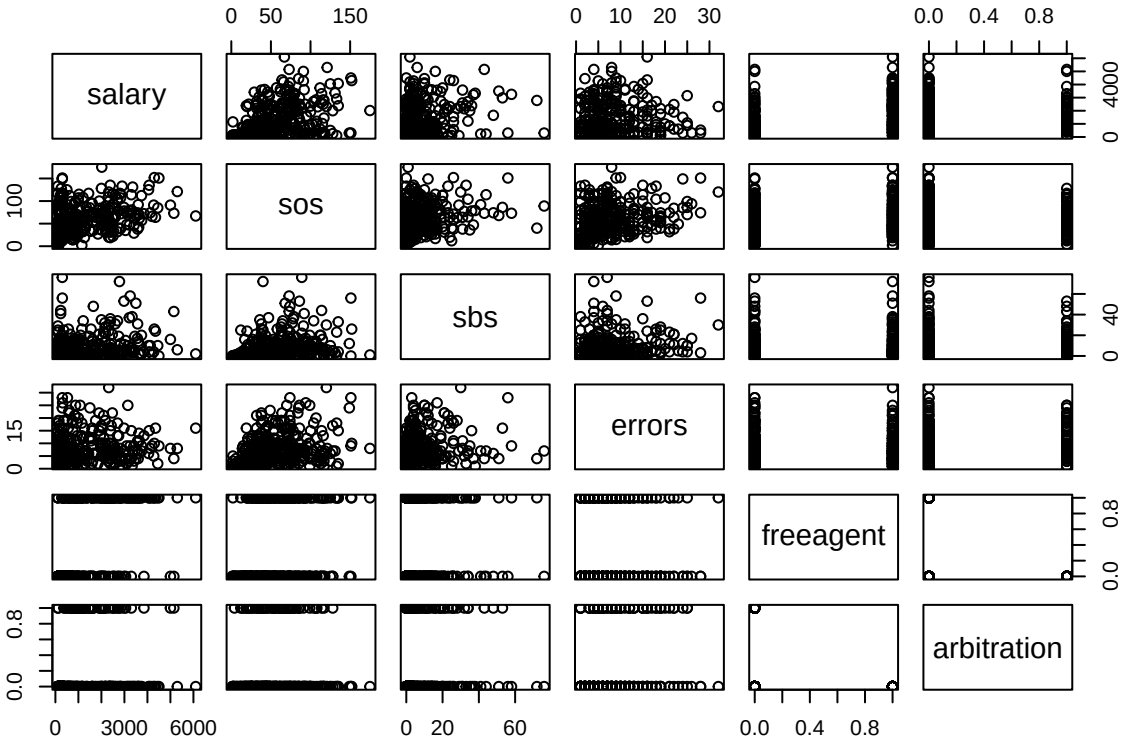
Computational Statistics. Chapter 2: Combinatorial optimization. Solution of exercises

Thierry Denoeux

9/5/2021

Question 1

```
baseball <- read.table("/Users/Thierry/Documents/R/Data/Compstat/baseball.dat",header=TRUE)
attach(baseball)
plot(baseball[,c(1,11:15)])
```



We notice that we cannot plot more that 5-6 variables in a matrix plot.

Question 2

Question 2a

We use once again the very useful function `sample`. The following functions simulates p random draws from the set $\{0, 1\}$ with replacement:

```
initialize<-function(p) theta0<-sample(c(0,1),p,replace=TRUE)
```

Question 2b

In the following function, we treat separately the case where all predictors have been removed (i.e., all elements of vector `theta` are equal to zero), in which case we run the linear regression with the intercept only:

```
aic <-function(theta,data){
  if(all(theta==0)){  # if all predictors have been removed, we have ony the intercept
    crit<-AIC(lm(salary~0,data=data))} else{
    crit<-AIC(lm(salary~.,data=data[,c(1,which(theta==1)+1)]))}
  return(crit)
}
```

Question 2c

The following function `neighborhood` generates a matrix of size $p \times p$ containing the p neighbors of the current solution encoded in vector `theta0`:

```
neighborhood <- function(theta0){
    p<-length(theta0)
    Theta<-matrix(theta0,p,p,byrow=TRUE)
  diag(Theta)<-1-diag(Theta)
    return(Theta)
}
```

Question 2d

This is the main function. The arguments are: `fun` (the function to be optimized), `neighbor` (the function that computes the neighborhood of the current solution), the initial solution `theta0`, the maximum number `N` of iterations, the dataset `data` passed to function `fun`, and an argument `disp` with default value `TRUE`; if the argument has the value `FALSE`, intermediate results are not printed. The output is a list with two elements: the minimum of the objective function (`objective`) and the corresponding optimum value of the parameter (`optimum`).

```
local_search <- function(fun,neighbor,theta0,N=1000,data,disp=TRUE){
    p<-length(theta0)
    obj0<-fun(theta0,data)
    go_on<-TRUE
    t <- 0
    while ((t<N) & go_on){
        t<-t+1
        Theta<-neighbor(theta0)
        Obj <- apply(Theta,1,fun,data)
    i_best<-which.min(Obj)
    obj<-Obj[i_best]
      if (obj>=obj0){   # solution has not improved
        go_on <- FALSE # stop
        }
      else{   # solution has improved
        theta0<-Theta[i_best,]
        obj0<-obj
```

```
        }
      if(disp) print(c(t,obj0))
      } # end while
    return(list(objective=obj0,optimum=theta0))
    }
```

Let us now use this function with a random search strategy we run it 50 times from 50 random starting points, and we keep the best solution:

```
p<-ncol(baseball)-1
M<-50
AICbest<-Inf
for(i in 1:M){
  theta0<-initialize(p)
  opt<-local_search(aic,neighborhood,theta0,100,baseball,disp=FALSE)
  if(opt$objective<AICbest){
    opt_best<-opt
    AICbest<-opt$objective
  }
}
```

We print the AC value and names of the predictors for the best model:

```
print(opt_best$objective)
```

```
## [1] 5375.362
```

```
Names<-names(baseball)
var_best<-which(opt_best$optimum==1)
print(Names[var_best+1])
```

```
## [1] "homeruns"    "rbis"        "walks"       "sos"         "freeagent"
## [6] "arbitration" "walksperso"  "sbsobp"
```

Question 3

Question 3a

This function returns one randomly selected element in the neighborhood of `theta0`:

```
new <- function(theta0){
    p<- length(theta0)
    i<-sample(p,1)
    theta <- theta0
  theta[i]<-1-theta[i]
  return(theta)
}
```

Question 3b

The arguments of function `simulated_annealing` below are: `fun` (the function to be optimized), `new` (the function that returns a random candidate solution), the initial solution `theta0`, the initial temperature `tau0`, the initial stage length `m0`, parameters `a` and `b` of the cooling schedule, the minimum temperature `taumin` used as a stopping criterion, the dataset `data` passed to function `fun`, and the trace flag `disp`. The output is