Creating Functions in R Challenge

Geoffery Mullings

March 10, 2016

```
mymean <- function(Y)

{

  Ybar <- sum(Y) / length(Y)

  return(Ybar)

}

x = c(1,3,10)
mymean(x)

## [1] 4.666667
```

Not surprisingly, the result is the same as the one obtained using the mean function. More generally, a function can take several arguments, but it has to return only one outcome, which could be a list of items. The function we defined above is quite simple and it has several limitations: 1) it does not take into account that the series might have NAs...

```
x = c(1, 3, 10, NA)
mymean(x)

## [1] NA
```

...and 2) it does not calculate the mean of each column in case there are several. As an exercise, modify the mymean function to accomodate for these issues.

```
mymean <- function(Y, na.rm = TRUE)

 {
  N=ncol(Y)
     if (na.rm==TRUE) {Y=na.exclude(Y)} #Remove NAs from the variable

     if (!is.null(N) && length(N) > 1) {for (k in 1:N) #Makes sure the
number of columns isn't none.
       totals <- sum(Y[[k]])/length(Y[[k]])

     Ybar <- sum(totals) / length(totals)

     return(Ybar)}
```

```
     else

     {Ybar <- sum(Y) / length(Y)

     return(Ybar)}

}

mymean(x, na.rm=FALSE)

## [1] NA

mymean(x)

## [1] 4.666667

y = c(1,3,10)

mymean(y)

## [1] 4.666667

mean(y)

## [1] 4.666667

Data = as.data.frame(cbind(c(1, 3, 10, 50), c(4, 7, 10, 1), c(9, 2, 9,
4)))
mymean(Data)

## [1] 36.66667

mean(Data)

## Warning in mean.default(Data): argument is not numeric or logical:
## returning NA

## [1] NA
```

G.M.