Hadoop + R

Ways to use Hadoop and R together:

1. RHadoop

RHadoop is a collection of R packages from Revolution Analytics that allows managing and analyzing data with Hadoop.

RHDFS provides functions for HDFS file management from R

RMR provides functions to implement Hadoop MapReduce functionality in R

Rhbase provides functions to access and manipulate HBase data from R using thrift API

Plyrmr provides functions to perform data HDFS data manipulation

Ravro can be used to read and write Avro files

Available HDFS file management functions:

hdfs.copy, hdfs.move, hdfs.rename, hdfs.put, hdfs.get, hdfs.file,

hdfs.write, hdfs.close, hdfs.flush, hdfs.read, hdfs.seek,

hdfs.tell, hdfs.defaults, hdfs.ls, hdfslist.files, hdfs.delete, hdfs.rm,

hdfs.del, hdfs.dircreate, hdfs.mkdir, hdfs.chmod,hdfs.chown,

hdfs.file.info, hdfs.exists, hdfs.init, hdfs.line.reader, hdfs.read.text.file

```
> Library(rhdfs)
> sour <-  hdfs.read.text.file("/hdfs/path/to/file.csv")
        No encryption was performed by peer.
> sour
```

2. RHive (Available on CNJ test cluster)

RHive package does not support accessing Hive data which is secured by Kerberos.

(Jira ticket is open)

But RHive can be used to access and manage HDFS data directly.

```
> library("rJava")

> library("RHive")

>.jinit()

> for(l in list.files('/usr/lib/hive/bin/')){.jaddClassPath(paste("/usr/lib/hive/bin/",l,sep=""))}

> for(l in list.files('/usr/lib/hive/lib/')){ .jaddClassPath(paste("/usr/lib/hive/lib/",l,sep=""))}

> for(l in  list.files('/usr/lib/hadoop/bin/')){.jaddClassPath(paste("/usr/lib/hadoop/bin/",l,sep=""))}

> for(l in list.files('/usr/lib/hadoop/lib/')){.jaddClassPath(paste("/usr/lib/hadoop/lib/",l,sep=""))}

> for(l in list.files('/etc/hadoop/conf/')){.jaddClassPath(paste("/etc/hadoop/conf/",l,sep=""))}

> for(l in list.files('/usr/lib/hadoop/')){ .jaddClassPath(paste("/usr/lib/hadoop/",l,sep=""))}

> for(l in list.files('/usr/lib/hadoop-mapreduce/')){ .jaddClassPath(paste("/usr/lib/hadoop-
mapreduce/",l,sep=""))}

>.jclassPath()

> rhive.hdfs.cat("/hdfs/path/to/file.csv")
```

3. Hadoop Streaming

Using Hadoop Streaming, you can write your Map and Reduce functions in R that can read from stdin and write to stdout.

4. RHIPE

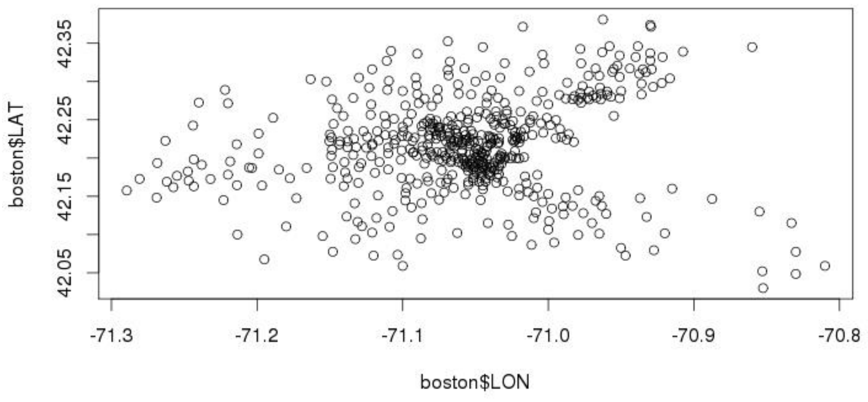
Using Rhipe, we can access HDFS and submit Map Reduce jobs from R.

Sample code (Using RHDFS)

(Note: This code snippet focuses on accessing HDFS data using R, than the data analysis.)

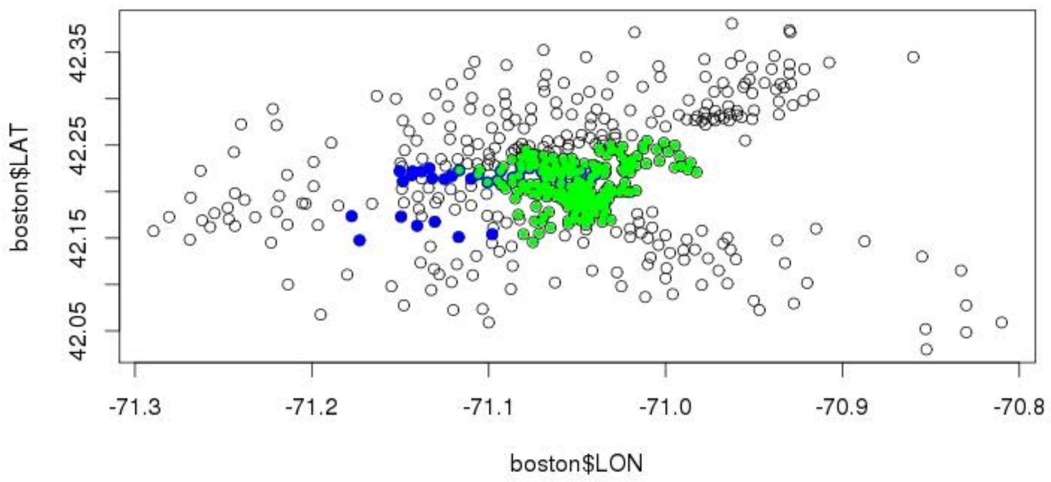
Here we are predicting the house prices based on the location in Boston area. This data comes from the UCI Machine Learning Repository.

```
# Set Hadoop env variables

> Sys.setenv("HADOOP_CMD"="/usr/lib/hadoop/bin/hadoop")

> Sys.setenv("HADOOP_STREAMING"="/usr/lib/hadoop-mapreduce/hadoop-streaming-2.4.0.2.1.2.0-402.jar")

# Load rhdfs

> library(rhdfs)

> hdfs.init()


# Read data from HDFS (When data is read using hdfs.read, its class is character)

> sour <- hdfs.read.text.file("/user/xbblj3x/boston.csv")

> cons <- textConnection(sour)

> boston <- read.csv(cons)


# Plot observations

> plot(boston$LON, boston$LAT)
```



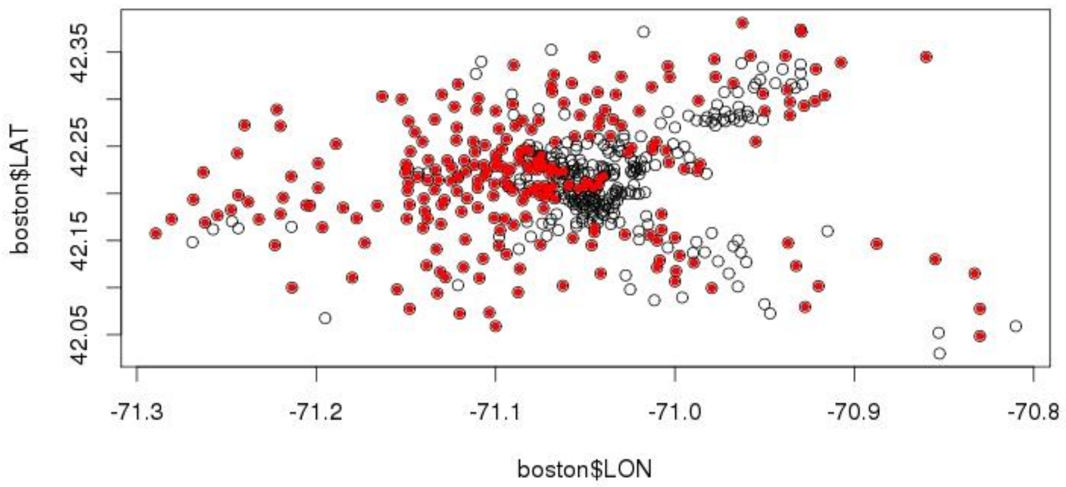
# Plot polution

```
> summary(boston$NOX)

> points(boston$LON[boston$NOX>=0.55], boston$LAT[boston$NOX>=0.55], col="green", pch=20)
```

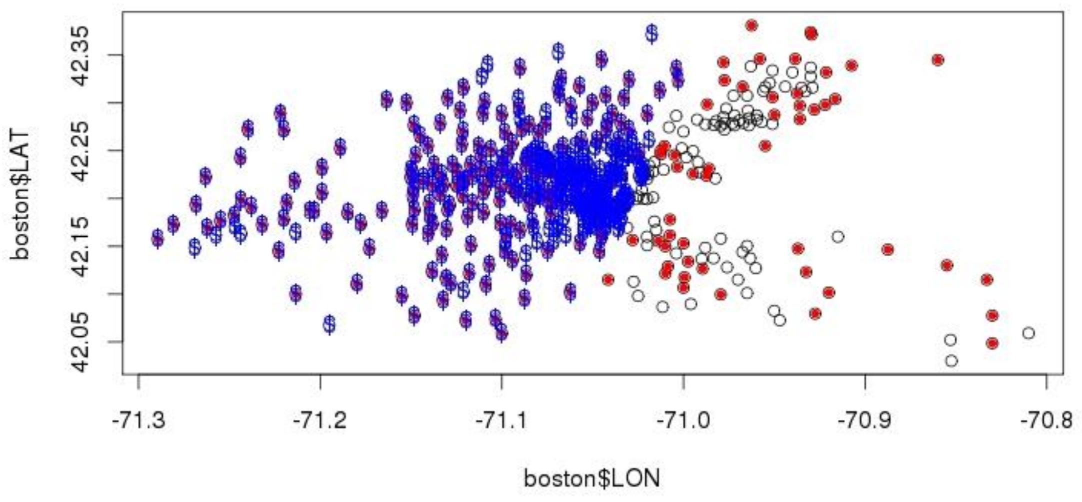


# Plot prices

```
> plot(boston$LON, boston$LAT)

> summary(boston$MEDV)

> points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)
```



```
# Linear Regression using LAT and LON

> plot(boston$LAT, boston$MEDV)

> plot(boston$LON, boston$MEDV)

> latlonlm = lm(MEDV ~ LAT + LON, data=boston)

> summary(latlonlm)

# Visualize regression output

> plot(boston$LON, boston$LAT)

> points(boston$LON[boston$MEDV>=21.2], boston$LAT[boston$MEDV>=21.2], col="red", pch=20)


> latlonlm$fitted.values

> points(boston$LON[latlonlm$fitted.values >= 21.2], boston$LAT[latlonlm$fitted.values >= 21.2], col="blue",
pch="$")
```



Accessing Hadoop (Hive) from Windows R

1. RODBC

Prerequisites:

a. Install Windows MIT Kerberos Ticket Manager

b. Install Hortonworks Hive ODBC Driver 1.4

c. Get a Valid Kerberos Ticket

d. Create ODBC connection ("r37cn00_TPC") to Hive

```
> library(RODBC)

> sourconn= odbcConnect("r37cn00_TPC",uid="")

> odbcGetInfo(sourconn)

> sqlTables(sourconn)

> Sour_data <- sqlQuery(sourconn,"select * from infosec.cities limit 10;",rows_at_time = 100)
```

2. RJDBC

Using JDBC connection we can connect to Hive from R.

Prerequisites:

1. Valid Kerberos ticket

2. Hive jar

```
>library(RJDBC)

>Hive_JDBC <- JDBC(driverClass="org.apache.hadoop.hive.jdbc.HiveDriver",
classPath="/path/to/hive/jar")

>Hive_con <- dbConnect(Hive_JDBC, "jdbc:hive://localhost:10000/default", "USERID", "PWD")
```

Summary

Most of the statistics, analytics and visualization libraries in R are non-distributed and operate on data in-memory. In order to use real parallel power of Hadoop, R programs should be written in such a way that the data analysis can be distributed across the data nodes so that data can be processed in parallel i.e. using Map Reduce paradigm.

For writing Map Reduce programs in R, RHadoop (RMR) and Rhipe packages can be used.

RHDFS and RHbase can be used to access and manipulate HDFS and HBase data respectively from R.

To access Hive data, RHive and RODBC/RJDBC packages are available.

For parallel processing in R, packages like distributedR, parallel and many other are available.

Revolution Analytics has also introduced RevoScaleR package for big data analytics.

Hopefully we will have more R packages in future which can access, manipulate and analyze the data residing in Hadoop and also support Kerberos.

References

https://github.com/RevolutionAnalytics/RHadoop/wiki

https://blog.udemy.com/r-hadoop/

http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html

https://github.com/jseidman/hadoop-R

https://www.datadr.org/

http://www.slideshare.net/ChicagoHUG/getting-started-with-r-hadoop-chug-20120815