Big Data Analysis with Microsoft R server & Azure Spark (150 Million Rows)

Bharath Bolla and Soujanya Mandalapu
Department of Statistics and Analytical Sciences
Advisors: Dr. Jennifer Priestley, Jie Hao

Kennesaw State University
College of Science and Mathematics
Department of Statistics and Analytical Sciences

Introduction

R is the most popular statistical software. R is not only free but is open source and has rich contribution form users. Over 6000 packages are available and many are being developed. But it has several limitations with respect to processing 'Big Data'. Data flow overwhelms R as it processes data by in-memory operation. R lacks implicit parallelism and has expensive data movement.

One way is to overcome the memory limitation of R is by avoiding the memory altogether and processing the data by chunks. Microsoft R server implements these capabilities with novel High Performance Analytic functions and new file format system called XDF(External Data Format). XDF is a compressed file in binary format. The compression makes the file around eight times smaller in size than CSV. We have implemented these functions in a local computer and also in the Azure cloud computing with a customized cluster.

In this poster we have analyzed the Airline information from 1987 to 2012. This data was downloaded from Bureau of Transportation Statistics and revolutionanalytics.com. This data consists of 148 Million rows and 46 variables.

Procedure

We have presented the data in two parts. The left hand side of the figures section depict the architecture of parallel processing in local computer and in cloud by using Microsoft Azure HDInsight. The right side depicts the exploratory data analysis done on 148 million rows, 56 variables, XDF file of 4gb size. Microsoft R Server (version 8.0) has been installed on Windows 10 Laptop with 12gb RAM and i7quad core processor, we tested the R server with various High Performance Analytic functions and also in Azure Spark HDInsight(Fig_5).

First, we tried to load the files in Cran R with 1, 5, 11, 23, 59 million rows which have a size of 120, 500, 1200, 2300, 6000mb respectively. After failing to load 50 million rows, we used Microsoft R server High Performance Analytic functions which have a prefix 'rx' or 'Rx'. These functions process the data chunk by chunk with each chunk size having default 300,000 rows (Fig_2). We checked the time taken to implement rxSummary on 1, 5, 11, 25, 59 and 148 Million rows for one column (Arr Delay in minutes) on both CSV file and XDF file(Fig_3). We used RxTextData and RxXDFData functions to point as source objects for CSV and XDF files respectively. These functions avoid loading the data into memory altogether and get the data from hard disk chunk by chunk. But there is a cost involved in converting CSV to XDF file (Fg_4).

All the analysis was performed on Airline Data consisting of 148 million rows and 46 variables XDF file of 4gb size. The total flights operated by each carrier from 1987-2012 (Fig_6) and the percentage of flights arrived late by more than 15 minutes for each carrier (Fig_7) were analyzed by rxSummary function. The reason for the delay caused in minutes by various factors like weather, security check, carrier, late aircraft and NAS delay has been summarized and presented as density plots (Fig_8). As we found that NAS delay causes significant delay which was due to congestion at the airports we were interested to find out the average NAS delay for all the airports and how the five busiest airports of U.S fared with the rest (Table_1). A scatter plot on log flights operated at the airport and NAS delay revealed a weak relationship (r2=0.2). From our analysis it was evident that Carrier and Airport(NAS) were the major factors for delay. We were curious to find the association between time of the day and day of the week with arrival and departure delays (Fig 10 & Fig 11). We did a linear regression (rxLinMod) of Arrival and Departure delays with the day of the week (ArrDelay~ Day of the Week) and plotted the coefficients (Figure_10) using rxLineplot function. The coefficients of linear regression for the arrival and departure delays for each carrier were also plotted (ArrDelay~ Carrier) in Fig_12. We have also checked the number of interstate flights operated and plotted a gradient heat map using rxCube, rxCrosstabs &ggplot2 (Figure_13).

Architecture

Figure 1: MRS Parallel Architecture

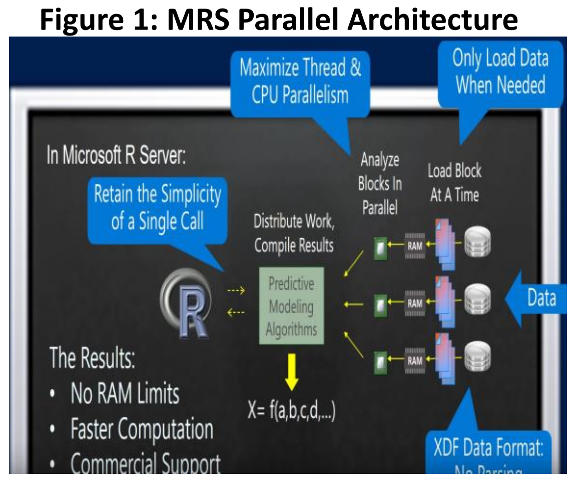
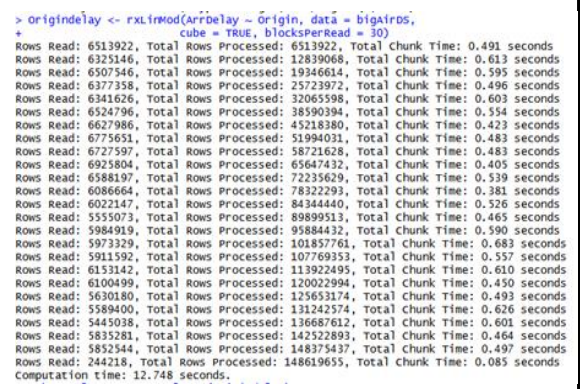

Maximize Thread & CPU Parallelism
Only Load Data When Needed
In Microsoft R Server:
Retain the Simplicity of a Single Call
Analyze Blocks In Parallel
Load Block At A Time
Distribute Work, Compile Results
Predictive Modeling Algorithms
Data
The Results:
No RAM Limits
Faster Computation
Commercial Support
X= f(a,b,c,d,...)
XDF Data Format: No Parsing

Figure 2: Reading Data in Chunks



```
> Origindelay <- rxLinMod(ArrDelay ~ Origin, data = bigAirDS,
+                         cube = TRUE, blocksPerRead = 30)
Rows Read: 6513922, Total Rows Processed: 6513922, Total Chunk Time: 0.491 seconds
Rows Read: 6325146, Total Rows Processed: 12839068, Total Chunk Time: 0.613 seconds
Rows Read: 6507546, Total Rows Processed: 19346614, Total Chunk Time: 0.595 seconds
Rows Read: 6377358, Total Rows Processed: 25723972, Total Chunk Time: 0.496 seconds
Rows Read: 6341626, Total Rows Processed: 32065598, Total Chunk Time: 0.603 seconds
Rows Read: 6524796, Total Rows Processed: 38590394, Total Chunk Time: 0.554 seconds
Rows Read: 6627986, Total Rows Processed: 45218380, Total Chunk Time: 0.423 seconds
Rows Read: 6775651, Total Rows Processed: 51994031, Total Chunk Time: 0.483 seconds
Rows Read: 6727597, Total Rows Processed: 58721628, Total Chunk Time: 0.483 seconds
Rows Read: 6925804, Total Rows Processed: 65647432, Total Chunk Time: 0.405 seconds
Rows Read: 6588197, Total Rows Processed: 72235629, Total Chunk Time: 0.539 seconds
Rows Read: 6086664, Total Rows Processed: 78322293, Total Chunk Time: 0.381 seconds
Rows Read: 6022147, Total Rows Processed: 84344440, Total Chunk Time: 0.526 seconds
Rows Read: 5555073, Total Rows Processed: 89899513, Total Chunk Time: 0.465 seconds
Rows Read: 5984919, Total Rows Processed: 95884432, Total Chunk Time: 0.590 seconds
Rows Read: 5973329, Total Rows Processed: 101857761, Total Chunk Time: 0.683 seconds
Rows Read: 5911592, Total Rows Processed: 107769353, Total Chunk Time: 0.557 seconds
Rows Read: 6153142, Total Rows Processed: 113922495, Total Chunk Time: 0.610 seconds
Rows Read: 6100499, Total Rows Processed: 120022994, Total Chunk Time: 0.450 seconds
Rows Read: 5630180, Total Rows Processed: 125653174, Total Chunk Time: 0.493 seconds
Rows Read: 5589400, Total Rows Processed: 131242574, Total Chunk Time: 0.626 seconds
Rows Read: 5445038, Total Rows Processed: 136687612, Total Chunk Time: 0.601 seconds
Rows Read: 5835281, Total Rows Processed: 142522893, Total Chunk Time: 0.464 seconds
Rows Read: 5852544, Total Rows Processed: 148375437, Total Chunk Time: 0.497 seconds
Rows Read: 244218, Total Rows Processed: 148619655, Total Chunk Time: 0.085 seconds
Computation time: 12.748 seconds.
```

Figure 3: Time Taken for Rx Summary (Seconds)

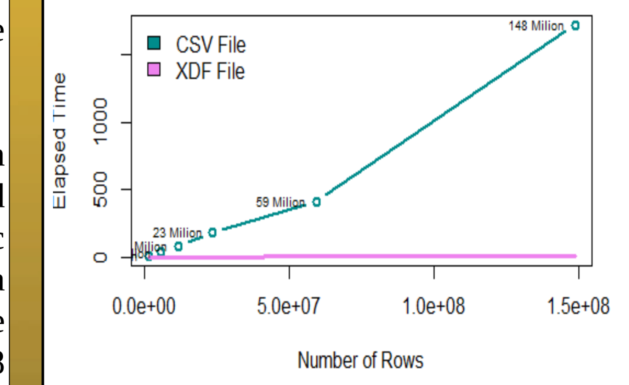


Figure 4: Time Taken to Convert from CSV to XDF (Minutes)

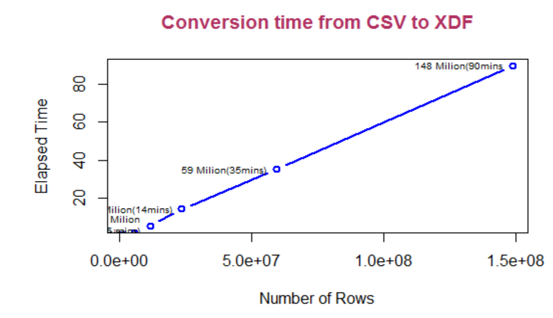
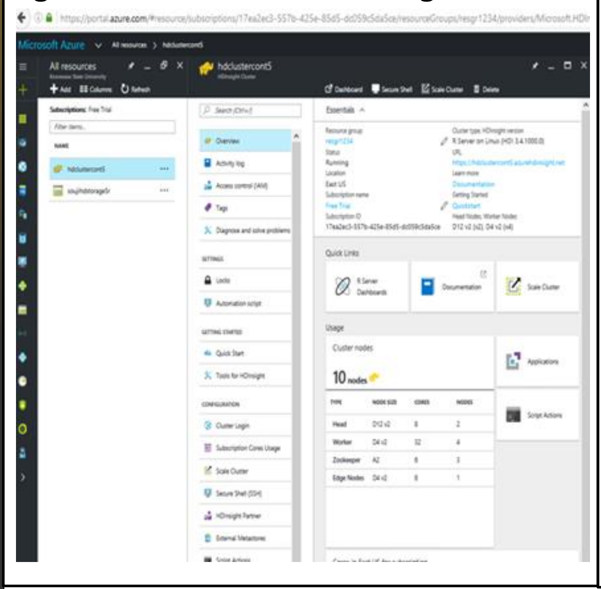


Figure 5: Azure Cluster HDInsight in Cloud



Analysis

Figure 6: Total no of flights operated by each Carrier.

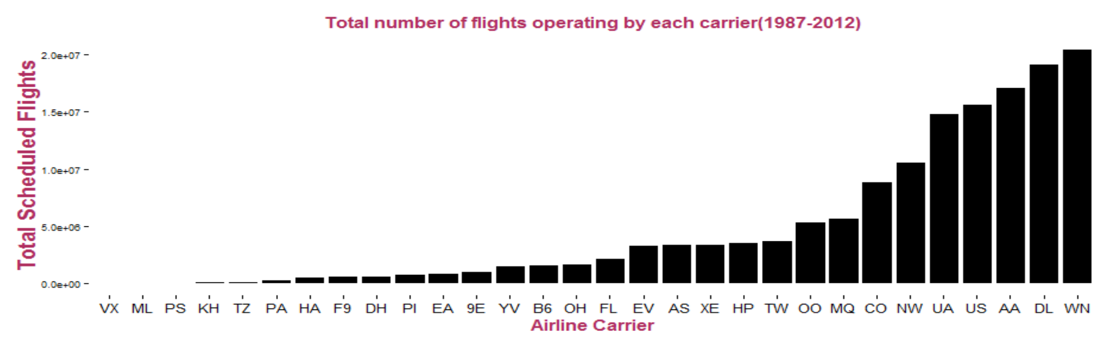


Figure 7: Percentage of Airline Arrival Delays (>15mins) by Each Carrier

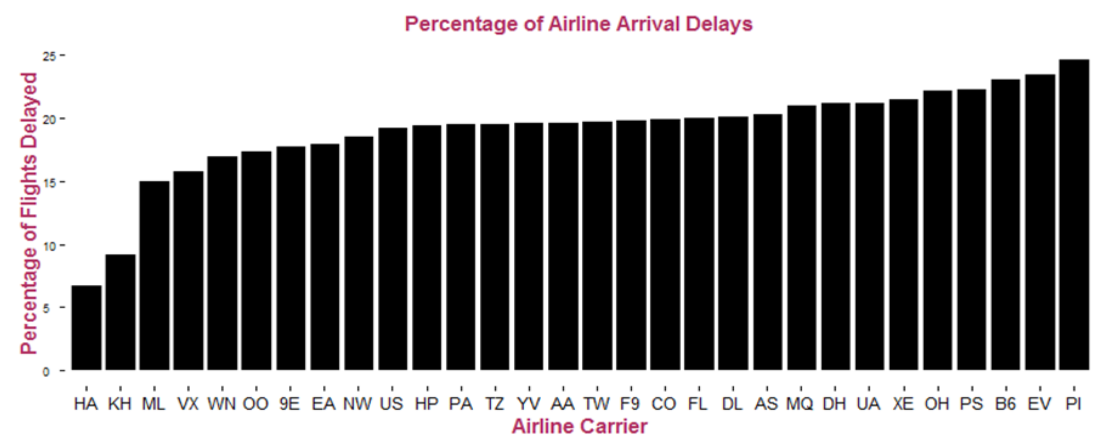


Figure 8: Average Delays on all carriers by Various Causes

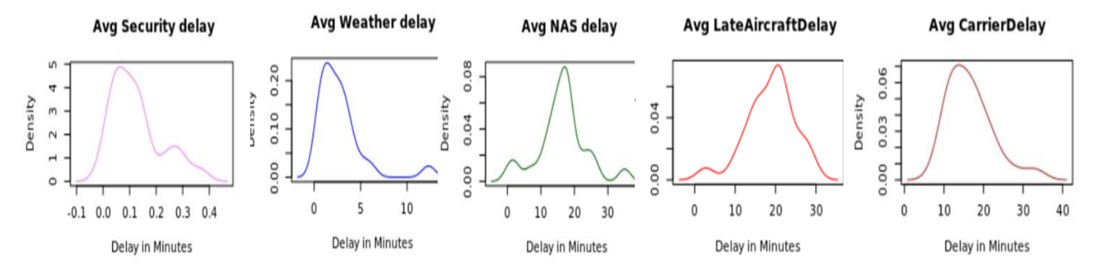
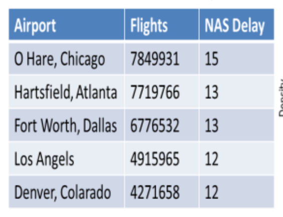


Figure 9: NAS Delays in all Airports

Table:1 Five Busiest Airports



Airport	Flights	NAS Delay
O Hare, Chicago	7849931	15
Hartsfield, Atlanta	7719766	13
Fort Worth, Dallas	6776532	13
Los Angels	4915965	12
Denver, Colarado	4271658	12

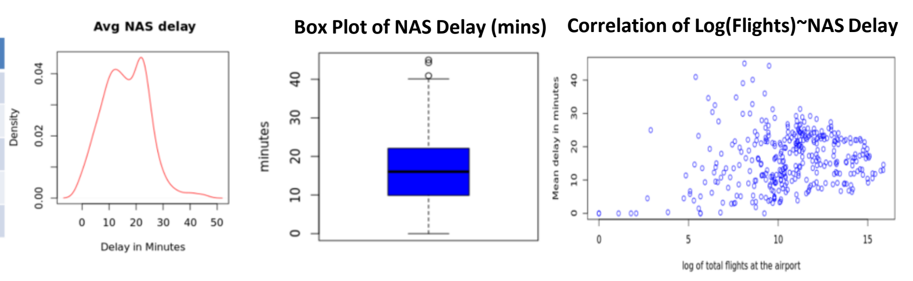


Figure 10: Arrival and Departure Delays (Coefficients) by Day of Week

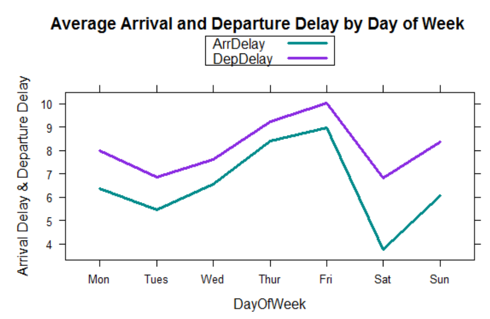


Fig_11: Mean Arrival delay by Day of Week by Hour

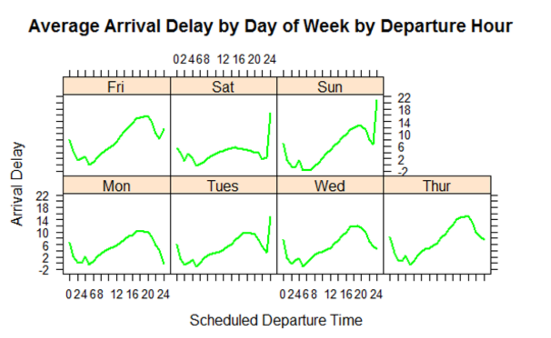


Figure 12: Average Arrival & Departure Delay (Coefficients) by Carrier

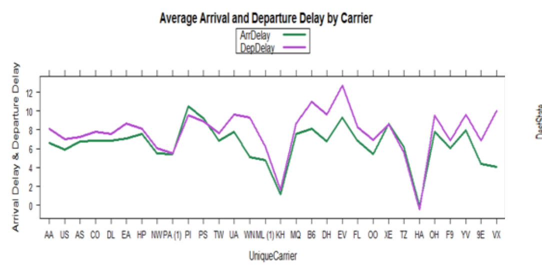
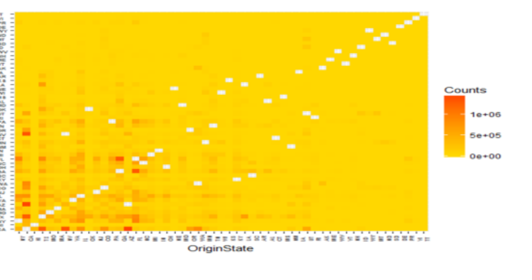


Fig 13: Heat map of Total Interstate flights



Results

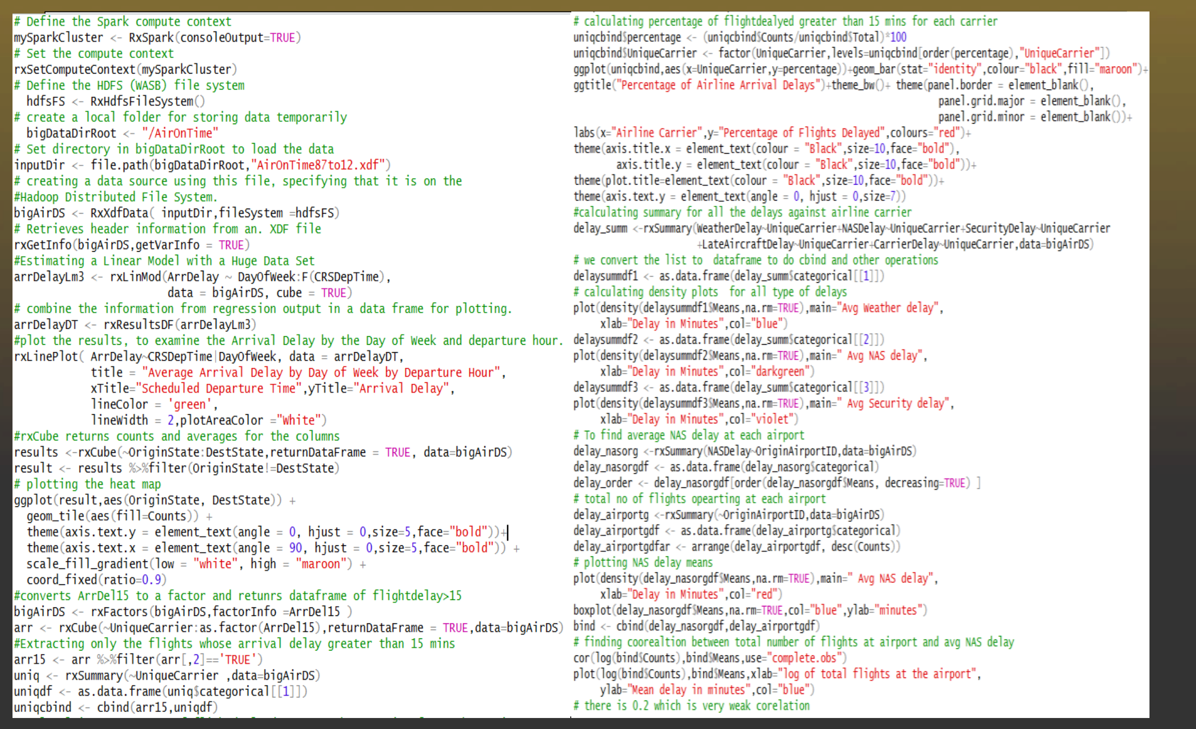
- The memory limitation of R can be overcome by using Microsoft R Server and its high performance analytic functions.
- A local computer installed with MRS is sufficient to do analysis on up to 150 million rows of data and it can be clustered in Azure for bigger data.
- XDF file is efficient for processing bigdata than CSV.
- The top five carriers are Southwest, Delta, American, US Airways and United. Lowest are Virgin America and Hawaiian airlines.
- The worst performing carriers by greater than 15 minutes delay are Pacific and Express Jet, Jet Blue etc whereas Southwest has only 17% delayed flights.
- Security and Weather delays contribute very less for the overall delays and majority of the delays are by Carrier, Late aircraft, and NAS.
- Average delay caused by Airports(NAS delay) is 30 minutes and surprisingly all five busiest airports of U.S perform better than average.
- There is a weak correlation (0.2) with number of flights operated and delay.
- As expected Fridays are worst for arrival and departure delays and Saturdays are the least.
- The delay is more between 4pm and 8pm and least between 2am-8am.
- Highest number of interstate flights are between California-Nevada, California-Arizona, California-Texas.

Conclusion

Though there are many Big Data packages available in R, MRS comes as the best option to overcome the memory problem. With the Azure cloud service, enterprise security and deployable web apps are made possible. Data analysts who are proficient in R need not learn Python to solve big data problems.

Though,we have done exploratory data analysis using rxSummary, rxCrosstabs, rxCube, RxLineplot, rxHistogram functions, and linear regression(Fig_10 & Fig_12) using rxLinmod, we can also do Logistic(rxLogit), Decision Tree(rxDTree), Decision Forest(rxDForest), Clustering(rxKmeans), Naïve Bayes(rxNaiveBayes) analysis in MRS. For complex analysis we can use Azure cloud to create a cluster and scale it as we need.

Sample Code



```r
# Define the Spark compute context
mySparkCluster <- RxSpark(consoleOutput=TRUE)
# Set the compute context
rxSetComputeContext(mySparkCluster)
# Define the HDFS (WASB) file system
  hdfsFS <- RxHdfsFileSystem()
# create a local folder for storing data temporarily
  bigDataDirRoot <- "/AirOnTime"
# Set directory in bigDataDirRoot to load the data
inputDir <- file.path(bigDataDirRoot,"AirOnTime87to12.xdf")
# creating a data source using this file, specifying that it is on the
#Hadoop Distributed File System.
bigAirDS <- RxXdfData( inputDir,fileSystem =hdfsFS)
# Retrieves header information from an. XDF file
rxGetInfo(bigAirDS,getVarInfo = TRUE)
#Estimating a Linear Model with a Huge Data Set
arrDelayLm3 <- rxLinMod(ArrDelay ~ DayOfWeek:F(CRSDepTime),
                        data = bigAirDS, cube = TRUE)
# combine the information from regression output in a data frame for plotting.
arrDelayDT <- rxResultsDF(arrDelayLm3)
#plot the results, to examine the Arrival Delay by the Day of Week and departure hour.
rxLinePlot( ArrDelay~CRSDepTime|DayOfWeek, data = arrDelayDT,
            title = "Average Arrival Delay by Day of Week by Departure Hour",
            xTitle="Scheduled Departure Time",yTitle="Arrival Delay",
            lineColor = 'green',
            lineWidth = 2,plotAreaColor ="White")
#rxCube returns counts and averages for the columns
results <-rxCube(~OriginState:DestState,returnDataFrame = TRUE, data=bigAirDS)
result <- results %>%filter(OriginState!=DestState)
# plotting the heat map
ggplot(result,aes(OriginState, DestState)) +
  geom_tile(aes(fill=Counts)) +
  theme(axis.text.y = element_text(angle = 0, hjust = 0,size=5,face="bold"))+
  theme(axis.text.x = element_text(angle = 90, hjust = 0,size=5,face="bold")) +
  scale_fill_gradient(low = "white", high = "maroon") +
  coord_fixed(ratio=0.9)
#converts ArrDel15 to a factor and retunrs dataframe of flightdelay>15
bigAirDS <- rxFactors(bigAirDS,factorInfo =ArrDel15 )
arr <- rxCube(~UniqueCarrier:as.factor(ArrDel15),returnDataFrame = TRUE,data=bigAirDS)
#Extracting only the flights whose arrival delay greater than 15 mins
arr15 <- arr %>%filter(arr[,2]=='TRUE')
uniq <- rxSummary(~UniqueCarrier ,data=bigAirDS)
uniqdf <- as.data.frame(uniq$categorical[[1]])
uniqcbind <- cbind(arr15,uniqdf)
```

```r
# calculating percentage of flightdealyed greater than 15 mins for each carrier
uniqcbind$percentage <- (uniqcbind$Counts/uniqcbind$Total)*100
uniqcbind$UniqueCarrier <- factor(UniqueCarrier,levels=uniqcbind[order(percentage),"UniqueCarrier"])
ggplot(uniqcbind,aes(x=UniqueCarrier,y=percentage))+geom_bar(stat="identity",colour="black",fill="maroon")+
ggtitle("Percentage of Airline Arrival Delays")+theme_bw()+ theme(panel.border = element_blank(),
                                                                  panel.grid.major = element_blank(),
                                                                  panel.grid.minor = element_blank())+
labs(x="Airline Carrier",y="Percentage of Flights Delayed",colours="red")+
theme(axis.title.x = element_text(colour = "Black",size=10,face="bold"),
      axis.title.y = element_text(colour = "Black",size=10,face="bold"))+
theme(plot.title=element_text(colour = "Black",size=10,face="bold"))+
theme(axis.text.y = element_text(angle = 0, hjust = 0,size=7))
#calculating summary for all the delays against airline carrier
delay_summ <-rxSummary(WeatherDelay~UniqueCarrier+NASDelay~UniqueCarrier+SecurityDelay~UniqueCarrier
                       +LateAircraftDelay~UniqueCarrier+CarrierDelay~UniqueCarrier,data=bigAirDS)
# we convert the list to  dataframe to do cbind and other operations
delaysummdf1 <- as.data.frame(delay_summ$categorical[[1]])
# calculating density plots  for all type of delays
plot(density(delaysummdf1$Means,na.rm=TRUE),main="Avg Weather delay",
     xlab="Delay in Minutes",col="blue")
delaysummdf2 <- as.data.frame(delay_summ$categorical[[2]])
plot(density(delaysummdf2$Means,na.rm=TRUE),main=" Avg NAS delay",
     xlab="Delay in Minutes",col="darkgreen")
delaysummdf3 <- as.data.frame(delay_summ$categorical[[3]])
plot(density(delaysummdf3$Means,na.rm=TRUE),main=" Avg Security delay",
     xlab="Delay in Minutes",col="violet")
# To find average NAS delay at each airport
delay_nasorg <-rxSummary(NASDelay~OriginAirportID,data=bigAirDS)
delay_nasorgdf <- as.data.frame(delay_nasorg$categorical)
delay_order <- delay_nasorgdf[order(delay_nasorgdf$Means, decreasing=TRUE) ]
# total no of flights opearting at each airport
delay_airportg <-rxSummary(~OriginAirportID,data=bigAirDS)
delay_airportgdf <- as.data.frame(delay_airportg$categorical)
delay_airportgdfar <- arrange(delay_airportgdf, desc(Counts))
# plotting NAS delay means
plot(density(delay_nasorgdf$Means,na.rm=TRUE),main=" Avg NAS delay",
     xlab="Delay in Minutes",col="red")
boxplot(delay_nasorgdf$Means,na.rm=TRUE,col="blue",ylab="minutes")
bind <- cbind(delay_nasorgdf,delay_airportgdf)
# finding coorealtion between total number of flights at airport and avg NAS delay
cor(log(bind$Counts),bind$Means,use="complete.obs")
plot(log(bind$Counts),bind$Means,xlab="log of total flights at the airport",
     ylab="Mean delay in minutes",col="blue")
# there is 0.2 which is very weak corelation
```