Doing Data Science Case Study 2

Ashwin Thota and Kevin OKiah

August 2, 2017

Table of Contents

## Question 01 (10 points)

Create the X matrix and print it from SAS, R, and Python.

#### SAS code

insert screen short of SAS Code and output

#### R code

X = matrix(c(4,5,1,2, 1,1,4,5,2,1,0,2),   
 nrow = 3,   
 ncol =4,  
 byrow =TRUE)  
  
print(X)

## [,1] [,2] [,3] [,4]  
## [1,] 4 5 1 2  
## [2,] 1 1 4 5  
## [3,] 2 1 0 2

#### Python Code

import numpy  
x = numpy.matrix('4,5,1,2; 1,1,4,5;2,1,0,2')  
  
print(x)

## [[4 5 1 2]  
## [1 1 4 5]  
## [2 1 0 2]]

# Question 02 (15 points)

Please watch videos1 and 2 in week 11 lecture assignment. You can download the code which used for S&P from files tab.

Please do the following with your assigned stock.

* Download the data.
* Calculate log returns.
* Calculate volatility measure.
* Calculate volatility over entire length of series for various three different decay factors.
* Plot the results, overlaying the volatility curves on the data, just as was done in the S&P example.

Ashwin and Kevin->XOM

If you cannot download data from your assigned stock you can pick a different stock from the following list. PM, PG, HON, URTY, PEN

You can use Yahoo finance or google finance to find your data! Submit your final code and the plot.

# Question 03 (20 points)

The built-in data set called Orange in R is about the growth of orange trees. The Orange data frame has 3 columns of records of the growth of orange trees.

Variable description Tree : an ordered factor indicating the tree on which the measurement is made. The ordering is according to increasing maximum diameter.

age : a numeric vector giving the age of the tree (days since 1968/12/31) circumference : a numeric vector of trunk circumferences (mm). This is probably  
“circumference at breast height”, a standard measurement in forestry.

#Check the structure of the dataset  
str(Orange)

## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 35 obs. of 3 variables:  
## $ Tree : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...  
## $ age : num 118 484 664 1004 1231 ...  
## $ circumference: num 30 58 87 115 120 142 145 33 69 111 ...  
## - attr(\*, "formula")=Class 'formula' language circumference ~ age | Tree  
## .. ..- attr(\*, ".Environment")=<environment: R\_EmptyEnv>   
## - attr(\*, "labels")=List of 2  
## ..$ x: chr "Time since December 31, 1968"  
## ..$ y: chr "Trunk circumference"  
## - attr(\*, "units")=List of 2  
## ..$ x: chr "(days)"  
## ..$ y: chr "(mm)"

names(Orange)

## [1] "Tree" "age" "circumference"

head(Orange)

## Tree age circumference  
## 1 1 118 30  
## 2 1 484 58  
## 3 1 664 87  
## 4 1 1004 115  
## 5 1 1231 120  
## 6 1 1372 142

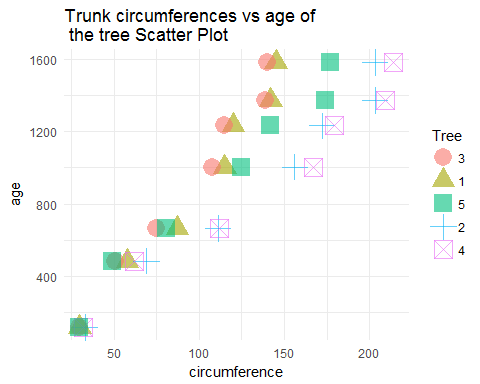
1. Calculate the mean and the median of the trunk circumferences for different size of the trees. (Tree)

pacman::p\_load(pacman, dplyr) #load required packages  
  
suppressMessages(  
 Summary\_<-Orange%>%  
 group\_by(Tree)%>%  
 select(circumference)%>%  
 summarise(  
 Mean = mean(circumference),  
 Median = median(circumference)  
 )  
)  
Summary\_

## # A tibble: 5 × 3  
## Tree Mean Median  
## <ord> <dbl> <dbl>  
## 1 3 94.00000 108  
## 2 1 99.57143 115  
## 3 5 111.14286 125  
## 4 2 135.28571 156  
## 5 4 139.28571 167

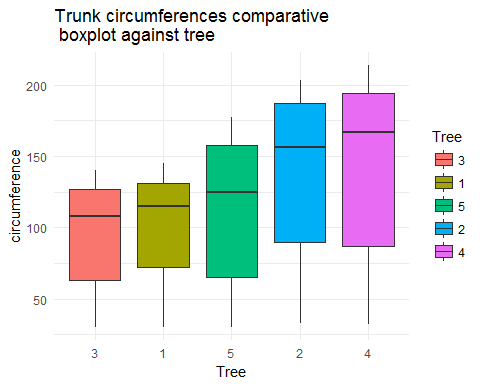
1. Make a scatter plot of the trunk circumferences against the age of the tree. Use different plotting symbols for different size of trees.

pacman::p\_load(ggplot2)#load ggplot2   
  
# Set shape by Tree  
ggplot(Orange, aes(x=circumference, y=age, shape=Tree , color=Tree)) + geom\_point(size=6, alpha=0.6)+  
labs(title="Trunk circumferences vs age of \n the tree Scatter Plot")+ theme\_minimal()



1. Display the trunk circumferences on a comparative boxplot against tree. Be sure you order the boxplots in the increasing order of maximum diameter.

ggplot(data = Orange,aes(y = circumference, x = Tree, fill=Tree)) + geom\_boxplot() +  
labs(title="Trunk circumferences comparative \n boxplot against tree")+ theme\_minimal()



# Question 04 (40 points)

Download “Temp” data set.

Temp<-data.frame((read.csv('TEMP.csv')))  
  
first<-data.frame(as.Date(Temp$Date[1:741]))  
names(first)<-c('Formated\_Date')  
second<-data.frame(as.Date(Temp$Date[742:length(Temp$Date)], format = "%d/%m/%Y"))  
names(second)<-c('Formated\_Date')  
Temp$Formated\_Date =rbind(first, second)

1. Find the difference between the maximum and the minimum monthly average temperatures for each country and report/visualize top 20 countries with the maximum differences for the period since 1900.

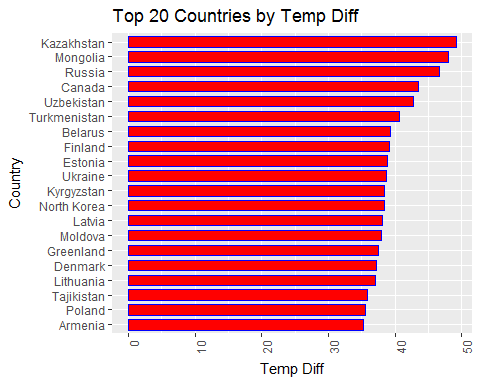
New\_Temp<-na.omit(Temp[which(Temp$Formated\_Date>='1900/1/1'),]) # subset data with start date >1900  
Temp\_summary<-New\_Temp%>%  
 group\_by(Country)%>%  
 select(Monthly.AverageTemp)%>%  
 summarise(  
 MAX = max(Monthly.AverageTemp),  
 MIN = min(Monthly.AverageTemp),  
 Diff = MAX-MIN  
 )

## Adding missing grouping variables: `Country`

#order by difference in decreasing order  
newdata <- Temp\_summary[order(-Temp\_summary$Diff),]   
# print top 20 difference   
head(newdata, 20)

## # A tibble: 20 × 4  
## Country MAX MIN Diff  
## <fctr> <dbl> <dbl> <dbl>  
## 1 Kazakhstan 25.562 -23.601 49.163  
## 2 Mongolia 20.716 -27.294 48.010  
## 3 Russia 16.893 -29.789 46.682  
## 4 Canada 14.796 -28.736 43.532  
## 5 Uzbekistan 30.375 -12.323 42.698  
## 6 Turkmenistan 32.136 -8.443 40.579  
## 7 Belarus 22.811 -16.527 39.338  
## 8 Finland 18.967 -20.101 39.068  
## 9 Estonia 22.332 -16.483 38.815  
## 10 Ukraine 23.936 -14.724 38.660  
## 11 Kyrgyzstan 19.275 -19.161 38.436  
## 12 North Korea 23.952 -14.390 38.342  
## 13 Latvia 22.279 -15.784 38.063  
## 14 Moldova 25.231 -12.781 38.012  
## 15 Greenland 0.339 -37.177 37.516  
## 16 Denmark 0.699 -36.439 37.138  
## 17 Lithuania 21.791 -15.179 36.970  
## 18 Tajikistan 19.363 -16.466 35.829  
## 19 Poland 22.509 -13.107 35.616  
## 20 Armenia 25.291 -9.982 35.273

#plotting the data  
TempN<-head(newdata, 20)  
#plot bar char for the top 20 City Temps  
i<-ggplot(data=TempN, aes(y=Diff, reorder(Country, Diff))) +  
 geom\_bar(stat="identity", color="blue", fill="red",width=.7)+ theme(axis.text.x = element\_text(angle = 90, hjust = 1))+labs(title="Top 20 Countries by Temp Diff", y="Temp Diff", x = "Country")+ coord\_flip()  
i



1. Select a subset of data called “UStemp” where US land temperatures from 01/01/1990 in Temp data. Use UStemp dataset to answer the followings.

UStemp<-na.omit(Temp[which(Temp$Formated\_Date>='1900/1/1'),])

1. Create a new column to display the monthly average land temperatures in Fahrenheit (°F).

UStemp$Fahrenheit<-(UStemp$Monthly.AverageTemp\*(180/100))+32 #create a fahrenheit   
  
#print first 6 rows  
head(UStemp)

## Date Monthly.AverageTemp Monthly.AverageTemp.Uncertainty  
## 742 1/1/1900 -3.428 0.936  
## 743 2/1/1900 1.234 1.135  
## 744 3/1/1900 10.545 0.933  
## 745 4/1/1900 13.352 0.536  
## 746 5/1/1900 20.260 0.524  
## 747 6/1/1900 24.448 0.944  
## Country Formated\_Date Fahrenheit  
## 742 Afghanistan 1900-01-01 25.8296  
## 743 Afghanistan 1900-01-02 34.2212  
## 744 Afghanistan 1900-01-03 50.9810  
## 745 Afghanistan 1900-01-04 56.0336  
## 746 Afghanistan 1900-01-05 68.4680  
## 747 Afghanistan 1900-01-06 76.0064

1. Calculate average land temperature by year and plot it. The original file has the average land temperature by month.

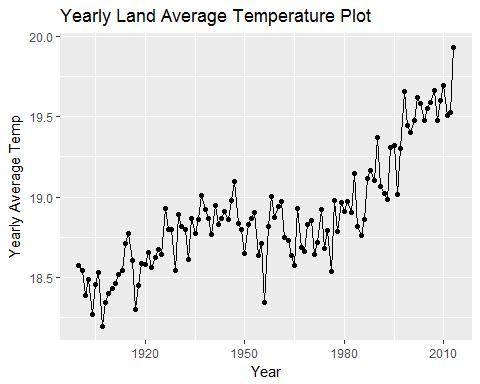
UStemp<-(Temp[which(Temp$Formated\_Date>='1900/1/1'),])# raw data  
  
Year<-format((UStemp$Formated\_Date), format = "%Y")# create a new Year column based on Formated Date  
names(Year)<-c('Year')  
UStemp<-cbind(UStemp,Year) #bind the new created column  
  
UStemp<-na.omit(UStemp) #Omit Nas  
#Yearly Land Temp  
Yearly\_Land\_Avg<-UStemp%>%  
 group\_by(Year)%>%  
 select(Monthly.AverageTemp)%>%  
 summarise(  
 avg = mean(as.numeric(Monthly.AverageTemp)))

## Adding missing grouping variables: `Year`

Yearly\_Land\_Avg

## # A tibble: 114 × 2  
## Year avg  
## <S3: AsIs> <dbl>  
## 1 1900 18.57393  
## 2 1901 18.54531  
## 3 1902 18.39099  
## 4 1903 18.48791  
## 5 1904 18.26810  
## 6 1905 18.45356  
## 7 1906 18.53389  
## 8 1907 18.19317  
## 9 1908 18.34625  
## 10 1909 18.40127  
## # ... with 104 more rows

ggplot(Yearly\_Land\_Avg, aes(x=as.numeric(Year), y=as.numeric(avg))) +geom\_line()+geom\_point() +labs(title="Yearly Land Average Temperature Plot", y = 'Yearly Average Temp', x = 'Year')



1. Calculate the one year difference of average land temperature by year and provide the maximum difference (value) with corresponding years. (for example, year 2000: add all 12 monthly averages and divide by 12 to get average temperature in 2000. You can do the same thing for all the available years. Then you can calculate the one year difference as 1991-1990, 1992-1991, etc)

new<-transform(Yearly\_Land\_Avg, One\_Year\_Difference = ave(avg, FUN = function(x) c(NA, diff(x))))  
  
#print first 6 rows to confirm calculations  
head(new)

## Year avg One\_Year\_Difference  
## 1 1900 18.57393 NA  
## 2 1901 18.54531 -0.02861817  
## 3 1902 18.39099 -0.15431941  
## 4 1903 18.48791 0.09691749  
## 5 1904 18.26810 -0.21981067  
## 6 1905 18.45356 0.18546757

#order by difference in decreasing order  
newdata <- new[order(-new$One\_Year\_Difference),]   
# print top 20 difference   
  
print('The max difference')

## [1] "The max difference"

head(newdata, 1)

## Year avg One\_Year\_Difference  
## 58 1957 18.81424 0.4675591

Y = head(newdata, 1)  
  
print(paste(' The one Year Max difference is' , round(Y$One\_Year\_Difference, 4),'and the corresponding Years are', Y$Year, '-', (as.numeric(Y$Year)-1)))

## [1] " The one Year Max difference is 0.4676 and the corresponding Years are 1957 - 1956"

1. Download “CityTemp” data set (check your SMU email). Find the difference between the maximum and the minimum temperatures for each major city and report/visualize top 20 cities with maximum differences for the period since 1900.

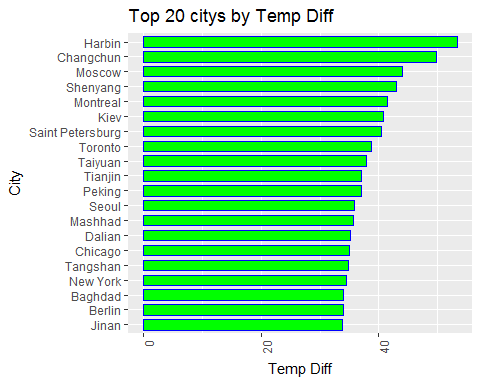
CityTemp<-read.csv('CityTemp.csv')  
first1<-data.frame(as.Date(CityTemp$Date[1:600]))  
names(first1)<-c('Formated\_Date')  
second1<-data.frame(as.Date(CityTemp$Date[601:length(CityTemp$Date)], format = "%d/%m/%Y"))  
names(second1)<-c('Formated\_Date')  
  
CityTemp$Formated\_Date =(rbind(first1, second1))  
  
New\_CityTemp<-na.omit(CityTemp[which(CityTemp$Formated\_Date>='1900/1/1'),]) # subset data with start date >1900  
  
  
City\_Temp<-New\_CityTemp%>%  
 group\_by(Country,City)  
  
CityTemp\_summary = select(City\_Temp,Monthly.AverageTemp)%>%  
 summarise(MAX = max(Monthly.AverageTemp),  
 MIN = min(Monthly.AverageTemp),  
 Diff = MAX-MIN)

## Adding missing grouping variables: `Country`, `City`

#order by difference in decreasing order  
City\_Temp\_Summary\_sort <- CityTemp\_summary[order(-CityTemp\_summary$Diff),]   
# print top 20 diff  
  
head(City\_Temp\_Summary\_sort, 20)

## Source: local data frame [20 x 5]  
## Groups: Country [9]  
##   
## Country City MAX MIN Diff  
## <fctr> <fctr> <dbl> <dbl> <dbl>  
## 1 China Harbin 26.509 -26.772 53.281  
## 2 China Changchun 26.572 -23.272 49.844  
## 3 Russia Moscow 24.580 -19.376 43.956  
## 4 China Shenyang 26.010 -17.035 43.045  
## 5 Canada Montreal 23.059 -18.363 41.422  
## 6 Ukraine Kiev 24.593 -16.191 40.784  
## 7 Russia Saint Petersburg 21.921 -18.589 40.510  
## 8 Canada Toronto 23.181 -15.502 38.683  
## 9 China Taiyuan 24.718 -13.116 37.834  
## 10 China Peking 28.936 -8.017 36.953  
## 11 China Tianjin 28.936 -8.017 36.953  
## 12 South Korea Seoul 26.791 -8.992 35.783  
## 13 Iran Mashhad 27.226 -8.384 35.610  
## 14 China Dalian 25.875 -9.348 35.223  
## 15 United States Chicago 26.372 -8.590 34.962  
## 16 China Tangshan 27.346 -7.487 34.833  
## 17 United States New York 25.313 -9.147 34.460  
## 18 Iraq Baghdad 38.283 4.236 34.047  
## 19 Germany Berlin 23.795 -10.125 33.920  
## 20 China Jinan 28.389 -5.389 33.778

#plotting the data  
CDAT<-head(City\_Temp\_Summary\_sort, 20)  
#plot bar char for the top 20 City Temps  
iii<-ggplot(data=CDAT, aes(y=Diff, reorder(City, Diff))) +  
 geom\_bar(stat="identity", color="blue", fill="green",width=.7)+ theme(axis.text.x = element\_text(angle = 90, hjust = 1))+labs(title="Top 20 citys by Temp Diff", y="Temp Diff", x = "City")+ coord\_flip()  
  
iii



1. Compare the two graphs in (i) and (iii) and comment it.

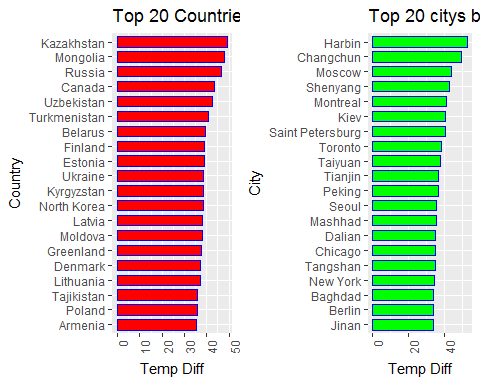
require(gridExtra)

## Loading required package: gridExtra

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

grid.arrange(i, iii, ncol=2)



# Question 05 (15 points)

Write a function in R or Python that converts temperature to either Fahrenheit or Celsius. Your function definition should be as follows –

Inputs:

Temp\_val = Int or list of temperature values to be converted.  
   
 Convert\_to = Str. "F" to convert to Fahrenheit or "C" to convert to Celsius. Raise error   
 if the strings do not match specified input.

Output:

Int or list of converted temperature values with temperature unit.

TempConverter<-function(Temp\_val , convert\_to){  
 #make sure all inputs passed are numeric  
 tryCatch({as.numeric(Temp\_val)  
 },  
 error = function(e) {  
 print("EXCEPTION!! Can only convert numeric inputs")})  
 #only F or C conversions are supported  
 if(convert\_to=='F' || convert\_to=='C'){  
 #from Celsius to Fahrenheit: first multiply by 180/100, then add 32  
 if(toupper(convert\_to)=='F'){  
 print(paste("Converting",paste0(Temp\_val," degree Celsius")," to Fahrenheit"))  
 return(paste0((Temp\_val\*(180/100))+32," degree Fahrenheit"))  
 }  
 #from Fahrenheit to Celsius: first subtract 32, then multiply by 100/180  
 if(toupper(convert\_to)=='C'){  
 print(paste("Converting",paste0(Temp\_val," degree Fahrenheit")," to Celsius"))  
 return(paste0((Temp\_val-32)\*(100/180)," degree Celsius"))  
 }  
 }  
 else{  
 print("EXCEPTION!! Only an 'F'to convert to Fahrenheit or 'C' to convert to Celsius are supported")  
 }  
}  
  
#Testing the Function  
TempConverter((68.5), 'F')

## [1] "Converting 68.5 degree Celsius to Fahrenheit"

## [1] "155.3 degree Fahrenheit"

TempConverter(c(10, 0, 23), 'C')

## [1] "Converting 10 degree Fahrenheit to Celsius"  
## [2] "Converting 0 degree Fahrenheit to Celsius"   
## [3] "Converting 23 degree Fahrenheit to Celsius"

## [1] "-12.2222222222222 degree Celsius" "-17.7777777777778 degree Celsius"  
## [3] "-5 degree Celsius"

TempConverter(c(10, 0, 23), 'F')

## [1] "Converting 10 degree Celsius to Fahrenheit"  
## [2] "Converting 0 degree Celsius to Fahrenheit"   
## [3] "Converting 23 degree Celsius to Fahrenheit"

## [1] "50 degree Fahrenheit" "32 degree Fahrenheit"   
## [3] "73.4 degree Fahrenheit"

TempConverter(32, 'C')

## [1] "Converting 32 degree Fahrenheit to Celsius"

## [1] "0 degree Celsius"

TempConverter(0, 'X')

## [1] "EXCEPTION!! Only an 'F'to convert to Fahrenheit or 'C' to convert to Celsius are supported"