```
library(OpenMx)
```

```
## Loading required package: digest
```

```
## Loading required package: MASS
```

```
## Loading required package: Matrix
```

```
## Loading required package: Rcpp
```

```
## Loading required package: parallel
```

```
## To take full advantage of multiple cores, use:
##   mxOption(NULL, 'Number of Threads', parallel::detectCores())
```

```
##
## Attaching package: 'OpenMx'
```

```
## The following objects are masked from 'package:Matrix':
##
##     %&%, expm
```

```
mxVersion()
```

```
## OpenMx version: 2.6.8.198 [GIT v2.6.8-198-gacf5469]
## R version: R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu
## Default optimiser: CSOLNP
```

```
# seems to happen more often with SLSQP
mxOption(NULL, key="Default optimizer", value="SLSQP")
#mxOption(NULL, key="Default optimizer", value="NPSOL")
mxOption(NULL, key="Default optimizer")
```

```
## [1] "SLSQP"
```

```
nruns<-10
for(i in 1:nruns){

print(i)

# set up data
dat <- read.table(file="dat100NormalNA198.dat")
dat<-as.data.frame(dat)
colnames(dat)<-paste("X",1:8,sep="")
data <- mxData( observed=dat, type="raw" )

# residual variances
resVars      <- mxPath( from=paste("X",1:8,sep=""), arrows=2,
```

```
                        free=TRUE, values=rep(1,8),
                        labels=paste("e",1:8,sep="") )
# latent variance
latVars       <- mxPath( from=c("F1","F2"), arrows=2,connect="unique.pairs",
                         free=c(FALSE,TRUE,FALSE), values=c(1,0,1), labels =c("varF1","cov","varF2") )
# factor loadings
facLoads1     <- mxPath( from="F1", to=paste("X",1:4,sep=""), arrows=1,
                         free=rep(TRUE,4), values=rep(1,4),
                         labels =paste("l",1:4,sep="") )
facLoads2     <- mxPath( from="F2", to=paste("X",5:8,sep=""), arrows=1,
                         free=rep(TRUE,4), values=rep(1,4),
                         labels =paste("l",5:8,sep="") )
# means
means        <- mxPath( from="one", to=c(paste("X",1:8,sep=""),"F1","F2"), arrows=1,
                        free=c(rep(TRUE,8),FALSE,FALSE), values=c(rep(1,8),0,0),
                        labels =c(paste("meanx",1:8,sep=""),NA,NA) )

# what parameter are we interested in...
mxpindx<-"cov"

CIModel <- mxModel("CIModel", type="RAM",
                   manifestVars=paste("X",1:8,sep=""), latentVars=c("F1","F2"),
                   data, resVars, latVars, facLoads1, facLoads2, means)

# fit model
#CIModel<-mxRun(CIModel)

# Interval; typical .95
alpha<-.05
CI<-mxCI(mxpindx,interval=1-alpha,type="both", boundAdj=TRUE)
CIFit <- try(mxRun(mxModel(model=CIModel, CI), intervals = TRUE, silent=TRUE))
if(class(CIFit)=="try-error"){
  print(CIFit)
}

}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] "Error in runHelper(model, frontendStart, intervals, silent, suppressWarnings,  : \n  NLOPT unre
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in runHelper(model, frontendStart, intervals, silent, suppressWarnings,     unsafe, che
## [1] 8
## [1] "Error in runHelper(model, frontendStart, intervals, silent, suppressWarnings,  : \n  NLOPT unre
## attr(,"class")
## [1] "try-error"
```

```
## attr(,"condition")
## <simpleError in runHelper(model, frontendStart, intervals, silent, suppressWarnings,     unsafe, che
## [1] 9
## [1] "Error in runHelper(model, frontendStart, intervals, silent, suppressWarnings,  : \n  NLOPT unre
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in runHelper(model, frontendStart, intervals, silent, suppressWarnings,     unsafe, che
## [1] 10
## [1] "Error in runHelper(model, frontendStart, intervals, silent, suppressWarnings,  : \n  NLOPT unre
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in runHelper(model, frontendStart, intervals, silent, suppressWarnings,     unsafe, che
```