimnames:List of 2
##   .. ..$ pTrash     : chr [1:2] "pTrash=10" "pTrash=50"
##   .. ..$ reliability: chr [1:3] "reliability=0.6" "reliability=0.8" "reliability=1.0"
```

```r
# create seed number for parallel cluster (reproducibility of generated data)
set.seed(8967369)
seedNum <- sample(1:999999, dim(gridFull)[1], replace = FALSE)
gridFull$sampleSeed <- seedNum[1:dim(gridFull)[1]]

# simulate a very large sample size to accurately estimate parameters
N <- 100000

checkSimData <- function(pTrash, reliability, sampleSeed){
  set.seed(sampleSeed)
  print(paste0("in progress: ", pTrash, " x ", reliability))
```

```r
##### simulate data ######
P <- setParam$dgp$p + pTrash # total number of variables
# generate matrix of (almost) uncorrelated predictors

# get predictor values without (!) measurement error
X <- createPredictors(N = N, P = P,
                      corMat = setParam$dgp$predictorCorMat[seq_len(P), seq_len(P)])

# add names to variables
colnames(X) <- paste0("Var", seq_len(P))

# create model formula (allows polynomial and interaction effects of any degree/depth)
popModel <- genModel(colnames(X), setParam$dgp$interDepth, setParam$dgp$poly)

# predictor matrix that allows for polynomials and interactions
X_int <- model.matrix(as.formula(popModel), data.frame(X))

# remove first degree polynomials from data (they are duplicates!)
#   only if poly in model matrix, else error
if (setParam$dgp$poly > 0) {
  X_int <- rmDuplicatePoly(X_int)
}

# generate matrix of regression coefficients (matrix includes all conditions)
# rows represent predictors (thus, number of rows depends on pTrash which varies
#     between simulated conditions)
# columns represent conditions (= combination of R2 and lin/inter effect balance)
bMatrix <- genBmat(X_int, setParam)

# calculate R^2 for every combination of R2 and lin/inter effect balance
# print R^2 as a quick sanity check (removed for speed sake)
R2 <- sapply(seq_len(ncol(bMatrix)), function(x) getR2(X_int, bMatrix[,x], setParam$dgp$sigmaE))

# calculate dependent variable for every combination of R2 and lin/inter effect balance
# dependent variable is simulated from predictors without measurement error!
yMatrix <- sapply(seq_len(ncol(bMatrix)), function(x) {
  calcDV(X = X_int, b = bMatrix[,x],
         sigmaE = setParam$dgp$sigmaE, N = N)
})
colnames(yMatrix) <- setParam$dgp$condLabels

# add measurement error/reliability manipulatio to data
#   add measurement error only to X (poly & interactions are calculated based on X)
#   measurement error ...
#     ... independent for each predictor
#     ... normally distributed with M = 0 & SD according to reliability

# error variance according to reliability
covMatError <- diag(P) * (1 - reliability)/reliability
measureError <- rmvnorm(n = N, mean = rep(0, P), sigma = covMatError)

# add measurement error to predictors
X_wME <- X + measureError
```

```r
  # predictor matrix that allows for polynomials and interactions
  X_final <- model.matrix(as.formula(popModel), data.frame(X_wME))

  # remove first degree polynomials from data (they are duplicates!)
  if (setParam$dgp$poly > 0) {
    X_final <- rmDuplicatePoly(X_final)
  }

  ##### fit SEM to check data simulation ######
  # run single indicator SEM to check if reliabilities are simulated correctly
  # idea: fix residuals of the items according to the simulated reliability and check
  #       if correlations between factors and path coefficients between predictors
  #       and outcome match the true, simulated parameters
  SImodel <- genSingleIndicatorModel(P, reliability)
  checkSimParam <- lapply(seq_len(dim(yMatrix)[2]), function(iR2_LI) {
    X_check <- cbind(X_final[,1:P], y = yMatrix[,iR2_LI])
    R2 <- stringr::str_sub(colnames(yMatrix)[iR2_LI], start = 3L, end = 5L)
    lin_inter <- stringr::str_sub(colnames(yMatrix)[iR2_LI], start = 15L)

    fit <- lavaan::sem(SImodel, data=X_check, se = "none")
    # lavaan::summary(fit) # check lavaan output

    # save path coefficients for predictors with simulated effects
    estBeta <- fit@Model@GLIST[["beta"]][(P+1),seq_along(setParam$dgp$linEffects)]

    # save correlations between latent variables of correlated predictors
    estPsi <- fit@Model@GLIST[["psi"]][seq_along(setParam$dgp$linEffects), seq_along(setParam$dgp$linEf
    estPsi <- estPsi[upper.tri(estPsi)] # F1F2, F1F3, F2F3, F1F4, F2F4, F3F4

    list(estBeta = estBeta,
         estPsi = estPsi,
         R2 = R2,
         lin_inter = lin_inter)
  })

  estPsi <- do.call(rbind, lapply(seq_along(checkSimParam), function(subList) {
    tmp <- rbind(checkSimParam[[subList]][["estPsi"]])
    cbind(tmp, checkSimParam[[subList]][["R2"]], checkSimParam[[subList]][["lin_inter"]])
  }))
  colnames(estPsi) <- c("F1F2", "F1F3", "F2F3", "F1F4", "F2F4", "F3F4", "R2", "lin_inter")

  estBeta <- do.call(rbind, lapply(seq_along(checkSimParam), function(subList) {
    tmp <- rbind(checkSimParam[[subList]][["estBeta"]])
    cbind(tmp, checkSimParam[[subList]][["R2"]], checkSimParam[[subList]][["lin_inter"]])
  }))
  estBeta <- cbind(estBeta, matrix(setParam$dgp$trueEffects$lin, ncol = 1))
  colnames(estBeta) <- c(setParam$dgp$linEffects, "R2", "lin_inter", "trueBeta")

  return(list(estBeta = estBeta,estPsi = estPsi))
}

# # test it
# checkSimData(pTrash = 10, reliability = 0.6, sampleSeed = 42)
# # run data simulation and SEM fitting again or load the saved data (see plot check data below)
# out <- do.call(mapply, c(FUN = checkSimData, gridFull, SIMPLIFY = FALSE))

# # save data
# save(out, file = "checkDataSimulation.rda")
################################################################################
# plot check data
################################################################################
# load data
load("checkDataSimulation.rda")

# concatenate matrices across condition sublists (R2 x lin_inter)
estBeta <- do.call(rbind, lapply(seq_along(out), function(subList) {
  cbind(out[[subList]][["estBeta"]],
        gridFull[subList,1:2])
}))
```

```
## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen
```

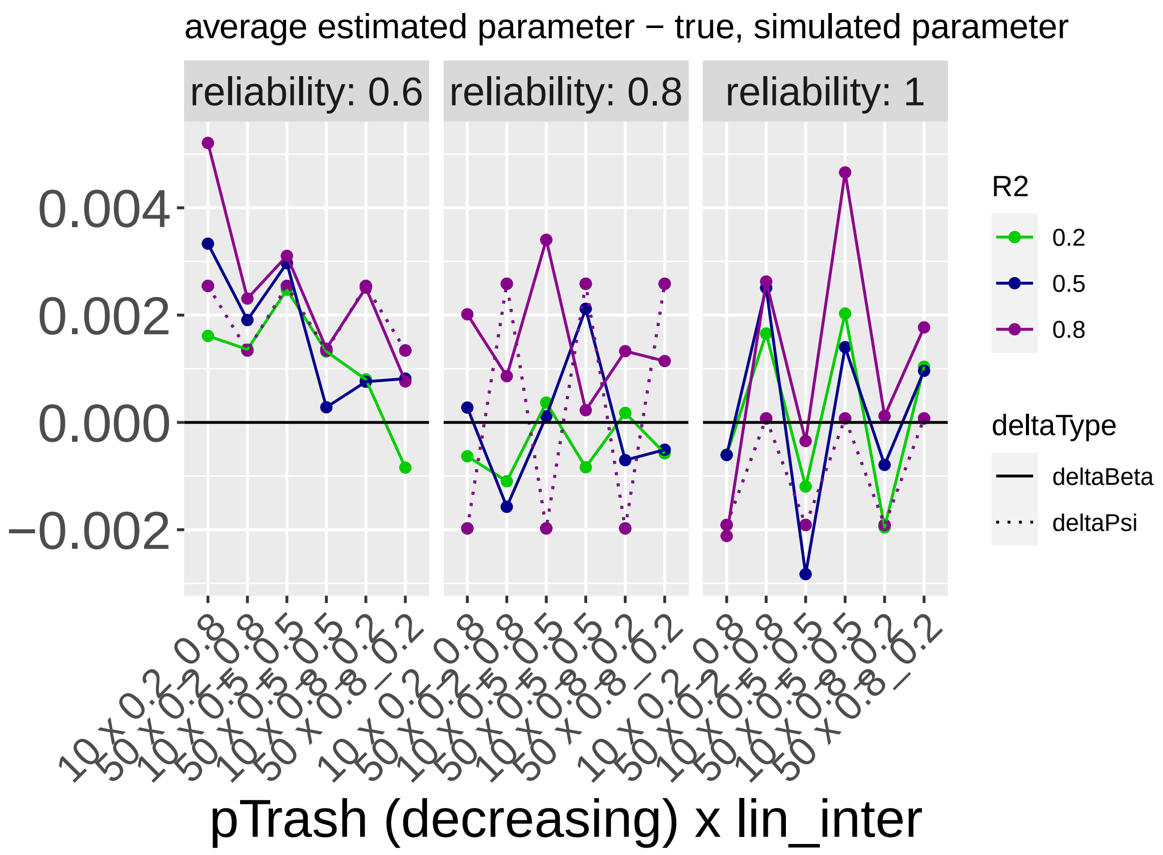
```
estPsi <- do.call(rbind, lapply(seq_along(out), function(subList) {
  cbind(out[[subList]][["estPsi"]],
        gridFull[subList,1:2])
}))
```

```
## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen

## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen
## Warning in data.frame(..., check.names = FALSE): Zeilennamen wurden in einer
## short Variablen gefunden und wurden verworfen
```

```
# change parameter value variables to numeric
str(estBeta)
```

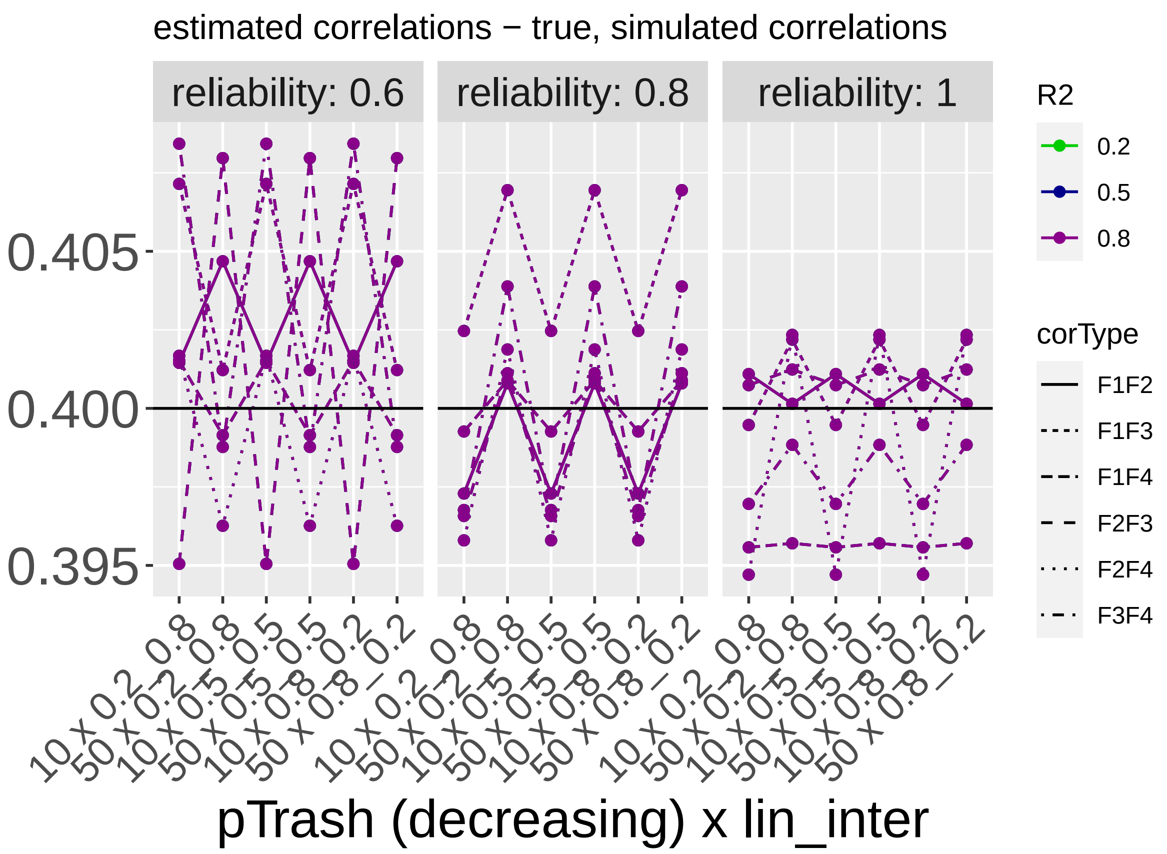
```
## 'data.frame':    54 obs. of  9 variables:
##  $ Var1       : chr  "0.11348426365509" "0.245807196994173" "0.476626626801551" "0.143951488723046"
##  $ Var2       : chr  "0.128582799980491" "0.236294463173194" "0.477503760824989" "0.15594492482797"
##  $ Var3       : chr  "0.121780105489031" "0.237543012287831" "0.481346146833463" "0.153156049006951"
##  $ Var4       : chr  "0.121049782866794" "0.241292761661512" "0.467483879514319" "0.151039256466369"
##  $ R2         : chr  "0.2" "0.5" "0.8" "0.2" ...
##  $ lin_inter  : chr  "0.5_0.5" "0.5_0.5" "0.5_0.5" "0.8_0.2" ...
##  $ trueBeta   : chr  "0.118751224814342" "0.237266112812037" "0.472635708907579" "0.150221615222395"
##  $ pTrash     : num  10 10 10 10 10 10 10 10 10 50 ...
##  $ reliability: num  0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 ...
```

```
col2num.beta <- c(setParam$dgp$linEffects, "trueBeta")
estBeta[col2num.beta] <- lapply(estBeta[col2num.beta], as.numeric)
col2num.psi <- c("F1F2", "F1F3", "F2F3", "F1F4", "F2F4", "F3F4")
estPsi[col2num.psi] <- lapply(estPsi[col2num.psi], as.numeric)

# calculate average estimates across variables and correlations
# (all variables with simulated effects and all correlations are respectively the same)
estBeta$mBeta <- rowMeans(estBeta[, setParam$dgp$linEffects], na.rm = T)
estBeta$deltaBeta <- estBeta$mBeta - estBeta$trueBeta

estPsi$mPsi <- rowMeans(estPsi[, col2num.psi], na.rm = T)
estPsi$deltaPsi <- estPsi$mPsi - setParam$dgp$Reffects

estParam <- merge(estBeta, estPsi)
estParamLong <- tidyr::pivot_longer(estParam, c(deltaPsi, deltaBeta),
                                    names_to = "deltaType", values_to = "delta")

# plot all conditions in any random arrangement to quickly check the recovered parameters
library(ggplot2)

colValues <- c("green3", "darkblue", "darkmagenta")

# mean values for correlations and pathcoefficients
(ggplot(estParamLong,
        aes(x = interaction(pTrash, lin_inter, sep = " x "), y = delta,
            group = interaction(R2, deltaType), colour = R2, linetype = deltaType)) +
    geom_point() +
    geom_line() +
    scale_linetype_manual(values = c("solid", "dotted")) +
    scale_color_manual(values = colValues) +
    geom_hline(aes(yintercept = 0)) +
    facet_wrap(~ reliability, labeller = label_both) +
    ylab("") +
    xlab("pTrash (decreasing) x lin_inter") +
    ggtitle("average estimated parameter - true, simulated parameter") +
    theme(axis.text.y = element_text(size = 20),
          axis.text.x = element_text(size = 15, angle = 45, vjust = 1, hjust=1),
          axis.title.x = element_text(size = 20),
          axis.title.y = element_text(size = 20),
          strip.text.x = element_text(size = 15),
          strip.text.y = element_text(size = 15)))
```

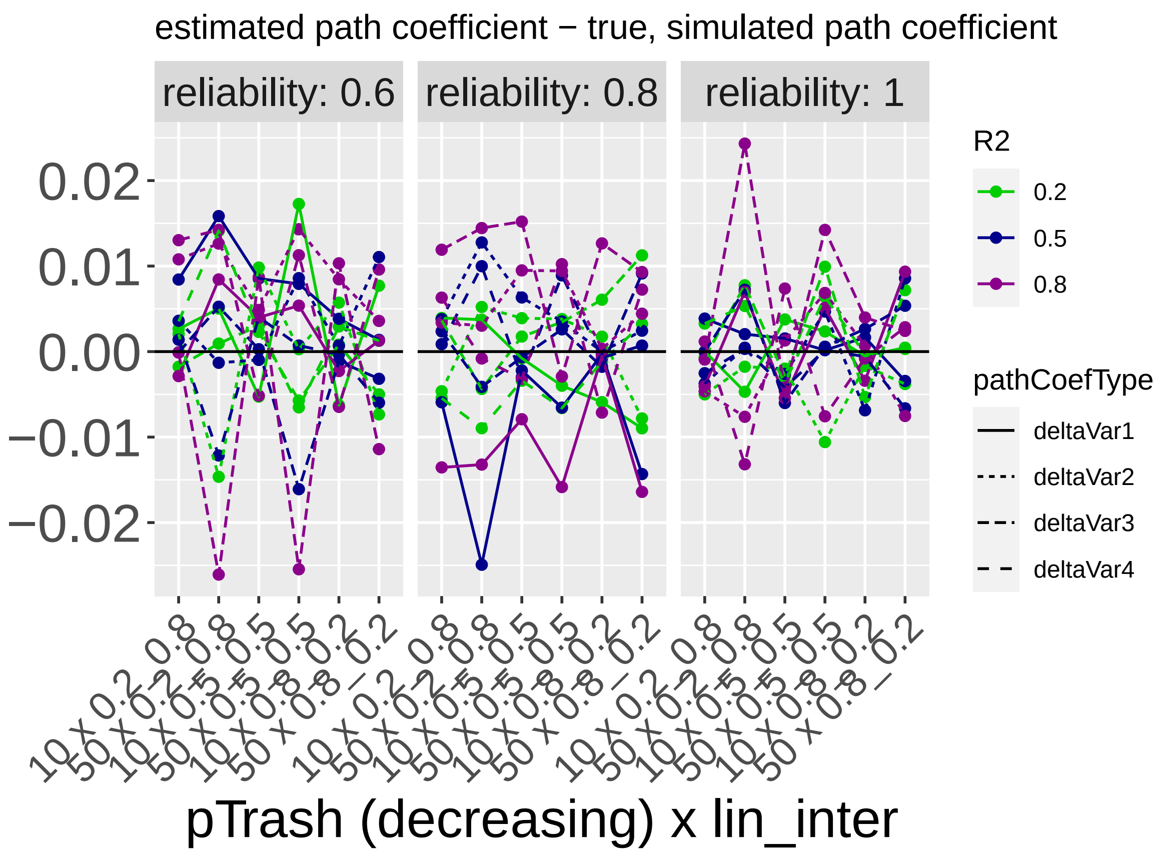


```
# all recovered correlations
estPsiLong <- tidyr::pivot_longer(estPsi, c(tidyr::all_of(col2num.psi)),
                                  names_to = "corType", values_to = "cor")
(ggplot(estPsiLong,
        aes(x = interaction(pTrash, lin_inter, sep = " x "), y = cor,
            group = interaction(R2, corType), colour = R2, linetype = corType)) +
    geom_point() +
    geom_line() +
    # scale_linetype_manual(values = c("solid", "dotted")) +
    # geom_errorbar(aes(ymin = M - SE, ymax = M + SE), width=.2) +
    scale_color_manual(values = colValues) +
    geom_hline(aes(yintercept = setParam$dgp$Reffects)) +
    facet_wrap(~ reliability, labeller = label_both) +
    ylab("") +
    xlab("pTrash (decreasing) x lin_inter") +
    ggtitle("estimated correlations - true, simulated correlations") +
    theme(axis.text.y = element_text(size = 20),
          axis.text.x = element_text(size = 15, angle = 45, vjust = 1, hjust=1),
          axis.title.x = element_text(size = 20),
          axis.title.y = element_text(size = 20),
          strip.text.x = element_text(size = 15),
```

```r
        strip.text.y = element_text(size = 15)))
```



```r
# all recovered path coefficients
# substract
subtractTrue <- function(varName){
  return(varName - estBeta$trueBeta)
}

estBeta <- cbind(estBeta, apply(estBeta[,setParam$dgp$linEffects], 2, subtractTrue))
deltaVars <- sapply(setParam$dgp$linEffects, function(x) paste0("delta", x))
colnames(estBeta)[12:15] <- deltaVars

estBetaLong <- tidyr::pivot_longer(estBeta, c("deltaVar1", "deltaVar2", "deltaVar3", "deltaVar4"),
                                  names_to = "pathCoefType", values_to = "pathCoef")

(ggplot(estBetaLong,
        aes(x = interaction(pTrash, lin_inter, sep = " x "), y = pathCoef,
            group = interaction(R2, pathCoefType), colour = R2, linetype = pathCoefType)) +
    geom_point() +
    geom_line() +
    scale_color_manual(values = colValues) +
    geom_hline(aes(yintercept = 0)) +
    facet_wrap(~ reliability, labeller = label_both) +
    ylab("") +
    xlab("pTrash (decreasing) x lin_inter") +
    ggtitle("estimated path coefficient - true, simulated path coefficient") +
    theme(axis.text.y = element_text(size = 20),
          axis.text.x = element_text(size = 15, angle = 45, vjust = 1, hjust=1),
          axis.title.x = element_text(size = 20),
          axis.title.y = element_text(size = 20),
          strip.text.x = element_text(size = 15),
          strip.text.y = element_text(size = 15)))
```


estimated path coefficient − true, simulated path coefficient
reliability: 0.6
reliability: 0.8
reliability: 1
10 x 0.2_0.8
50 x 0.2_0.8
10 x 0.5_0.5
50 x 0.5_0.5
10 x 0.8_0.2
50 x 0.8_0.2
pTrash (decreasing) x lin_inter
R2
pathCoefType
deltaVar1
deltaVar2
deltaVar3
deltaVar4