Numeric Example

init

Set the library's directory first!

```
#Load Libraries and functions
setwd("/home/tchronis/Projects/denisa/MSE-R")
source("mse.R")
```

Import precomputed data

Set the data's directory preferably in the variable 'filename'.

```
#filename<-"import/round1m1-1.xls.pre.dat"  # 3 attributes
filename<-"import/precomp_testdata.dat" # 5 attributes
```

Load the data in variables with meaningful names

```
x<-import(filename)
g(header,noM,noU,noD,noAttr,distanceMatrices,matchMatrix,mate)%=%x
```

Routines (calculate payoff matrix, inequalities members, dataArray)

```
#Create payoffMatrix
Cx<-Cx(noAttr)
payoffMatrix<-CpayoffMatrix(noM,noU,noD,Cx,distanceMatrices,noAttr)

#Assign payoffMatrix numerical values (set x's)
#xval<-c(1,2)  # 3 attributes
xval<-c(1,2,3,4)  # 5 attributes
payoffMatrix<-assignpayoffMatrix(payoffMatrix,xval)
```

```
#Create inequality members
ineqmembers<-Cineqmembers(mate)
```

```
#Create Data Array
dataArray<-CdataArray(distanceMatrices,ineqmembers)
```

Maximization

Differential Evolution Method

The default DifferentialEvolution parameters:

```
#Objective function
coefficient1<-1
b<-Cx #Define x1,x2,... values
#obj<-objective(b)

#maximize function
```

option name	default value	
lower,upper	-10,10	two vectors specifying scalar real lower and upper bounds on each parameter to be optimiz
CR	0.5	crossover probability from interval [0,1]
trace	FALSE	Positive integer or logical value indicating whether printing of progress occurs at each itera
itermax	100	the maximum iteration (population generation) allowed
F	0.6	differential weighting factor from interval [0,2]
NP	50	number of population members. Defaults to NA; if the user does not change the value of N
reltol	0.001	relative convergence tolerance. The algorithm stops if it is unable to reduce the value by a
RandomSeed	0	Random Seed to be used for result reproducibility

```
#lower <- c(-10, -10)  # 3 attributes
lower <- c(-10, -10, -10, -10)  # 5 attributes
upper <- -lower
par<-list(lower=lower,upper=upper,NP=50,itermax=100,trace=FALSE,reltol=0.001,CR=0.5,F=0.6,RandomSeed=0)
x<-maximize(par)
g(bestmem,bestval)%=%x
print(bestmem)
```

```
##       par1       par2       par3       par4
##  1.1744638 -6.7039417  3.3455970  0.2178138
```

```
print(bestval)
```

```
## [1] 94
```

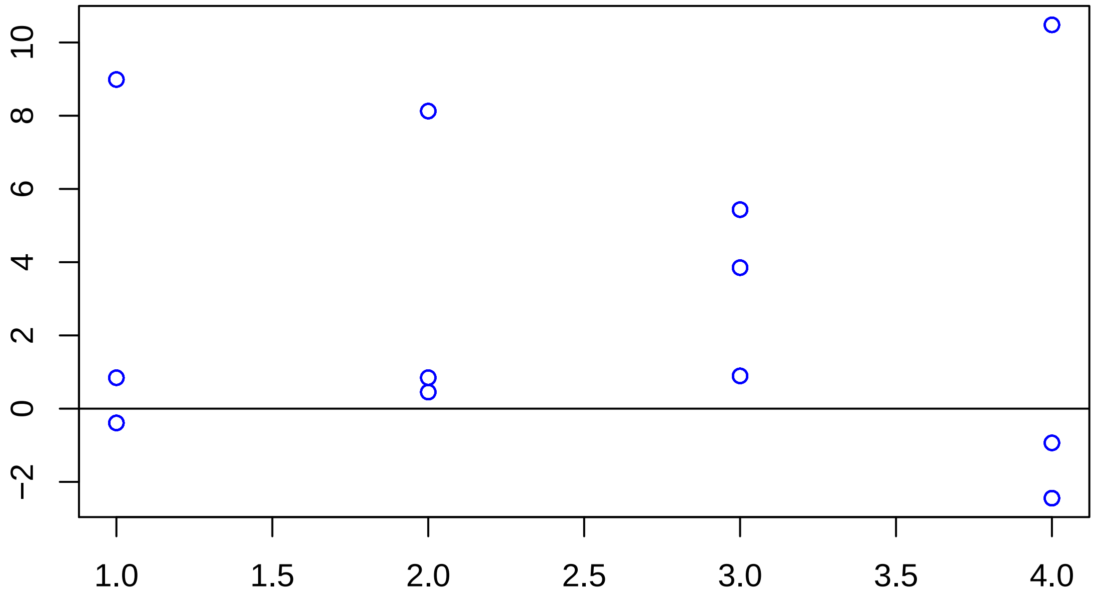
Confidence Intervals

Generate random subsample

```
#Create groupIDs
groupIDs<-groupIDs(ineqmembers)


#ssSize<-3  # 3 attributes, markets to select
ssSize<-2  # 5 attributes, markets to select

options<-list()
options["progressUpdate"]<-1
options["confidenceLevel"]<-0.95
options["asymptotics"]<-"nests"
options["symmetric"]<-FALSE

numSubsamples<-50
pointEstimate<-as.numeric(bestmem)
#b<-Cx[2:3]
b<-Cx[2:5]

#lower <- c(-10, -10)  # 3 attributes
lower <- c(-10, -10, -10, -10)  # 5 attributes
upper <- -lower
par<-list(lower=lower,upper=upper,NP=50,itermax=100,trace=FALSE,reltol=0.001,CR=0.5,F=0.6,RandomSeed=0)
```

```r
pointIdentifiedCR(ssSize, numSubsamples,pointEstimate,Cx,groupIDs,dataArray,options,par)
```

```
## Iterations completed: 1
## Iterations completed: 2
## Iterations completed: 3
## Iterations completed: 4
## Iterations completed: 5
## Iterations completed: 6
## Iterations completed: 7
## Iterations completed: 8
## Iterations completed: 9
## Iterations completed: 10
## Iterations completed: 11
## Iterations completed: 12
## Iterations completed: 13
## Iterations completed: 14
## Iterations completed: 15
## Iterations completed: 16
## Iterations completed: 17
## Iterations completed: 18
## Iterations completed: 19
## Iterations completed: 20
## Iterations completed: 21
## Iterations completed: 22
## Iterations completed: 23
## Iterations completed: 24
## Iterations completed: 25
## Iterations completed: 26
## Iterations completed: 27
## Iterations completed: 28
## Iterations completed: 29
## Iterations completed: 30
## Iterations completed: 31
## Iterations completed: 32
## Iterations completed: 33
## Iterations completed: 34
## Iterations completed: 35
## Iterations completed: 36
## Iterations completed: 37
## Iterations completed: 38
## Iterations completed: 39
## Iterations completed: 40
## Iterations completed: 41
## Iterations completed: 42
## Iterations completed: 43
## Iterations completed: 44
## Iterations completed: 45
## Iterations completed: 46
## Iterations completed: 47
## Iterations completed: 48
## Iterations completed: 49
## Iterations completed: 50
```



```
## [[1]]
## [[1]][[1]]
## [1] "Symmetric case"
##
## [[1]][[2]]
## [[1]][[2]][[1]]
## [1] -5.057317  7.406245
##
## [[1]][[2]][[2]]
## [1] -12.338085  -1.069799
##
## [[1]][[2]][[3]]
## [1] -0.4226243  7.1138184
##
## [[1]][[2]][[4]]
## [1] -7.048985  7.484612
##
##
##
## [[2]]
## [[2]][[1]]
## [1] "Asymmetric case"
##
## [[2]][[2]]
## [[2]][[2]][[1]]
## [1] -5.057317  1.444858
```

```
##
## [[2]][[2]][[2]]
## [1] -12.338085  -7.017411
##
## [[2]][[2]][[3]]
## [1] -0.4226243  2.7257690
##
## [[2]][[2]][[4]]
## [1] -7.048985  1.912858
##
##
##
## [[3]]
##             [,1]      [,2]      [,3]       [,4]
##  [1,]  8.9877832 0.8460268 3.8503105 10.4805373
##  [2,]  8.9877832 0.8460268 3.8503105 10.4805373
##  [3,]  0.8443868 0.4521011 0.8939467 -0.9350565
##  [4,]  0.8443868 0.4521011 0.8939467 -0.9350565
##  [5,]  8.9877832 0.8460268 3.8503105 10.4805373
##  [6,] -0.3899754 8.1258405 5.4347156 -2.4446774
##  [7,]  8.9877832 0.8460268 3.8503105 10.4805373
##  [8,]  0.8443868 0.4521011 0.8939467 -0.9350565
##  [9,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [10,]  8.9877832 0.8460268 3.8503105 10.4805373
## [11,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [12,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [13,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [14,]  8.9877832 0.8460268 3.8503105 10.4805373
## [15,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [16,]  8.9877832 0.8460268 3.8503105 10.4805373
## [17,]  8.9877832 0.8460268 3.8503105 10.4805373
## [18,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [19,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [20,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [21,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [22,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [23,]  8.9877832 0.8460268 3.8503105 10.4805373
## [24,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [25,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [26,]  8.9877832 0.8460268 3.8503105 10.4805373
## [27,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [28,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [29,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [30,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [31,]  8.9877832 0.8460268 3.8503105 10.4805373
## [32,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [33,]  8.9877832 0.8460268 3.8503105 10.4805373
## [34,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [35,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [36,]  8.9877832 0.8460268 3.8503105 10.4805373
## [37,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [38,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [39,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [40,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [41,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [42,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [43,]  8.9877832 0.8460268 3.8503105 10.4805373
## [44,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [45,]  0.8443868 0.4521011 0.8939467 -0.9350565
## [46,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [47,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [48,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [49,] -0.3899754 8.1258405 5.4347156 -2.4446774
## [50,]  0.8443868 0.4521011 0.8939467 -0.9350565
```