Skeleton tutorial

Raffaele A Calogero

Dissecting the skeleton.R

The skeleton function allows to control a bash script, **skeleton.sh**, located in docker.io/repbioinfo/ubuntu image in /bin.

The skeleton function has three parameters:

```
skeleton(group="docker", scratch.folder, data.folder)
```

- **group**, a character string. Two options: *sudo* or *docker*, depending to which group the user belongs
- **scratch.folder**, a character string indicating the path of the scratch folder. In principle the scratch folder is a temporary folder located in a disk with high I/O, e.g. a SSD disk, but if the fast disk is not available it represents only a temporary folder where the data generated by the application are saved.
- **data.folder**, a character string indicating the folder where input data are located and where output will be written

The first step in the skeleton function is storing the working folder and grabbing the process time for subsequent performance evaluation.

```
  #storing the position of the home folder
  home <- getwd()
  #running time 1
  ptm <- proc.time()
```

Then, it is tested if docker demon is running,

```
  #testing if docker is running
  test <- dockerTest()
  if(!test){
    cat("\nERROR: Docker seems not to be installed in your system\n")
    return()
  }
```

checking if data folder exists and setting it as working folder,

```
  #setting the data.folder as working folder
  if (!file.exists(data.folder)){
    cat(paste("\nIt seems that the ",data.folder, " folder does not exist\n"))
    return(2)
  }
  setwd(data.folder)
```

checking if scratch folder exists and creating a temporary folder.

```
  #check  if scratch folder exist
  if (!file.exists(scratch.folder)){
    cat(paste("\nIt seems that the ",scratch.folder, " folder does not exist\n"))
    return(3)
  }
  tmp.folder <- gsub(":","-",gsub(" ","-",date()))
  scrat_tmp.folder=file.path(scratch.folder, tmp.folder)
  writeLines(scrat_tmp.folder,paste(data.folder,"/tempFolderID", sep=""))
```

```
  cat("\ncreating a folder in scratch folder\n")
  dir.create(file.path(scrat_tmp.folder))
```

Executing the docker command:

- first is created a parameter string, which is made of:
  - –cidfile creates in data.folder the dockerID file that contains the docker job ID
  - -v the temporary folder, created in the scratch folder, is mounted as /scratch
  - -v the data folder is mounted as /data
  - -d docker.io/repbioinfo/ubuntu sh /bin/skeleton.sh is the command that executes the skeleton.sh script.
- the parameter string is passed to the runDocker that execute the job. runDocker check if docker is running and return **false** when is finished

```
  #executing the docker job
  if(group=="sudo"){
    params <- paste("--cidfile ",data.folder,"/dockerID -v ",scrat_tmp.folder,":/scratch -v ", data.fol
    resultRun <- runDocker(group="sudo",container="docker.io/repbioinfo/ubuntu", params=params)
  }else{
    params <- paste("--cidfile ",data.folder,"/dockerID -v ",scrat_tmp.folder,":/scratch -v ", data.fol
    resultRun <- runDocker(group="docker",container="docker.io/repbioinfo/ubuntu", params=params)
  }
```

The **skeleton.sh** scripts in docker.io/repbioinfo/ubuntu is the following:

```
#!/bin/bash
echo "skeleton 0.0.1"
#setting the scratch folder as workinng directory
SCRATCH_FOLDER=/scratch
DATA_FOLDER=/data
#moving to scratch folder
cd $SCRATCH_FOLDER
#adding information to run.info file or creating a run.info file
file="run.info"
if [ -f "$file" ]
then
        echo "skeleton 0.0.1" >> $SCRATCH_FOLDER/run.info
else
        echo "skeleton 0.0.1" > $SCRATCH_FOLDER/run.info
fi
#writing the result file helloworld in data scratch
echo "hello world" > $SCRATCH_FOLDER/helloworld.txt
# creating the out.info file indicating that run is finished
echo "analysis is finished" > $SCRATCH_FOLDER/out.info
#changing the properties of files and folders in /data/scratch
chmod 777 -R $SCRATCH_FOLDER/*
```

It writes hello world in the **helloworld.txt** and moves **helloworld.txt** to the data folder together with the **run.info** file, used to store information about the run, and the **out.info**, used to tell to the R script when the doker job is finished. The **skeleton.sh** scripts is a prototype for the handling of docker application(s).

Lets go back to the skeleton.R dissection:

The **resultRun** is used to check when the docker job is finished. The log of the docker job is saved with a

name made of the first 12 letters of the docker job ID. Then, the docker container is deleted as well as the temporary folder and few other files: out.info, dockerID, tempFolderID. Finally the home folder is restored as working directory.

```
#when container ends
if(resultRun=="false"){
  #everything is copied to the input folder
   system(paste("mv ", scrat_tmp.folder,"/* ",data.folder, sep=""))
    #saving log and removing docker container
   container.id <- readLines(paste(data.folder,"/dockerID", sep=""), warn = FALSE)
   system(paste("docker logs ", substr(container.id,1,12), " &> ", substr(container.id,1,12),".log", s
   system(paste("docker rm ", container.id, sep=""))
   #removing temporary folder
   cat("\n\nRemoving the temporary file ....\n")
   system(paste("rm -R ",scrat_tmp.folder))
   system("rm -fR out.info")
   system("rm -fR dockerID")
   system("rm  -fR tempFolderID")
   system(paste("cp ",paste(path.package(package="docker4seq"),"containers/containers.txt",sep="/")," 
}
```

Then, the computing time is estimated and saved in the run.info file

```
 #running time 2
 ptm <- proc.time() - ptm
 dir <- dir(data.folder)
 dir <- dir[grep("run.info",dir)]
 if(length(dir)>0){
   con <- file("run.info", "r")
   tmp.run <- readLines(con)
   close(con)
   tmp.run[length(tmp.run)+1] <- paste("user run time mins ",ptm[1]/60, sep="")
   tmp.run[length(tmp.run)+1] <- paste("system run time mins ",ptm[2]/60, sep="")
   tmp.run[length(tmp.run)+1] <- paste("elapsed run time mins ",ptm[3]/60, sep="")
   writeLines(tmp.run,"run.info")
 }else{
   tmp.run <- NULL
   tmp.run[1] <- paste("run time mins ",ptm[1]/60, sep="")
   tmp.run[length(tmp.run)+1] <- paste("system run time mins ",ptm[2]/60, sep="")
   tmp.run[length(tmp.run)+1] <- paste("elapsed run time mins ",ptm[3]/60, sep="")
   writeLines(tmp.run,"run.info")
 }
```

```
 setwd(home)
```