R for Beginners - R for Marketing Code File-1

This R code book written by Rohit Dhankar . GitHub - https://github.com/RohitDhankar

Code and Data > https://github.com/RohitDhankar/R-Beginners-Online-Virtual-Learning-Session

Good practice to keep track of current Working Directory , list all Objects in R ENVIRONMENT - specially so when committing changes to Git or any other version control Remote directory.

R for Marketing

```
# Simulating own Synthetic Data for analysis by Marketing Function

# Set Seed -- ensure reproducible results

set.seed(123)

# Presume a retail stores chain called - Mkt , having 200 Stores globally
# Each Country has a store within their capital city
# Do consider this code is NOT DRY :)

ms_ids <- 18801:19001 # Range to create Dummy Mkt Store ID's

# Scalar Vector Constants - tweak to change DF Dimensions
aa<-1500
bb<-1400
cc<-1100
dd<-2200
ee<-4200
ff<-2500
gg<-1400
hh<-1500
ii<-2200
jj<-2000

# Mkt Stores ID's == ms_ids

ms_cntry1 <- c(rep("IND",aa))
ms_cntry2 <- c(rep("AUS",bb))
ms_cntry3 <- c(rep("NZ",cc))
ms_cntry4 <- c(rep("RUS",dd))
ms_cntry5 <- c(rep("USA",ee))
ms_cntry6 <- c(rep("MEX",ff))
ms_cntry7 <- c(rep("CAN",gg))
ms_cntry8 <- c(rep("BRZ",hh))
ms_cntry9 <- c(rep("SPN",ii))
ms_cntry10 <- c(rep("FRA",jj))
# ms_cntry11 <- c(rep("GER",kk))
# ms_cntry12 <- c(rep("CHN",ll))

#
```

```
ms_cty1 <- c(rep("CTY_1",aa))
ms_cty2 <- c(rep("CTY_2",bb))
ms_cty3 <- c(rep("CTY_3",cc))
ms_cty4 <- c(rep("CTY_4",dd))
ms_cty5 <- c(rep("CTY_5",ee))
ms_cty6 <- c(rep("CTY_6",ff))
ms_cty7 <- c(rep("CTY_7",gg))
ms_cty8 <- c(rep("CTY_8",hh))
ms_cty9 <- c(rep("CTY_9",ii))
ms_cty10 <- c(rep("CTY_10",jj))
# ms_cty11 <- c(rep("CTY_11",kk))
# ms_cty12 <- c(rep("CTY_12",ll))
#
# Using - runif() # runif generates random deviates.
psale_1 <- runif(aa,min=100,max=120) ## How many values Required the - N == aa
psale_2 <- runif(bb,min=15,max=20) ##
psale_3 <- runif(cc,min=25,max=30) ##
psale_4 <- runif(dd,min=100,max=320) ##
psale_5 <- runif(ee,min=5,max=140) ##
psale_6 <- runif(ff,min=25,max=350) ##
psale_7 <- runif(gg,min=100,max=620) ##
psale_8 <- runif(hh,min=5,max=80) ##
psale_9 <- runif(ii,min=25,max=90) ##
psale_10 <- runif(jj,min=100,max=620) ##
# psale_11 <- runif(kk,min=5,max=43) ##
# psale_12 <- runif(ll,min=25,max=39) ##

#
# Using - runif() # runif generates random deviates.
pcost_1 <- runif(aa,min=111.49,max=120.56) ## How many values Required the - N == 5
pcost_2 <- runif(bb,min=65.05,max=100.42) ## Random MINIMUM Value  == 65.05
pcost_3 <- runif(cc,min=500.44,max=3000.78) ## Random MAXIMUM Value == 3000.78

pcost_4 <- runif(dd,min=300.44,max=3000.78) ##
pcost_5 <- runif(ee,min=400.44,max=3000.78) ##
pcost_6 <- runif(ff,min=900.44,max=3000.78) ##

pcost_7 <- runif(gg,min=1100.44,max=37000.78) ##
pcost_8 <- runif(hh,min=1400.44,max=32000.78) ##
pcost_9 <- runif(ii,min=1700.44,max=33000.78) ##

pcost_10 <- runif(jj,min=5500.44,max=30000.78) ##
# pcost_11 <- runif(kk,min=3500.44,max=45000.78) ##
# pcost_12 <- runif(ll,min=9900.44,max=13000.78) ##

# Data Frame from NUMERIC and CHARACTER VECTORS
#
# p_sale_count == PRODUCT Sale Count - How many Sold !
#
mdf <- data.frame(cty_name= c(ms_cty1,ms_cty2,ms_cty3,ms_cty4,ms_cty5,ms_cty6,ms_cty7,ms_cty8,ms_cty9,ms
                  country_name= c(ms_cntry1,ms_cntry2,ms_cntry3,ms_cntry4,ms_cntry5,ms_cntry6,ms_cntry7
                  p_sale_count= c(psale_1,psale_2,psale_3,psale_4,psale_5,psale_6,psale_7,psale_8,psale
                  p_sale_cost= c(pcost_1,pcost_2,pcost_3,pcost_4,pcost_5,pcost_6,pcost_7,pcost_8,pcost_
```

```
#
head(mdf)
```

```
##   cty_name country_name p_sale_count p_sale_cost
## 1    CTY_1          IND     105.7516    120.4795
## 2    CTY_1          IND     115.7661    114.2312
## 3    CTY_1          IND     108.1795    115.4242
## 4    CTY_1          IND     117.6603    112.9459
## 5    CTY_1          IND     118.8093    118.9549
## 6    CTY_1          IND     100.9111    113.3774
```

```
#
summary(mdf) # Summary of DF
```

```
##     cty_name     country_name   p_sale_count      p_sale_cost
##  CTY_5  :4200   USA    :4200   Min.   :  5.013   Min.   :   65.13
##  CTY_6  :2500   MEX    :2500   1st Qu.: 40.664   1st Qu.: 1048.47
##  CTY_4  :2200   RUS    :2200   Median :101.825   Median : 2235.46
##  CTY_9  :2200   SPN    :2200   Mean   :142.236   Mean   : 7080.53
##  CTY_10 :2000   FRA    :2000   3rd Qu.:201.950   3rd Qu.:11283.43
##  CTY_1  :1500   BRZ    :1500   Max.   :619.726   Max.   :36939.36
##  (Other):5400   (Other):5400
```

```
#
str(mdf) # Structure of DF
```

```
## 'data.frame':    20000 obs. of  4 variables:
##  $ cty_name    : Factor w/ 10 levels "CTY_1","CTY_10",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ country_name: Factor w/ 10 levels "AUS","BRZ","CAN",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ p_sale_count: num  106 116 108 118 119 ...
##  $ p_sale_cost : num  120 114 115 113 119 ...
```

Speeding up Code

Efficiency Tradeoff —

Will we Multiply TWO Vectors

OR

Will we Multiply TWO DF Column Vectors

There are ceratin text which recommend to Avoid “for Loops” or any other kind of iterations within R Code chunks
At the same time the core dev team at R Studio recommends we need not avoid “for Loops” , thus its best to measure our own codes performance - specially if we want to use it again.

We see below a brief intro to TIMING our code chunks. . . also a brief intro to memory allocation.

Further REFER -
UCLA- https://stats.idre.ucla.edu/r/faq/how-can-i-time-my-code/
Prof . Hadley Wickham - http://adv-r.had.co.nz/memory.html#object-size
Also many other sources from the net.


```
# Firstly lets create and multiply TWO Vectors
```

```
p_sale_count<-c(psale_1,psale_2,psale_3,psale_4,psale_5,psale_6,psale_7,psale_8,psale_9,psale_10)

p_sale_cost<-c(pcost_1,pcost_2,pcost_3,pcost_4,pcost_5,pcost_6,pcost_7,pcost_8,pcost_9,pcost_10)
```

```
# Start the clock!
ptm <- proc.time()

vec_gross_sale <- p_sale_count*p_sale_cost
summary(vec_gross_sale)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1012    58240   242700  1360000   856000 22730000
```

```
proc.time() - ptm
```

```
##    user  system elapsed
##   0.008   0.000   0.009
```

```
#
# As seen below in our case
# ELAPSED time - 1st 0.011 , 2nd - 0.012
# Thus the WALL CLOCK or REAL / ELAPSED
# timings are almost same .
#
# The USER TIME and SYSTEM TIME's in our case
# add upto -
# 1st - 0.008
# 2nd - 0.012

# Thus it would seem we are better off
# with Vector Multiplication

# But we also need to consider
# once we have the "vec_gross_sale"
# we will need to add it to out "mdf"

# Definition of user Time --- The 'user time' is the CPU time
# charged for execution of user instructions of the calling process.
#
# REFER- https://stat.ethz.ch/R-manual/R-devel/library/base/html/proc.time.html
```

```
# Now to multiply TWO Columns of the DF
# Also called COLUMNAR VECTORS

# Again start the clock!
ptm <- proc.time()

mdf$gross_sale<- mdf$p_sale_count*mdf$p_sale_cost

proc.time() - ptm
```

```
##    user  system elapsed
##   0.004   0.000   0.005
```

```
#
str(mdf)
```

```
## 'data.frame':    20000 obs. of  5 variables:
##  $ cty_name    : Factor w/ 10 levels "CTY_1","CTY_10",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ country_name: Factor w/ 10 levels "AUS","BRZ","CAN",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ p_sale_count: num  106 116 108 118 119 ...
##  $ p_sale_cost : num  120 114 115 113 119 ...
##  $ gross_sale  : num  12741 13224 12487 13289 14133 ...
```

```
#
summary(mdf)
```

```
##     cty_name     country_name   p_sale_count      p_sale_cost
##  CTY_5  :4200   USA    :4200   Min.   :  5.013   Min.   :   65.13
##  CTY_6  :2500   MEX    :2500   1st Qu.: 40.664   1st Qu.: 1048.47
##  CTY_4  :2200   RUS    :2200   Median :101.825   Median : 2235.46
##  CTY_9  :2200   SPN    :2200   Mean   :142.236   Mean   : 7080.53
##  CTY_10 :2000   FRA    :2000   3rd Qu.:201.950   3rd Qu.:11283.43
##  CTY_1  :1500   BRZ    :1500   Max.   :619.726   Max.   :36939.36
##  (Other):5400   (Other):5400
##    gross_sale
##  Min.   :    1012
##  1st Qu.:   58244
##  Median :  242693
##  Mean   : 1359584
##  3rd Qu.:  855994
##  Max.   :22728622
##
```

```
#
```