Since the first little useless-useful R function MixedCases gain a lot of interest in the R community, let us not stop here. 🙂

In this blog-post let’s make another useless function, that would be somehow intriguing to data scientist. Function that can create a dummy data.frame. Again, the fun part is to make the function long and useless, yet still giving it some usefulness.

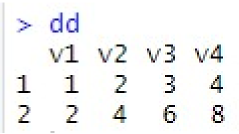
Following is the function to generate randomized dataframe using structure function.

```
DataFrameMaker <- function(col,row){
  command = "dd <- structure(list( "
  for (i in 1:col){
        var = paste("v",as.character(i),"= c(",sep="")
        command = paste(command, var ,sep = "")
        for (j in 1:row){
          a <- c(i*j)
          con = paste(a,sep="")
          if ((j < row) & (j %% row != 0)){
          command = paste(command, con,",",sep = "")
          }
          else {
            command = paste(command, con,"), ",sep = "")
          }
        }

      }
      rn = 'row.names = c('
      for(xx in 1:row){
        rn = paste(rn, xx, 'L,', sep = "")
        if (xx == row){rn = paste(substr(rn,1,nchar(rn)-1),')', sep = "")}
      }

  command <- substr(command, 1, nchar(command)-2)
  command <- paste(command,")",", rn, "," ,"class = 'data.frame')",sep = "")
  print(command)
  eval(parse(text=command))
}
```

Looks pretty long and useless, but it get’s the job done 🙂 When I input the following parameters:

```
DataFrameMaker(4,2)
```

I get the results:



```
> dd
  v1 v2 v3 v4
1  1  2  3  4
2  2  4  6  8
```

In the background the function returns a string that is evaluated in last step of the function. The string holds:

```
"dd <- structure(list( v1= c(1,2), v2= c(2,4), v3= c(3,6), v4=
c(4,8)),row.names = c(1L,2L),class = 'data.frame')"
```

Before you all go ranting about the useless code above and how long it is, here is a shorter version of it:

```
DataFrameMaker  <- function(col,row){
  dd <- matrix(nrow = row, ncol = col)
  for (i in 1:row) {
    for (j in 1:col) {
      dd[i, j] = (j*i)
    }
  }
  return(as.data.frame(dd))
}
```

That creates the same dataframe when called:

```
# Run the dataframe
dd <- DataFrameMaker(4,2)…
```