**Fall 2016 –** **OPIM 5503 Data Analytics Using R**

**Instructor: Prof. Ram Gopal**



**Team- Random R**

**Project 2 – Big data Analysis with R - Hadoop**

**Ashish Gupta, Krishna Muddodi, Nachiket Garge, Rajit Podder, Sairam Pramod Kotla**

**INDEX**

1. **Executive Summary 3**
2. **Business Objective 3**
3. **Introduction 3**
4. **Setting Up R-Hadoop Environment 4**
5. **Basic Hadoop Operations 15**
6. **Performance Test 18**
7. **Data Manipulation in R-Hadoop 19**
8. **Modeling in R-Hadoop 21**
9. **References 24**
10. **Appendix 25**

**Executive Summary:**

R and Hadoop are definitely two hottest buzz words that you must have heard recently. Through this report, we aim to display the analytical prowess that can be brought combining R and Hadoop. We as a team performed various analytical operations including data wrangling, handling data in Hadoop distributed file system and linear modeling using map reduce functionality. In this report, we have covered installation of cloudera VM and setup of R on the virtual machine. Once R is installed we performed various data manipulation operations using plyrmr package. We displayed how one can perform basic Hadoop operations using R. Finally, we performed modeling on a well-acquainted data set “women” data set which comes along with R.

**Business Objective:**

With amount of activities increasing daily on internet the data that we have is increasing rapidly. The data in huge volume that is collected during daily business operations is known as “Big data”. Big data can be analyzed for business insights that can lead to better strategic business decisions. R can successfully analyze data up to a million rows, but as the volume and variety of data increases it is necessary to scale things up in R. Our business objective through this project is to show how one can scale up R to analyze big data.

**Introduction**

**Limitations of R:** R loads the data in memory so if the data set is huge it goes out of memory or the processing often becomes very slow. To solve this problem, we can sample the data, but sampling the data means we are not using the data which we have and which can reduce the efficiency of our analysis. Another way is to buy an expensive hardware with larger in memory space in order to perform big data analysis, but this approach is expensive as bigger hardware would definitely cost more. Here we have used big data commodity clusters which is a cheap way to scale up the operations in R.

**Packages Used:** We have used three packages developed by Revolution Analytics.

1. RHDFS: Used to perform basic Hadoop operations in R
2. RMR2: Provides map and reduce in R used to perform complex analytical operations.
3. PLYRMR: Used for data manipulation in R

**Setting up the RHadoop environment**

Because RHadoop requires an R and Hadoop integrated environment, an environment with both R and Hadoop installed needs to be prepared. For this the Cloudera QuickStart VM (free), can be used which contains a single node Apache Hadoop Cluster and R.

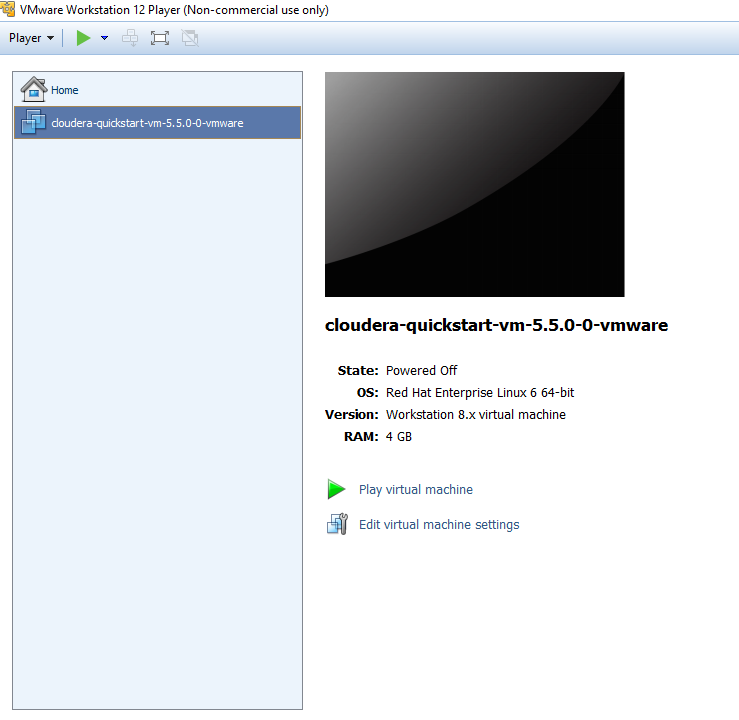
The following steps need to be performed to set up a Hadoop environment using the Cloudera QuickStart VM:

Download the VM from the link provided below:

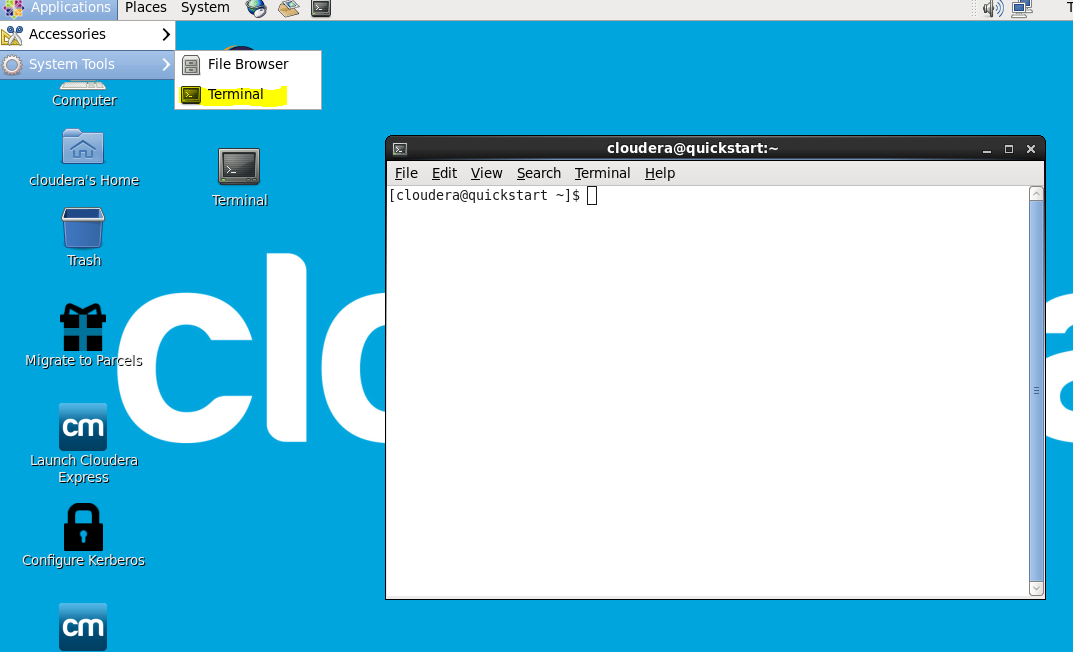
<https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0>



After downloading, launch the VM using a VM player. Here, we have used VMware Workstation 12 player.



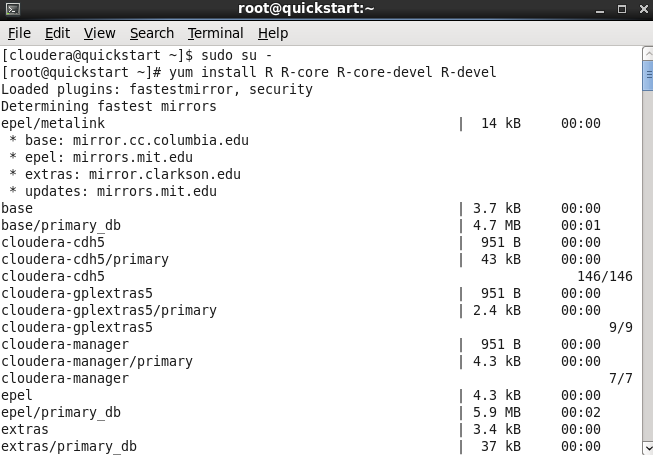
Then open the Terminal in the VM. Below is a screenshot of how to open the terminal if you are unsure about it. All the code will be written in the command prompt.



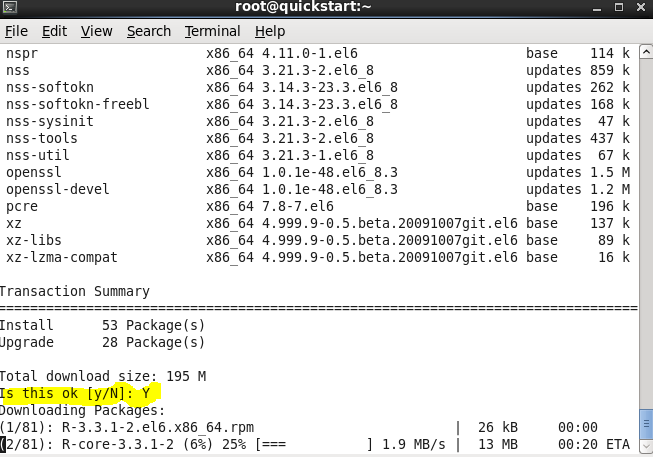
Use the following command to install R in the Virtual Machine make sure you are root using command **sudo su –**

**$ yum install R R-core R-core-devel R-devel**

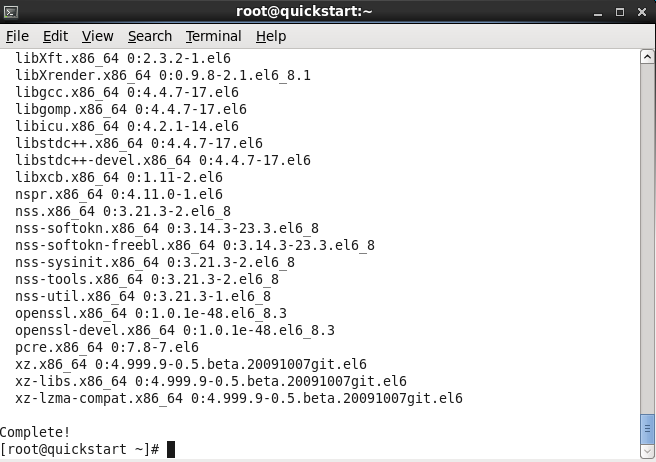
You should get the following result:



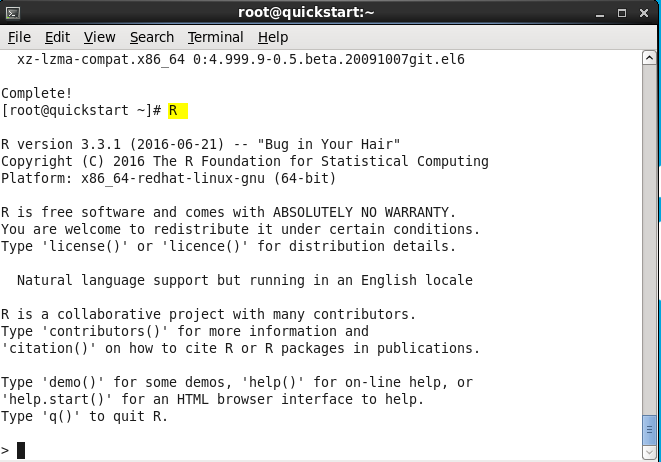
Choose “Y” for saying yes to download R.



After the installation is done, the following “complete” message should appear.

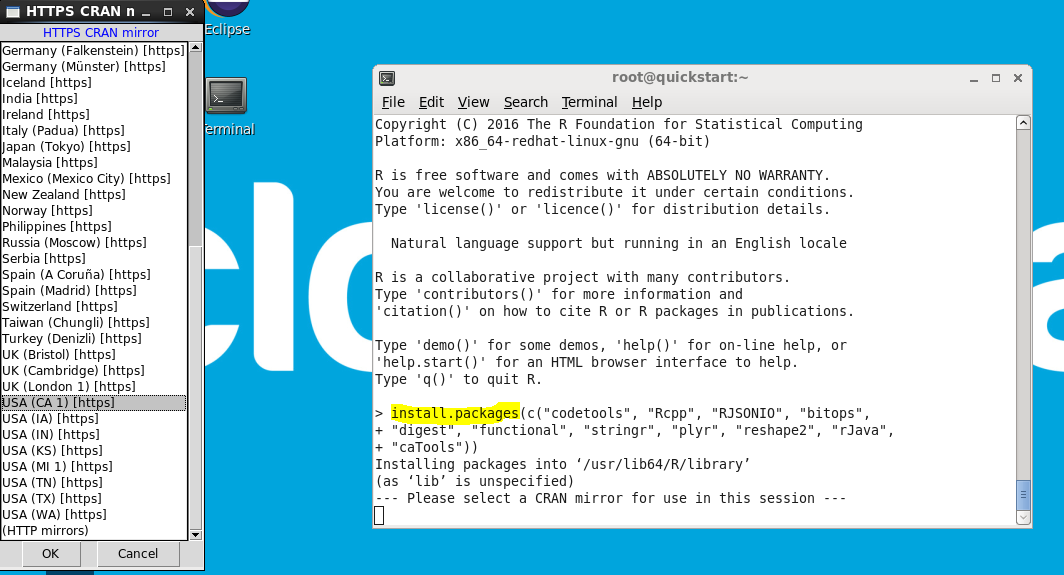


Then you can type “R” and check if R is now running on your virtual machine as shown below:

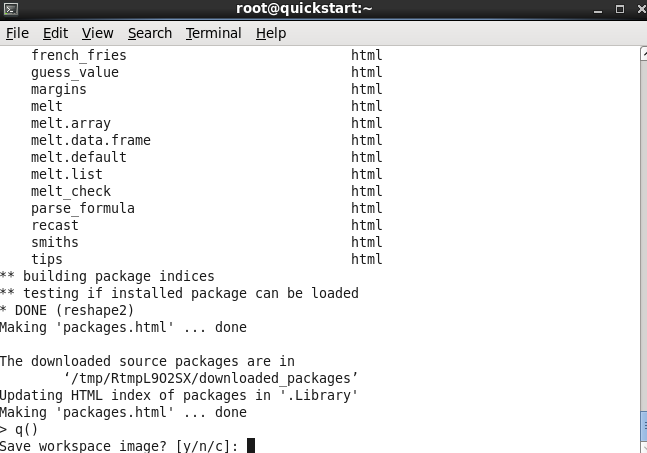


Then install packages in R using the command “install.packages”. We had the packages shown in the screenshot below installed which is required for rmr2. Also make sure to choose “USA[CA]” for the HTTPS CRAN mirror.

**install.packages(c("codetools", "Rcpp", "RJSONIO", "bitops", "digest", "functional", "stringr", "plyr", "reshape2", "rJava", "caTools"))**

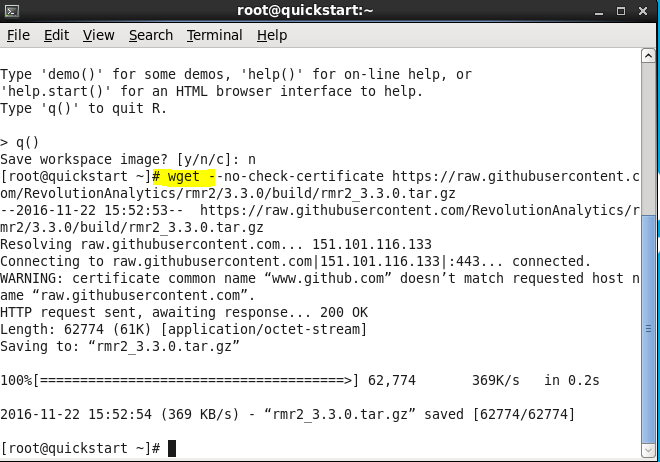


The screenshot below is what you will get after installing the dependent packages.

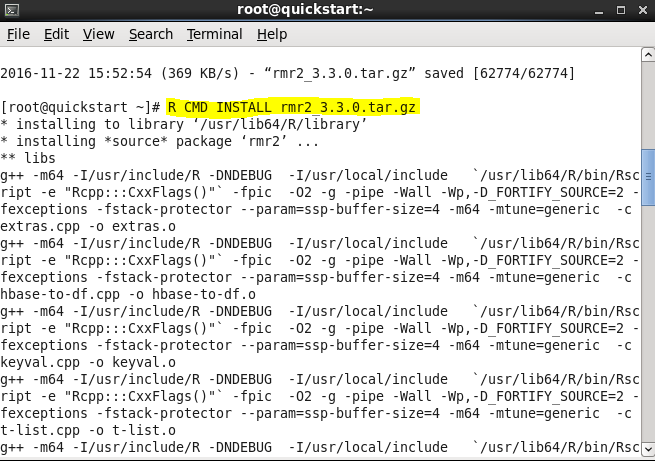


Next, rmr2 needs to be downloaded to the VM. The following is the command for it.

wget --no-check-certificate https://raw.githubusercontent.com/RevolutionAnalytics/rmr2/3.3.0/build/rmr2\_3.3.0.tar.gz



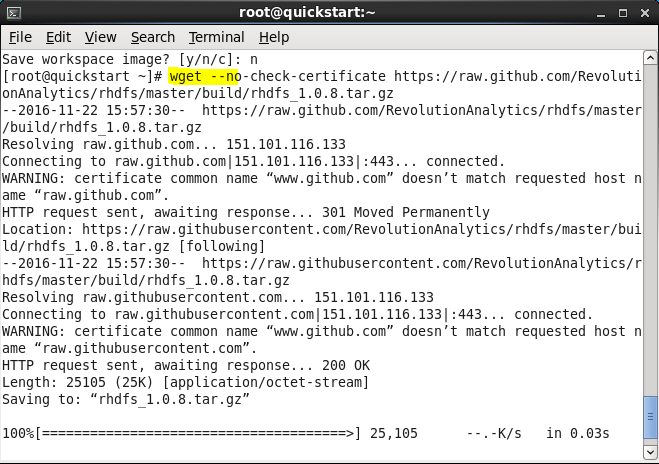
Then install the rmr2 package as shown in the screenshot below:



Next, we need to install the RHDFS package. The rhdfs package is the interface between R and HDFS, which allows users to access HDFS from an R console.

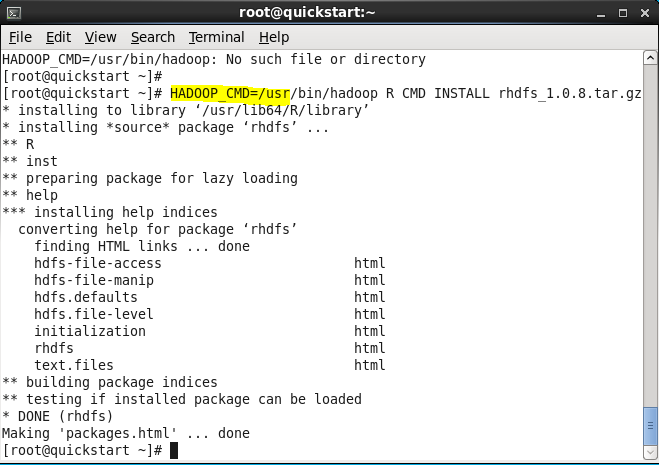
First, download the RHDFS package from github using the following command:

wget --no-check-certificate <https://raw.github.com/RevolutionAnalytics/rhdfs/master/build/rhdfs_1.0.8.tar.gz>



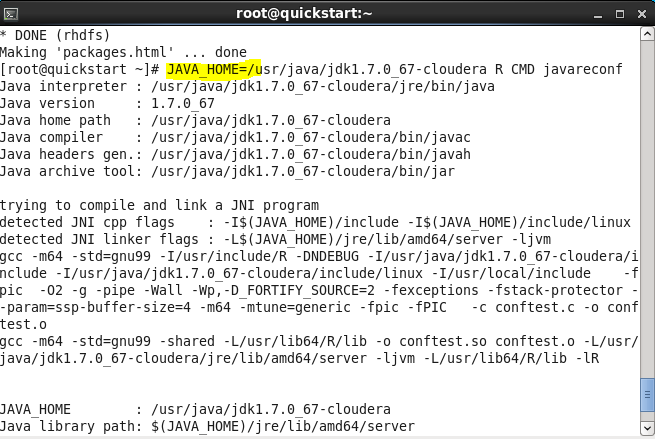
Next install RHDFS using the command below:

HADOOP\_CMD=/usr/bin/hadoop R CMD INSTALL rhdfs\_1.0.8.tar.gz



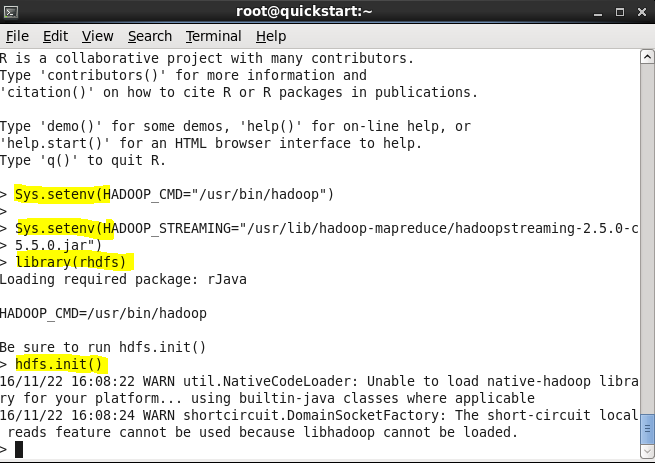
Then setup the JAVA\_HOME directory path using the command below:

JAVA\_HOME=/usr/java/jdk1.7.0\_67-cloudera R CMD javareconf



Finally, set up the system environment and initialize the RHDFS using the commands below:

* **Sys.setenv(HADOOP\_CMD="/usr/bin/hadoop")**
* **Sys.setenv(HADOOP\_STREAMING="/usr/lib/hadoop-mapreduce/hadoop-streaming-2.5.0-cdh5.5.0.jar")**
* **library(rhdfs)**
* **hdfs.init()**



Again after quitting the R session with q() command.

Our final step is to install R studio,

# wget <http://download2.rstudio.org/rstudio-server-0.98.490-x86_64.rpm>

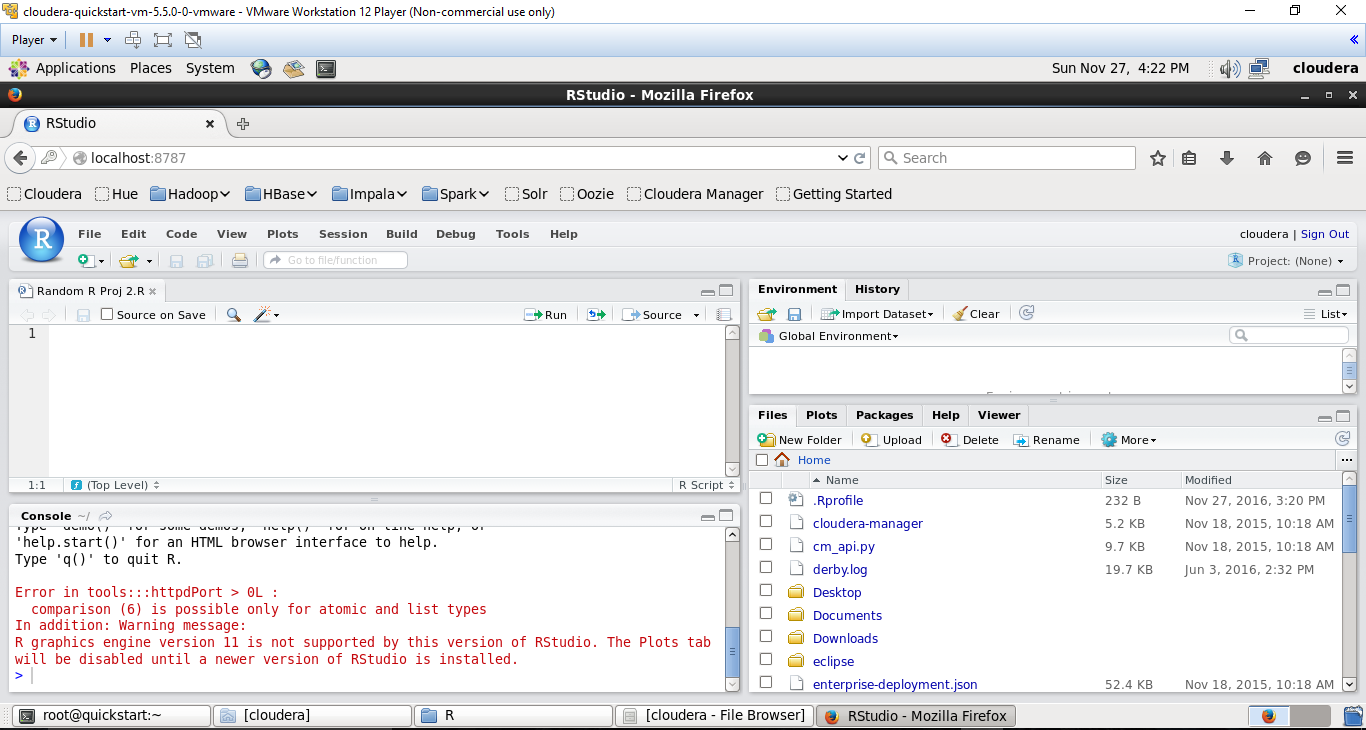
# yum install --nogpgcheck rstudio-server-0.97.248-x86\_64.rpm

Once the installation is done, open a browser and visit below URL:

<http://localhost:8787>

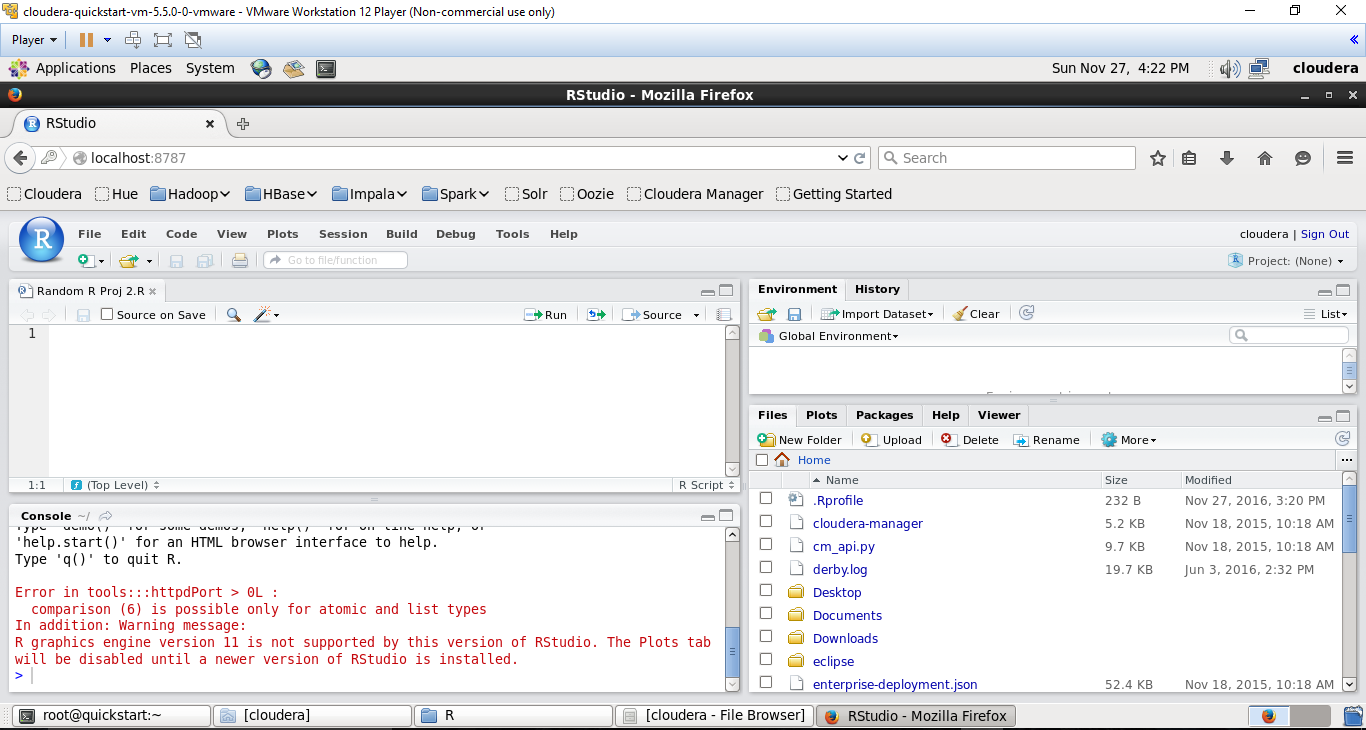
R studio works on local machine on 8787 port number as shown below,  
if prompted for user name and password,  
Username: cloudera

Password: cloudera



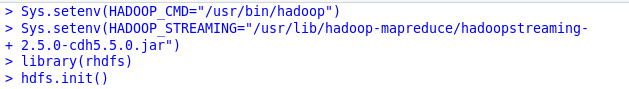
**Environment Setup:**

The below screenshot shows the installed Rstudio version in HDFS.



In order to validate the HDFS, we can initialize the HDFS by using the hdfs.init() function. Also, setting the environment variables.





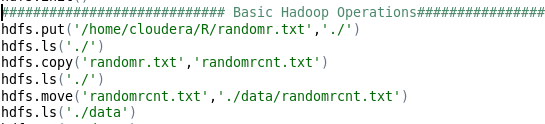
**Basic Hadoop Operations:**

The below illustrations show us how the basic operations are performed in HDFS.

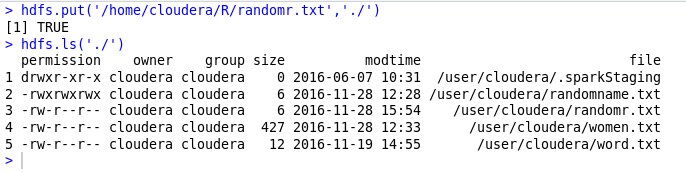
**Put:** Place a file in HDFS from a local system.

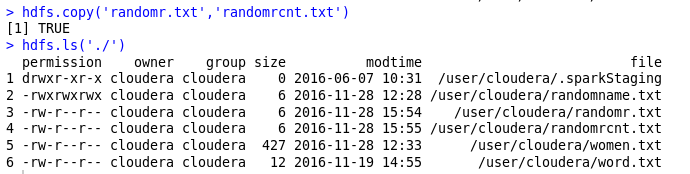
**Copy:** Copiesa file in the same directory in HDFS.

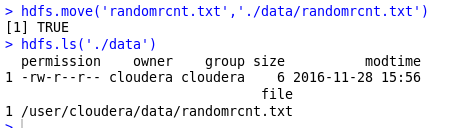
**Move:** Moves a file from one location to other in HDFS.



From the ls command, we can ensure the files are placed in HDFS after corresponding file operations.





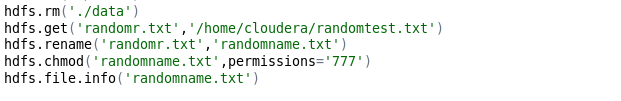


**Delete:** Deletes the directory from the HDFS.

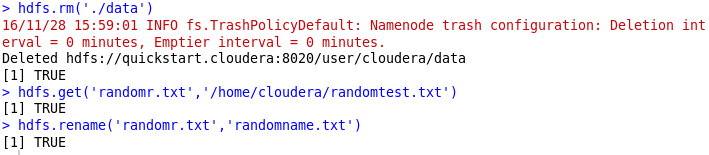
**Get:** Command used to retrieve a file from HDFS to local system.

**Rename:** Command used to rename a file in HDFS.

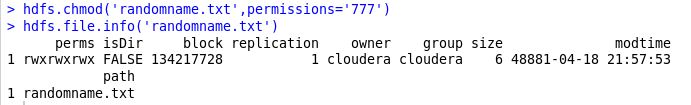
**Chmod:** Command used to change the file permissions in HDFS.



TRUE indicates the successful operation of the commands.



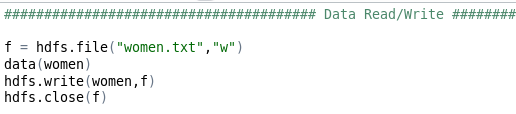
In order to check the file permissions and other properties we can use the hdfs.file.info() command.



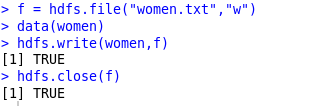
**READ/WRITE Operations:**

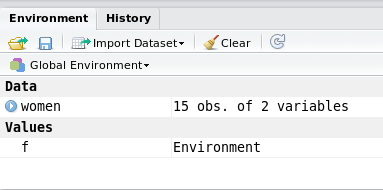
**Write:** Performs write operation to a file.

Initially we will open a file in write mode and assign to an object. Read the required file and then we will write the file to the assigned object f and close the object.



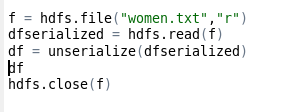
TRUE indicates the operation is successful.

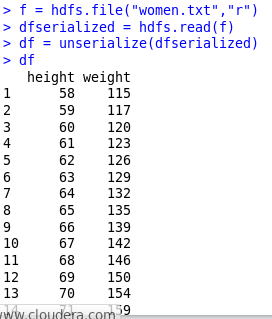




**Read:** Reads a file to an object.

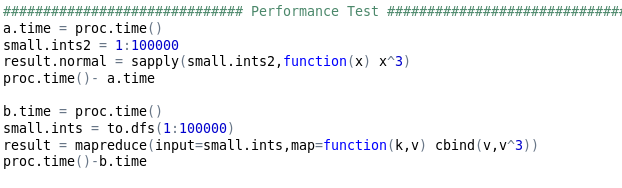
This operation performs opening and assigning the file to an object in Read mode. Unserialize function reads an object from connection or a raw vector.





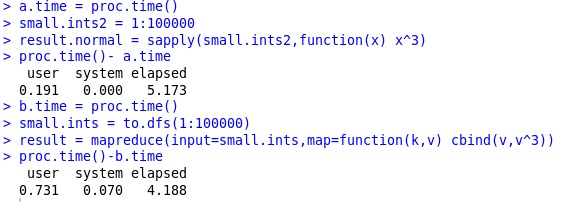
**Performance Test:**

Below code shows the time required to compute the operation using MapReduce and normal R functions.



Results specifies that the time required to compute using MapReduce is less than R computation time.

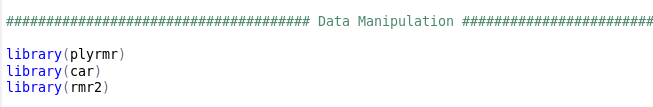
For the big data, the computation time is significantly lesser in HDFS using MapReduce.



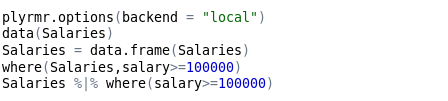
**Data Manipulations using PLYRMR package:**

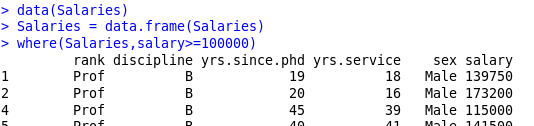
**Plyrmr** package is analogous to plyr or reshape2 function in R which performs the data manipulation via MapReduce framework.

Validating the below libraries are installed before manipulating the data.

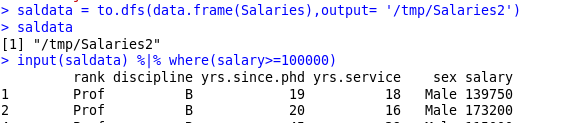


Setting the execution mode to local. Filtering the data with salary greater than or equals 100,000. Below code snippet demonstrates the same.





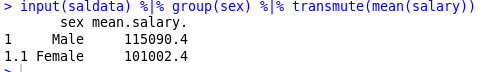
Setting the path of the file in HDFS to saldata.



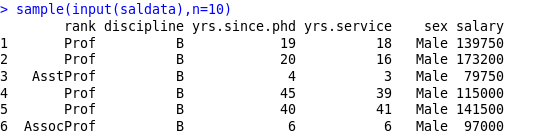
We can generate the mean of salary from the salary data and transmute function is used to restore the results to saldata. Pipes are used to provide the output of first command to input of second command.



Below code is used to group the salary by sex



Sampling the 10 records of saldata.



**Linear Regression in R-Hadoop**

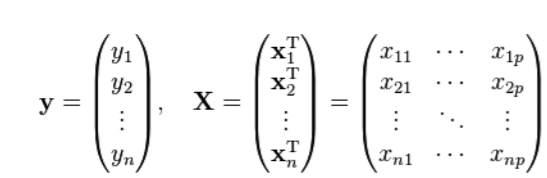
R-Hadoop is an open source project developed by Revolution Analytics. It provides a client-side integration between R and Hadoop. This allows running Map Reduce jobs within R with the help of several packages. In this section, we’ll consider the map-reduce version of multiple linear regression. This could be extremely helpful for boosting computational performance for large datasets and where loading data into memory for processing could be a serious limitation in terms of the volume of the data.

Let’s consider the following Regression,



Where, prediction for the ith observation is a linear combination of p independent variables times their coefficients (β1, β2........ βp) and the error term €i

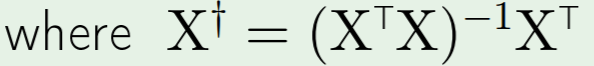
Linear Regression in the third form (Matrix form) can be represented as,

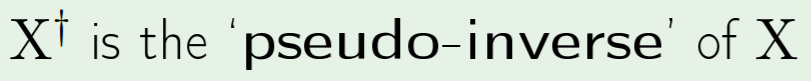




The solution of the above equation for β (coefficients of independent terms) would be







XTX is a square matrix and is called as the “Matrix of the Linear System”

XTy is not necessarily a square matrix and is called “Free Term”

Even for very large data sets (Big Data), the number of columns are far less than the number of rows. There could be millions of observations (tuples) but only some hundreds of characteristics (fields). For ex, let’s consider a Big Data with 100 million rows and 1001 columns. The dimension of matrix X (containing all independent variables) would be 108x1000. The Dimension of Matrix y (dependent variable) would be 108x1. Thus, it’s a big data challenge to compute XTX and XTy because of the order of the matrices X and y. However, the dimension of matrix XTX is 1000 x 1000 and dimension of matrix XTy is 1000 x 1. Therefore, once we have calculated XTX and XTy, any further computation over these matrix is not a big data challenge since the two matrices are manageable to load in the memory at once.

In R, there is a predefined function to solve linear systems (compute inverse of a matrix times another matrix)

Solve (a,b…….)

This function takes two parameters “Matrix of the Linear System” and “Free Term”. In our case, we’ll call this function as



The Call of this function can be easily computed in R as data could be loaded at once.

Hadoop can be used to store input data and to perform the two multiplication. The problem of big matrix multiplication ( XTX and XTy) can be transformed so that it can use the map-reduce programming model which is specific to Hadoop. Matrix multiplication is such a central operation in many numerical algorithms. One basic matrix multiplication algorithm is “Divide and Conquer” <https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm>. This algorithm could be used for map-reduce implementation.

First we add a new column to our independent variable dataset, that contains 1 for all observations (to deal with the intercept) and load this data in the variable x.



Then, we add a new column to this matrix which just contains the serial number of the observations (1,2,3,4………total observations in women dataset) and write this to HDFS distributed file system in the variable “x.index”.

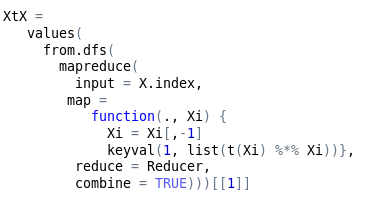


“x.index” is a big data object. In a real application, big data object would come from files already stored in HDFS. Similarly, we load the dependent variable data.

Now to compute XTX, following steps are performed

* Feed smaller chunks of ‘x.index’ to the map function as object Xi
* Subset these chunks barring the first column
* Compute XiTXi for each chunk
* Store the result of the above computation as the ‘value’ associated with a specific ‘key’
* Return this ‘key value’ pair and feed it to the reducer function
* Reducer function adds up these values (matrices) using their keys to obtain final Sum-Product.

The code for the above



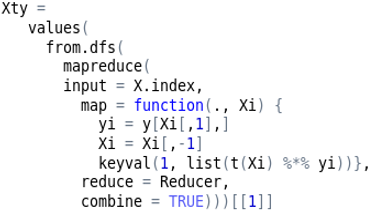


Parameter combine = T is used to combine all pairs (key, value) because mapper emits a single key (1).

Now to compute XTy, following steps are performed

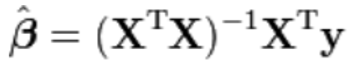
* Feed smaller chunks of ‘x.index’ to the mapper function as object Xi
* Subset these chunks barring the first column
* Subset y values corresponding to these observations and store in yi
* Compute XiTyi for each of these chunks
* Store the result of the above computation as the ‘value’ associated with a specific ‘key’
* Return this ‘key value’ pair and feed it to the reducer function
* Reducer function adds up these values (matrices) using their keys to obtain final Sum-Product.

The code for the above





Once XTX and XTy matrices are computed we compute classically on a single computer as

 by the following command in R .

For multiple variable regression refer “Regression on Boston Data set” code in Appendix.

**References:**

1. <https://www.r-bloggers.com/five-ways-to-handle-big-data-in-r/>
2. <http://www.columbia.edu/~sjm2186/EPIC_R/EPIC_R_BigData.pdf>
3. <https://www.r-bloggers.com/integration-of-r-rstudio-and-hadoop-in-a-virtualbox-cloudera-demo-vm-on-mac-os-x/>

**APPENDIX**

Multiple Linear Regression with R Hadoop on Boston Dataset code.

