Structured programming exercises: basic functions.

Paolo Francesco Griffo

Costruire una funzione in grado di indicare il numero di colonne di una matrice (parametro in ingresso alla funzione) contenenti valori missing (ovvero NA)

```
mis.col=function(a){
  if(!is.matrix(a))
    stop("il parametro in ingresso non Ã¨ una matrice")


ncol=dim(a)[2]
c=0
for( j in 1:ncol){
  if (sum(is.na(a[,j]))>0) c=c+1
}
print(paste("numero di colonne con dati mancanti:",c))
}
```

Genero una matrice con dati mancanti per testare la funzione

```
c=(seq(from = 1,to=40, by=2))
d=rep(NA,20)
(e=c(c,d))
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 NA NA NA
## [24] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

```
(matrix.prova=matrix(e,nrow = 4,ncol = 10,byrow = FALSE))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    9   17   25   33   NA   NA   NA   NA    NA
## [2,]    3   11   19   27   35   NA   NA   NA   NA    NA
## [3,]    5   13   21   29   37   NA   NA   NA   NA    NA
## [4,]    7   15   23   31   39   NA   NA   NA   NA    NA
```

```
mis.col(matrix.prova)
```

```
## [1] "numero di colonne con dati mancanti: 5"
```

Genero una matrice con dati mancanti in modo differente

```
matrix.prova2=matrix(1:50,nrow = 5)
matrix.prova2[lower.tri(matrix.prova2)]=NA
matrix.prova2
```

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

```r
mis.col(matrix.prova2)
```

```
## [1] "numero di colonne con dati mancanti: 4"
```

Funzione che conta i missing values in una matrice

```r
countmissing = function(matrix){

  if(!is.matrix(matrix))
    stop("Format non correct")

  ncol = dim(matrix)[2]
  nrow = dim(matrix)[1]
  nmiss = 0

  for( j in 1:ncol){
    for(i in 1:nrow)

      if(sum(is.na(matrix[i,j])) > 0) nmiss = nmiss + 1

  }
    print(paste("N° missing value:", nmiss, sep=""))

      }
```

```r
countmissing(matrix.prova2)
```

```
## [1] "N° missing value:10"
```

Funzione che imputa ai dati mancanti di un vettore la media del vettore.

Indicare il numero di dati mancanti, la media del vettore ed il vettore di dati completo.

```r
AAA = function(x) {

  if(!is.vector(x))
    stop("format not correct")

  pos = which(is.na(x))
  mm = mean(x[-pos])
  x[pos] = mm


  print(paste("N° missing values:", length(pos), sep=""))
  print(paste("Mean of the vector:", mm, sep = ""))

  return(x)

}
```

```r
missingvalue = c(seq(1,100,4), rep(NA,33))
AAA(missingvalue)
```

```
## [1] "N° missing values:33"
## [1] "Mean of the vector:49"
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89
## [24] 93 97 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49
## [47] 49 49 49 49 49 49 49 49 49 49 49 49
```

Si sommino le differenze tra il massimo di un vettore ed ogni suo elemento.

```
somma = function(vettore){
  summae = 0
   for( j in vettore){
    summae = (max(vettore)-vettore[j])+summae
  }

  print(summae)
}
```

```
vettore = c ( 1: 11 )
somma(vettore)
```

```
## [1] 55
```

Esercizio: ordinare un vettore secondo questa procedura: ogni coppia di elementi nel vettore viene comparata e se essi sono nell'ordine sbagliato vengono invertiti. L'algoritmo scorre tutto il vettore finché non vengono più eseguiti scambi, situazione che indica che il vettore é ordinato.

```
my.sort=function(a){
  if(!is.vector(a))
    stop("il parametro in ingresso non Ã¨ un vettore")

  nn=length(a)
  for(i in 1:(nn-1))
    for(j in (i+1):nn)
      if(a[i]>a[j]){
        aux=a[i]
        a[i]=a[j]
        a[j]=aux
      }
  return(a)
}
```

```
v=sample(1:10)
v
```

```
##  [1]  8 10  5  1  2  7  4  3  6  9
```

```
my.sort(v)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

La funzione my.sort() produce lo stesso risultato della funzione interna sort() con maggiore efficienza

```
v=sample(5000)
t0=Sys.time()
```

```r
v1=sort(v)
Sys.time()-t0
```

```
## Time difference of 0.002000093 secs
```

```r
t0=Sys.time()
v1=my.sort(v)
Sys.time()-t0
```

```
## Time difference of 1.357065 secs
```

Bubble Sort (Original)

```r
bubblesort = function(a){
  n = lenght(a)
  swap=1
  while(swap==1){
    swap = 0
    for (i in (1:(n-1))){
      if (a[i] > a[i+1]){
        abackup = a[i]
        a[i+1] = abackup
        swap = 1
      }
    }
  }
return(a)
}
```

Estrarre i numeri pari e dispari da un vettore

```r
numeripari = function(sample){

  if(!is.vector(sample))
    stop("formato non corretto")


 bin =(sample%%2)
 pari= which(bin==0)
 pari = sample[pari]

 dispari = which(bin==1)
 dispari = sample[dispari]


  print(paste("Numero di elementi pari:",length(pari),sep=""))
  print(paste("Numero di elementi dispari:", length(dispari),sep=""))

  cat("Numeri pari :", pari, "\n")
  cat("Numeri dispari :", dispari, "\n")

}
```

```r
prova = sample(1:10,10,replace=TRUE)
numeripari(prova)
```

```
## [1] "Numero di elementi pari:6"
## [1] "Numero di elementi dispari:4"
## Numeri pari : 10 10 10 2 4 10
## Numeri dispari : 3 7 1 1
```

Ricerca binaria: dato un vettore ordinato cominciare la ricerca dal centro se il numero centrale è maggiore di quello ricercato allora spostare la ricerca a sinistra se il numero centrale è minore di quello ricercato spostare la ricerca a destra se le posizioni finiscono e il vettore non è stato trovato indicare che non è stato trovato.

```r
ricbin = function(v,a){

  if(!is.vector(a)) stop("first input parameter must be a vector")
  if( sum(v == sort(v)) < length(v) ) stop("vector must be ordered")

  pos = 0
  flag = 0
  while (flag == 0){
    meta = ceiling( length(v)/2 )
  if(v[meta]==a) {
    print(paste("trovato in posizione: ",pos+meta))
    flag = 1
  }
  else if(meta == 1){
    print("non trovato")
    flag = 1
  }
  else if( v[meta] > a ) v = v[1:(meta-1)]
  else{
    pos = pos + meta
    v = v[(meta + 1):length(v)]
  }

  }
}
```

```r
ordered=c(sort(sample(1:90,90)))

ricbin(ordered,9)
```

```
## [1] "trovato in posizione:  9"
```

Estrarre una schedina da 6 numeri. Estrarre n schedine e verificare quante di queste risultino avere elementi in comune con la prima.

```r
bet = function(n){

  if(n == 0) stop("Numero di schedine insufficienti.")
```

```
  schedine = list()
  nn = rep(1,n)
  win = sample(1:90,6)
  vittorie = c()
  perdenti = c()

  for( i in 1:length(nn)){
    schedine[[i]] = sample(1:90,6)
  }

  for(i in 1:length(schedine)){

    if( sum((schedine[[i]]) == win) == 2){
      vittorie[i] = i
      cat("Ambo vincente: ", schedine[[i]],"\n")
    }
    else if( sum((schedine[[i]]) == win) == 3){
      vittorie[i] = i
      cat("Terna vincente: ", schedine[[i]],"\n")
    }
    else if( sum((schedine[[i]]) == win) == 4){
      vittorie[i] = i
      cat("Quaterna vincente: ", schedine[[i]],"\n")
    }
    else if( sum((schedine[[i]]) == win) == 5){
      vittorie[i] = i
      cat("Cinquina vincente: ", schedine[[i]],"\n")
    }
    else if( sum((schedine[[i]]) == win) == 6){
      vittorie[i] = i
      cat("Sestina vincente: ", schedine[[i]],"\n")

    }
    else{
      perdenti[i] = i
    }
  }

  vittorie = which(!is.na(vittorie))
  perdenti = which(!is.na(perdenti))

  print(paste("Schedine risultate vincenti: ", length(schedine[vittorie]),sep=""))
  print(paste("Schedine risultate perdenti: ", length(schedine[perdenti]),sep=""))
  cat("Numeri estratti: ", win,"\n")

}
```

```
bet(10000)
```

```
## Ambo vincente:  5 22 88 63 50 42
## Ambo vincente:  26 22 25 12 30 32
## Ambo vincente:  18 46 8 12 25 19
## Ambo vincente:  75 79 52 48 53 2
## Ambo vincente:  75 25 2 12 57 27
```

```
## Ambo vincente:  41 27 88 12 10 16
## Ambo vincente:  49 22 88 18 30 83
## Ambo vincente:  15 22 88 87 33 69
## Ambo vincente:  72 22 13 9 66 2
## Ambo vincente:  4 27 61 12 39 2
## Ambo vincente:  85 22 88 53 86 44
## Ambo vincente:  51 22 86 7 25 20
## Ambo vincente:  37 22 88 30 57 71
## Ambo vincente:  75 43 88 17 61 41
## Terna vincente:  34 82 88 12 25 53
## Ambo vincente:  79 30 88 12 69 1
## Ambo vincente:  75 29 88 66 11 67
## Ambo vincente:  90 22 32 68 15 2
## Ambo vincente:  63 22 88 56 28 23
## Ambo vincente:  56 45 18 75 25 2
## [1] "Schedine risultate vincenti: 20"
## [1] "Schedine risultate perdenti: 9980"
## Numeri estratti:  75 22 88 12 25 2
```