EARMOScript.r

moar82

Sun Jan 31 19:56:30 2016

```
####Replication script
####Date: January 2016
####Author Rodrigo Morales
####Subject: This is script is part of the replication package for the work presented in the
####Paper Titled: EARMO: An Energy-Aware Refactoring Approach for Mobile Apps
####Conference: MSR '16
####Contact: to rodrigo.morales@polymtl.ca


rm(list = ls())
require(data.table)
```

```
## Loading required package: data.table
```

```
## Warning: package 'data.table' was built under R version 3.2.3
```

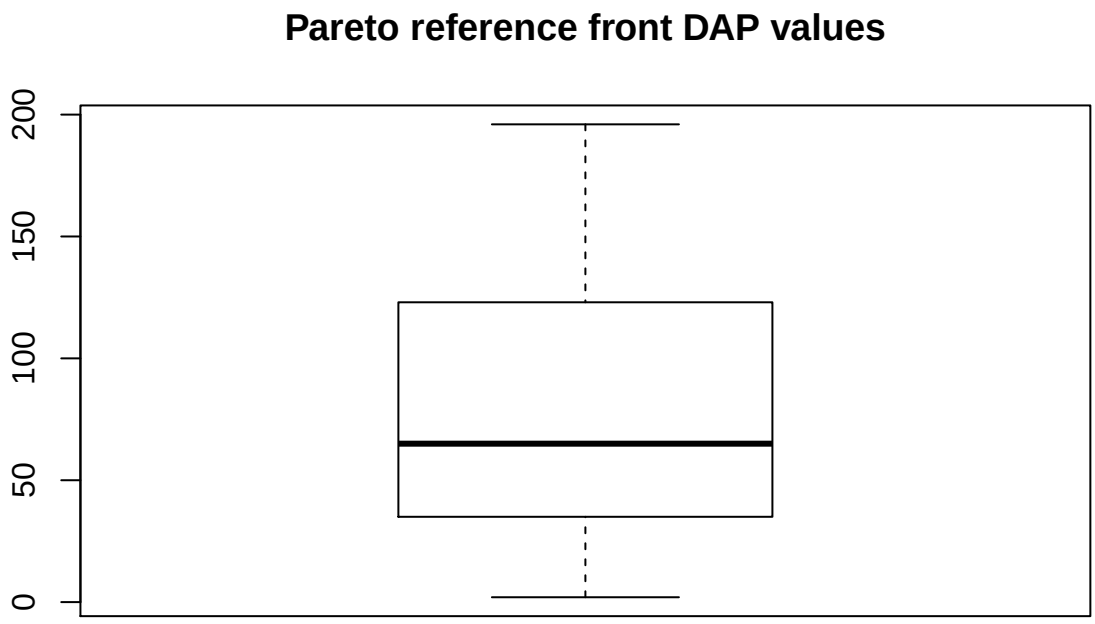
```
###Substitute for the path where EARMODataSetPretty.csv is located in your system
setwd("R:/msr16RepPack/")


dataRF <-read.csv("EARMODataSetPretty.csv",sep=",")


print ("dtotalAP corresponds to DAP")
```

```
## [1] "dtotalAP corresponds to DAP"
```

```
summary(dataRF$dtotalAP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   35.00   65.00   78.47  123.00  196.00
```

```
boxplot(dataRF$dtotalAP, main="Pareto reference front DAP values")
```




```
dev.copy(pdf,'antiPattDist.pdf')
```

```
## pdf
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

```
summary(dataRF$fitnessEnergy)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        2      130      248   341000      554 10560000
```

```
print("compute statistics about the number of classes and types of anti-patterns
this is necessary as the dataset contains all the non-dominated solutios per app,
and if we do not aggregate we will double count
The min function is neutral, because the original values are the same for all the rows.")
```

```
## [1] "compute statistics about the number of classes and types of anti-patterns\nthis is necessary as
```

```
vecNOC<-aggregate(NOC ~ appName, dataRF,
                  function(x) min (x))
sum(vecNOC$NOC)
```

```
## [1] 14907
```

```
vecBlob<-aggregate(BlobOriginalDesign ~ appName, dataRF,
                   function(x) min (x))
sum(vecBlob$BlobOriginalDesign)
```

```
## [1] 759
```

```
vecLC<-aggregate(LazyClassOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecLC$LazyClassOriginalDesign)
```

```
## [1] 855
```

```
vecLP<-aggregate(LongParameterListOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecLP$LongParameterListOriginalDesign)
```

```
## [1] 1493
```

```
vecRB<-aggregate(RefusedParentBequestOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecRB$RefusedParentBequestOriginalDesign)
```

```
## [1] 1427
```

```
vecSC<-aggregate(SpaghettiCodeOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecSC$SpaghettiCodeOriginalDesign)
```

```
## [1] 60
```

```
vecSG<-aggregate(SpeculativeGeneralityOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecSG$SpeculativeGeneralityOriginalDesign)
```

```
## [1] 252
```

```
vecNR<-aggregate(NoLowMemoryResolutionOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecNR$NoLowMemoryResolutionOriginalDesign)
```

```
## [1] 200
```

```
vecRL<-aggregate(ReleasingResources2LateOriginalDesign ~ appName, dataRF,
                 function(x) min (x))
sum(vecRL$ReleasingResources2LateOriginalDesign)
```

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

```
print("RQ1: To whatextentEARMOcanremove anti-patterns while controlling for energy usage?")
```

```
## [1] "RQ1: To whatextentEARMOcanremove anti-patterns while controlling for energy usage?"
```

```
print("To obtain the min and max values for DAP")
```

```
## [1] "To obtain the min and max values for DAP"
```

```
minDet<-aggregate(dtotalAP ~ appName, dataRF,
                  function(x) min (x))



maxDet<-aggregate(dtotalAP ~ appName, dataRF,
                  function(x) max (x))

minDetEnergy<-aggregate(fitnessEnergy ~ appName, dataRF,
                        function(x) min (x))

maxDetEnergy<-aggregate(fitnessEnergy ~ appName, dataRF,
                        function(x) max (x))

print("Number of solutions")
```

```
## [1] "Number of solutions"
```

```
numOfSols<-aggregate(dtotalAP ~ appName, dataRF,
                     function(x) NROW (x))

summary(numOfSols$dtotalAP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   3.974   5.000  10.000
```

```
mergedDAP<-merge(minDet,maxDet,by="appName")
mergedDAP<-merge(mergedDAP,     minDetEnergy,by="appName")
mergedDAP<-merge(mergedDAP,     maxDetEnergy,by="appName")
mergedDAP<-merge(mergedDAP,     numOfSols,by="appName")

write.table(mergedDAP, file = "mergedDAPECNDdet.csv", sep = ","
            , col.names =T,qmethod = "double",row.names = F)




print("5.1.1 Impact of refactoring sequences with respect to the type of anti-patterns")
```

```
## [1] "5.1.1 Impact of refactoring sequences with respect to the type of anti-patterns"
```

```
dataRF <- transform(dataRF, dRB=RefusedParentBequestOriginalDesign-RefusedParentBequest)
dataRF <- transform(dataRF, dLP=LongParameterListOriginalDesign-LongParameterList)
dataRF <- transform(dataRF, dSC=SpaghettiCodeOriginalDesign-SpaghettiCode)
dataRF <- transform(dataRF, dBL=BlobOriginalDesign-Blob)
dataRF <- transform(dataRF, dSG=SpeculativeGeneralityOriginalDesign-SpeculativeGenerality)
dataRF <- transform(dataRF, dBE=BindingResources2EarlyOriginalDesign-BindingResources2Early)
dataRF <- transform(dataRF, dRL=ReleasingResources2LateOriginalDesign-ReleasingResources2Late)
dataRF <- transform(dataRF, dNLMR=NoLowMemoryResolutionOriginalDesign-NoLowMemoryResolution)
dataRF <- transform(dataRF, dLC=LazyClassOriginalDesign-LazyClass)


summary(dataRF$dRB)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -2.00    1.00   11.00   15.73   26.50   63.00
```

```
summary(dataRF$dLP)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0    15.0    29.0    35.5    42.0   142.0
```

```
summary(dataRF$dSC)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -2.000   0.000   0.000   1.323   2.000   8.000
```

```
summary(dataRF$ dLCLC)
```

```
## Length  Class   Mode
##      0   NULL   NULL
```

```
summary(dataRF$dBL)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -13.00   -1.00    2.00    5.29   10.00   32.00
```

```
summary(dataRF$dSG)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -4.000   0.000   1.000   4.697   5.500  29.000
```

```
summary(dataRF$dBE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

```r
summary(dataRF$dRL)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.03226 0.00000 1.00000
```

```r
summary(dataRF$dNLMR)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   3.000   4.000   4.594   6.000  15.000
```

```r
summary(dataRF$dLC)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -57.00    2.00   14.00   11.31   27.00   71.00
```

```r
print ("5.1.2 Performance of the metaheuristics employed")
```

```
## [1] "5.1.2 Performance of the metaheuristics employed"
```

```r
print ("The statistics were computed using JMetal Framework")
```

```
## [1] "The statistics were computed using JMetal Framework"
```

```r
print ("PDF of the HIV and Spread are included in the same  the same zip of this file.")
```

```
## [1] "PDF of the HIV and Spread are included in the same  the same zip of this file."
```

```r
print("RQ2: To what extent is design quality improved after applying energy-aware refactoring?")
```

```
## [1] "RQ2: To what extent is design quality improved after applying energy-aware refactoring?"
```

```r
dataRF <- transform(dataRF, dReus=100 * (1-(Reusability/ReusabilityOriginalDesign) ) )
dataRF <- transform(dataRF, dUnde=100 * (1-(Understandability/UnderstandabilityOriginalDesign)))
dataRF <- transform(dataRF, dFlex=100 * (1-(Flexibility/FlexibilityOriginalDesign)))
dataRF <- transform(dataRF, dEffe=100 * (1-(Effectiveness/EffectivenessOriginalDesign)))
dataRF <- transform(dataRF, dExte=100 * (1-(Extendibility/ExtendibilityOriginalDesign)))

print("The detailed table writing just one file using merge")
```

```
## [1] "The detailed table writing just one file using merge"
```

```r
minDetReus<-aggregate(dReus ~ appName, dataRF,
                      function(x) min (x))

maxDetReus<-aggregate(dReus ~ appName, dataRF,
                      function(x) max (x))
```

```
minDetUnde<-aggregate(dUnde ~ appName, dataRF,
                      function(x) min (x))

maxDetUnde<-aggregate(dUnde ~ appName, dataRF,
                      function(x) max (x))

minDetFlex<-aggregate(dFlex ~ appName, dataRF,
                      function(x) min (x))

maxDetFlex<-aggregate(dFlex ~ appName, dataRF,
                      function(x) max (x))

minDetEffe<-aggregate(dEffe ~ appName, dataRF,
                      function(x) min (x))

maxDetEffe<-aggregate(dEffe ~ appName, dataRF,
                      function(x) max (x))

minDetExte<-aggregate(dExte ~ appName, dataRF,
                      function(x) min (x))

maxDetExte<-aggregate(dExte ~ appName, dataRF,
                      function(x) max (x))


mergedOnes<-merge(minDetReus,     maxDetReus,by="appName")
mergedOnes<-merge(mergedOnes,   minDetUnde,by="appName")
mergedOnes<-merge(mergedOnes,   maxDetUnde,by="appName")
mergedOnes<-merge(mergedOnes,   minDetFlex,by="appName")
mergedOnes<-merge(mergedOnes,   maxDetFlex,by="appName")
mergedOnes<-merge(mergedOnes,   minDetEffe,by="appName")
mergedOnes<-merge(mergedOnes,   maxDetEffe,by="appName")
mergedOnes<-merge(mergedOnes,   minDetExte,by="appName")
mergedOnes<-merge(mergedOnes,   maxDetExte,by="appName")


write.table(mergedOnes, file = "mergedQMOODdet.csv", sep = ",",
               col.names =T,qmethod = "double",row.names = F)


###END
```