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

```r
# Simulating Synthetic Data

# 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 :)
# I need to recode this bit keeping in mind the
# DONT REPEAT YOURSELF rule.


# Scalar Vector Constant - tweak to change DF Dimensions
aa<-1500

# Dates on which Data gathered
# we simulate 10 sets of dates
# when aa == 1500 , we get 150 dates in each set
# we then combine these sets into a DATES vector
# assign this DATES vector

dates_aa <-seq(as.Date("2000/1/1"), by = "day", length.out = aa/10)
str(dates_aa)
```

```
##  Date[1:150], format: "2000-01-01" "2000-01-02" "2000-01-03" "2000-01-04" ...
```

```r
date_temp_1<-dates_aa[aa/10]
date_temp_1
```

```
## [1] "2000-05-29"
```

```r
dates_bb <-seq(as.Date(date_temp_1), by = "day", length.out = aa/10)
str(dates_bb) ; date_temp_2<-dates_bb[aa/10] ; date_temp_2
```

```
##  Date[1:150], format: "2000-05-29" "2000-05-30" "2000-05-31" "2000-06-01" ...
```

```
## [1] "2000-10-25"
```

```r
dates_cc <-seq(as.Date(date_temp_2), by = "day", length.out = aa/10)
str(dates_cc) ; date_temp_3<-dates_cc[aa/10] ; date_temp_3
```

```
##  Date[1:150], format: "2000-10-25" "2000-10-26" "2000-10-27" "2000-10-28" ...
## [1] "2001-03-23"
```

```
dates_dd <-seq(as.Date(date_temp_3), by = "day", length.out = aa/10)
str(dates_dd) ; date_temp_4<-dates_dd[aa/10] ; date_temp_4
```

```
##  Date[1:150], format: "2001-03-23" "2001-03-24" "2001-03-25" "2001-03-26" ...

## [1] "2001-08-19"
```

```
dates_ee <-seq(as.Date(date_temp_4), by = "day", length.out = aa/10)
str(dates_ee) ; date_temp_5<-dates_ee[aa/10] ; date_temp_5
```

```
##  Date[1:150], format: "2001-08-19" "2001-08-20" "2001-08-21" "2001-08-22" ...

## [1] "2002-01-15"
```

```
dates_ff <-seq(as.Date(date_temp_5), by = "day", length.out = aa/10)
str(dates_ff) ; date_temp_6<-dates_ff[aa/10] ; date_temp_6
```

```
##  Date[1:150], format: "2002-01-15" "2002-01-16" "2002-01-17" "2002-01-18" ...

## [1] "2002-06-13"
```

```
dates_gg <-seq(as.Date(date_temp_6), by = "day", length.out = aa/10)
str(dates_gg) ; date_temp_7<-dates_gg[aa/10] ; date_temp_7
```

```
##  Date[1:150], format: "2002-06-13" "2002-06-14" "2002-06-15" "2002-06-16" ...

## [1] "2002-11-09"
```

```
dates_hh <-seq(as.Date(date_temp_7), by = "day", length.out = aa/10)
str(dates_hh) ; date_temp_8<-dates_hh[aa/10] ; date_temp_8
```

```
##  Date[1:150], format: "2002-11-09" "2002-11-10" "2002-11-11" "2002-11-12" ...

## [1] "2003-04-07"
```

```
dates_ii <-seq(as.Date(date_temp_8), by = "day", length.out = aa/10)
str(dates_ii) ; date_temp_9<-dates_ii[aa/10] ; date_temp_9
```

```
##  Date[1:150], format: "2003-04-07" "2003-04-08" "2003-04-09" "2003-04-10" ...

## [1] "2003-09-03"
```

```
dates_jj <-seq(as.Date(date_temp_9), by = "day", length.out = aa/10)
str(dates_jj) ; date_temp_10<-dates_jj[aa/10] ; date_temp_10
```

```
##  Date[1:150], format: "2003-09-03" "2003-09-04" "2003-09-05" "2003-09-06" ...

## [1] "2004-01-30"
```

```
# CHECK --- Could i have done this faster in Python ??


# Func - seq(as.Date ...)
# REFER -- https://stat.ethz.ch/R-manual/R-devel/library/base/html/seq.Date.html

# Mkt Stores ID's == ms_ids

ms_cntry1 <- c(rep("IND",aa))
ms_cntry2 <- c(rep("AUS",aa))
ms_cntry3 <- c(rep("NZ",aa))
ms_cntry4 <- c(rep("RUS",aa))
```

```r
ms_cntry5 <- c(rep("USA",aa))
ms_cntry6 <- c(rep("MEX",aa))
ms_cntry7 <- c(rep("CAN",aa))
ms_cntry8 <- c(rep("BRZ",aa))
ms_cntry9 <- c(rep("SPN",aa))
ms_cntry10 <- c(rep("FRA",aa))
#
# #
ms_cty1 <- c(rep("CTY_1",aa))
ms_cty2 <- c(rep("CTY_2",aa))
ms_cty3 <- c(rep("CTY_3",aa))
ms_cty4 <- c(rep("CTY_4",aa))
ms_cty5 <- c(rep("CTY_5",aa))
ms_cty6 <- c(rep("CTY_6",aa))
ms_cty7 <- c(rep("CTY_7",aa))
ms_cty8 <- c(rep("CTY_8",aa))
ms_cty9 <- c(rep("CTY_9",aa))
ms_cty10 <- c(rep("CTY_10",aa))
# #
# #
# # Using - runif() # runif generates random deviates.
psale_1 <- runif(aa,min=100,max=120) ## How many values Required the - N == aa
psale_2 <- runif(aa,min=15,max=20) ##
psale_3 <- runif(aa,min=25,max=30) ##
psale_4 <- runif(aa,min=100,max=320) ##
psale_5 <- runif(aa,min=5,max=140) ##
psale_6 <- runif(aa,min=25,max=350) ##
psale_7 <- runif(aa,min=100,max=620) ##
psale_8 <- runif(aa,min=5,max=80) ##
psale_9 <- runif(aa,min=25,max=90) ##
psale_10 <- runif(aa,min=100,max=620) ##
#
# #
# # 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(aa,min=65.05,max=100.42) ## Random MINIMUM Value  == 65.05
pcost_3 <- runif(aa,min=500.44,max=3000.78) ## Random MAXIMUM Value == 3000.78
pcost_4 <- runif(aa,min=300.44,max=3000.78) ##
pcost_5 <- runif(aa,min=400.44,max=3000.78) ##
pcost_6 <- runif(aa,min=900.44,max=3000.78) ##
pcost_7 <- runif(aa,min=1100.44,max=37000.78) ##
pcost_8 <- runif(aa,min=1400.44,max=32000.78) ##
pcost_9 <- runif(aa,min=1700.44,max=33000.78) ##
pcost_10 <- runif(aa,min=5500.44,max=30000.78) ##
#
# 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_
                  var_dates=rep(dates_aa,10))
# #
head(mdf)
```

```
##   cty_name country_name p_sale_count p_sale_cost  var_dates
## 1    CTY_1          IND     105.7516    117.2661 2000-01-01
## 2    CTY_1          IND     115.7661    112.3257 2000-01-02
## 3    CTY_1          IND     108.1795    118.6762 2000-01-03
## 4    CTY_1          IND     117.6603    120.0371 2000-01-04
## 5    CTY_1          IND     118.8093    114.2354 2000-01-05
## 6    CTY_1          IND     100.9111    116.5965 2000-01-06
```

```
#
length(mdf$cty_name)
```

```
## [1] 15000
```

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

```
##     cty_name     country_name   p_sale_count      p_sale_cost
##  CTY_1  :1500   AUS    :1500   Min.   :  5.042   Min.   :   65.06
##  CTY_10 :1500   BRZ    :1500   1st Qu.: 29.620   1st Qu.:  834.79
##  CTY_2  :1500   CAN    :1500   Median :100.839   Median : 2339.83
##  CTY_3  :1500   FRA    :1500   Mean   :143.938   Mean   : 7763.81
##  CTY_4  :1500   IND    :1500   3rd Qu.:209.595   3rd Qu.:13708.06
##  CTY_5  :1500   MEX    :1500   Max.   :619.898   Max.   :36937.84
##  (Other):6000   (Other):6000
##    var_dates
##  Min.   :2000-01-01
##  1st Qu.:2000-02-07
##  Median :2000-03-15
##  Mean   :2000-03-15
##  3rd Qu.:2000-04-22
##  Max.   :2000-05-29
##
```

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

```
## 'data.frame':    15000 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  117 112 119 120 114 ...
##  $ var_dates   : Date, format: "2000-01-01" "2000-01-02" ...
```

```
# #
```

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.
Rohit Dhankar claims no copyright to any of this code.

```
# 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.
##     1007    35310   248000  1558000  1116000 22480000
```

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

```
##    user  system elapsed
##   0.012   0.000   0.010
```

```
#
# 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"

# Kindly also note the Timings will
# differ for each system - also for each run
# of the chunk of code on same sys

# Definition of user Time --- The ‘user time’ is the CPU time
```

```r
# 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
```

```r
# 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   0.004
```

```r
#
str(mdf)
```

```
## 'data.frame':    15000 obs. of  6 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  117 112 119 120 114 ...
##  $ var_dates   : Date, format: "2000-01-01" "2000-01-02" ...
##  $ gross_sale  : num  12401 13004 12838 14124 13572 ...
```

```r
#
summary(mdf)
```

```
##     cty_name     country_name   p_sale_count      p_sale_cost     
##  CTY_1  :1500   AUS    :1500   Min.   :  5.042   Min.   :   65.06  
##  CTY_10 :1500   BRZ    :1500   1st Qu.: 29.620   1st Qu.:  834.79  
##  CTY_2  :1500   CAN    :1500   Median :100.839   Median : 2339.83  
##  CTY_3  :1500   FRA    :1500   Mean   :143.938   Mean   : 7763.81  
##  CTY_4  :1500   IND    :1500   3rd Qu.:209.595   3rd Qu.:13708.06  
##  CTY_5  :1500   MEX    :1500   Max.   :619.898   Max.   :36937.84  
##  (Other):6000   (Other):6000                                      
##    var_dates            gross_sale      
##  Min.   :2000-01-01   Min.   :    1007  
##  1st Qu.:2000-02-07   1st Qu.:   35315  
##  Median :2000-03-15   Median :  247960  
##  Mean   :2000-03-15   Mean   : 1557641  
##  3rd Qu.:2000-04-22   3rd Qu.: 1116263  
##  Max.   :2000-05-29   Max.   :22481409  
## 
```

```r
#
write.csv(mdf,file="DATA_Files/mdf.csv")
## Writes to Sub Directory - DATA_Files
#
```