Large Data Analysis Using Rhipe/RHadoop

赵扬 (Kevin)

Behavioral Insights and Science Team

11/2/2013

Agenda

1. What is Rhipe/RHadoop?

Introduce the basic structure of R+Hadoop platform

2. Why do we use Rhipe/RHadoop?

Advantages/Disadvantages of using Rhipe

3. How to use Rhipe/RHadoop?

Share 2 specific user cases about data analysis using Rhipe

4. How to learn R+Hadoop more efficiently

Learn Rhipe step by step, easy and fast

1. What is Rhipe/RHadoop?

What is Rhipe/RHadoop?

- Rhipe/RHadoop is just a R package which provides an API(应用程序接口) to use Hadoop.
- RHadoop and Rhipe(“hree-pay”)

➢ RHadoop is comprised of three packages and developed by Revolution Analytics.

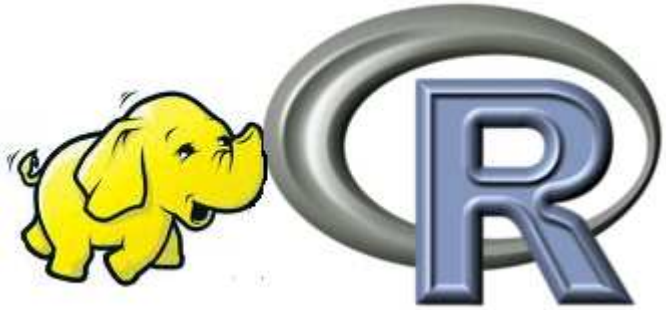
1. *RHDFS*
2. *RHBASE*
3. *rmr*

➢ RHIPE is developed by my classmate Saptarshi Guha in Purdue University:

1. *Same idea of Rhadoop, different API design*
2. *More Integrated with plots(package: Trelliscope)*
3. *I have used Rhipe to do project for 2 years when I pursuing my Phd degree, it works perfect!*

What is Rhipe/RHadoop?

Who is the prefect user of Rhipe/RHadoop?



1. Familiar with R

2. Know some basic statistical knowledge(mean, max, min)

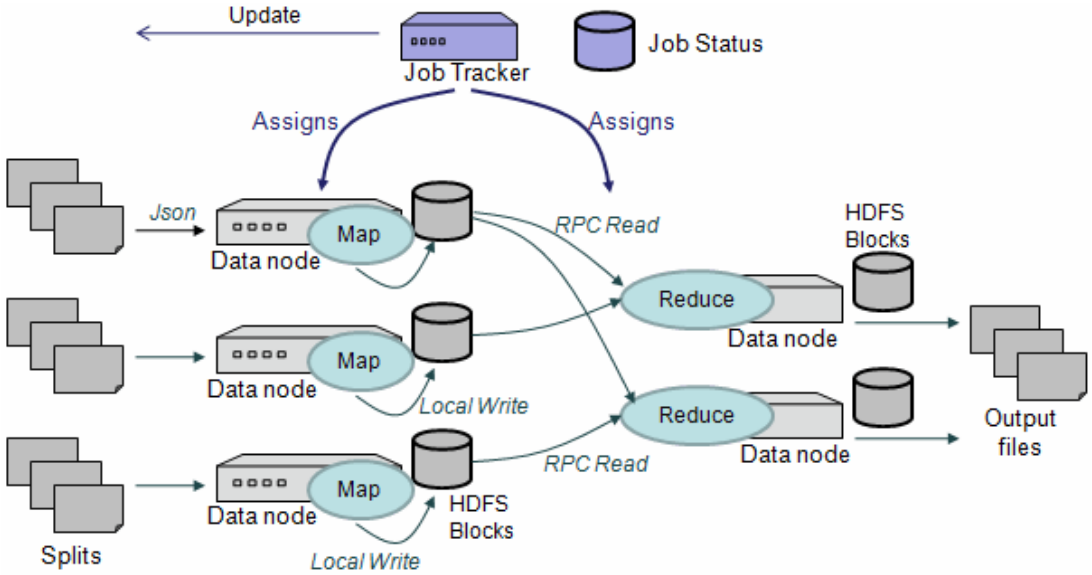
3. Get general idea about Hadoop MapReduce framework

Idea of Hadoop

What is Hadoop? Hadoop is used for parallel computing? (HDFS+MapReduce)

1. Divide(map) and Recombine(reduce)
2. Every mapper/Reducer has key-value pairs

Concrete user case: Parallel compute average student weights in a school by gender



2. Why do we use Rhipe?

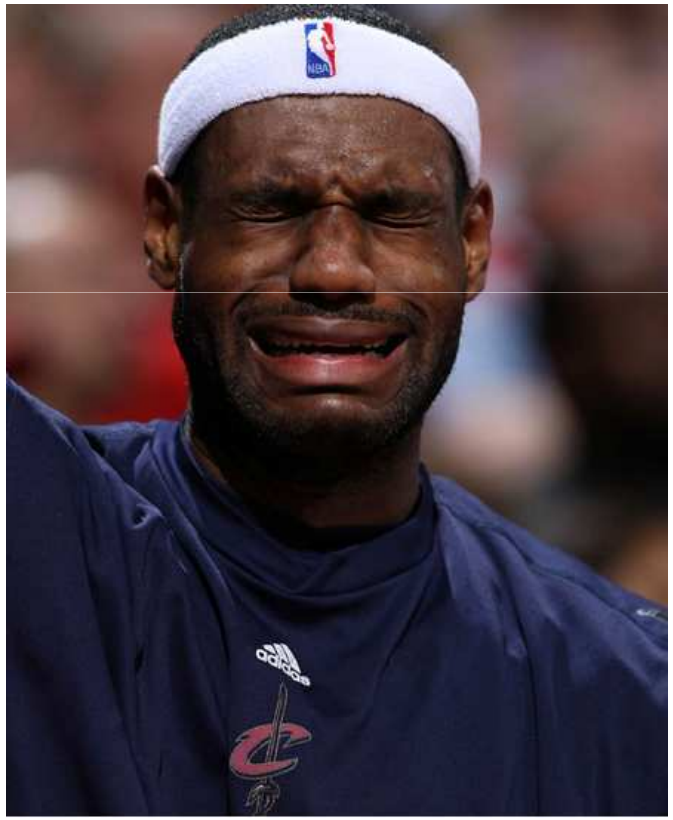
Why do we use Rhipe/RHadoop?

Advantages of Rhipe

	Rhipe	Hadoop(java)	Pig/scoobi/cascading	Snowfall /multicore (R parallel pacakges)
User Friendly	✔	✖	✔	✔
Handle Large data set	✔	✔	✔	✖
Apply statistical algorithm	✔	✖	✖	✔
Large Data visualization	✔	✖	✖	✖

Why do we use Rhipe/RHadoop?

Drawbacks of Rhipe



1. Need to write R code in map reduce format(learning curve)
2. Not as mature and stable as Hadoop(java), pig.
3. Need more formal R packages for statistical algorithm computation in large data set.

3. How to use Rhipe ?

How to use Rhipe/RHadoop?

- Main page is here: http://www.datadr.org/index.html
- Rhipe installation

1. Install **Hadoop** on clusters
2. Install **R** as a shared library. This must be either installed on each of the nodes, or packaged as a zip to be passed to the nodes for each job.
3. Install **Google Protocol Buffers**
4. Set **Environment Variables** like HADOOP Path and R Path
5. Install **Rhipe package**

Word Count example using Rhipe

```
1.  example=list()
2.  # map step
3.  example$map = expression({
4.      words = unlist(strsplit(unlist(map.values), “ “))
5.      lapply(words,function(r){rhcollect(r,1)})
6.  })
7.  #reduce step
8.  example$reduce = expression(
9.      pre = { total = 0 },
10.     reduce = { total = total + sum(unlist(reduce.values)) },
11.     post = { rhcollect(reduce.key,total) }
12.  )
13. #set arguments and run
14. example$ifolder = input_path
15. example$ofolder = output_path
16. example$inout = c("text" ,"sequence")
17. example$jobname = "wordCount”
18. example$zips = zips
19. mr = do.call("rhmr",example)
20. ex = rhex(mr)
```

More complex case to use Rhipe

- Background: We monitored DNS transactions from J-root server in US.
- Data set:
  - 3 days period(Do analysis on 6 months finally)
  - 109,341,181 transactions.
- Goal:
  - Monitor data status
  - Find out Hacker attacks if any

More complex case to use Rhipe

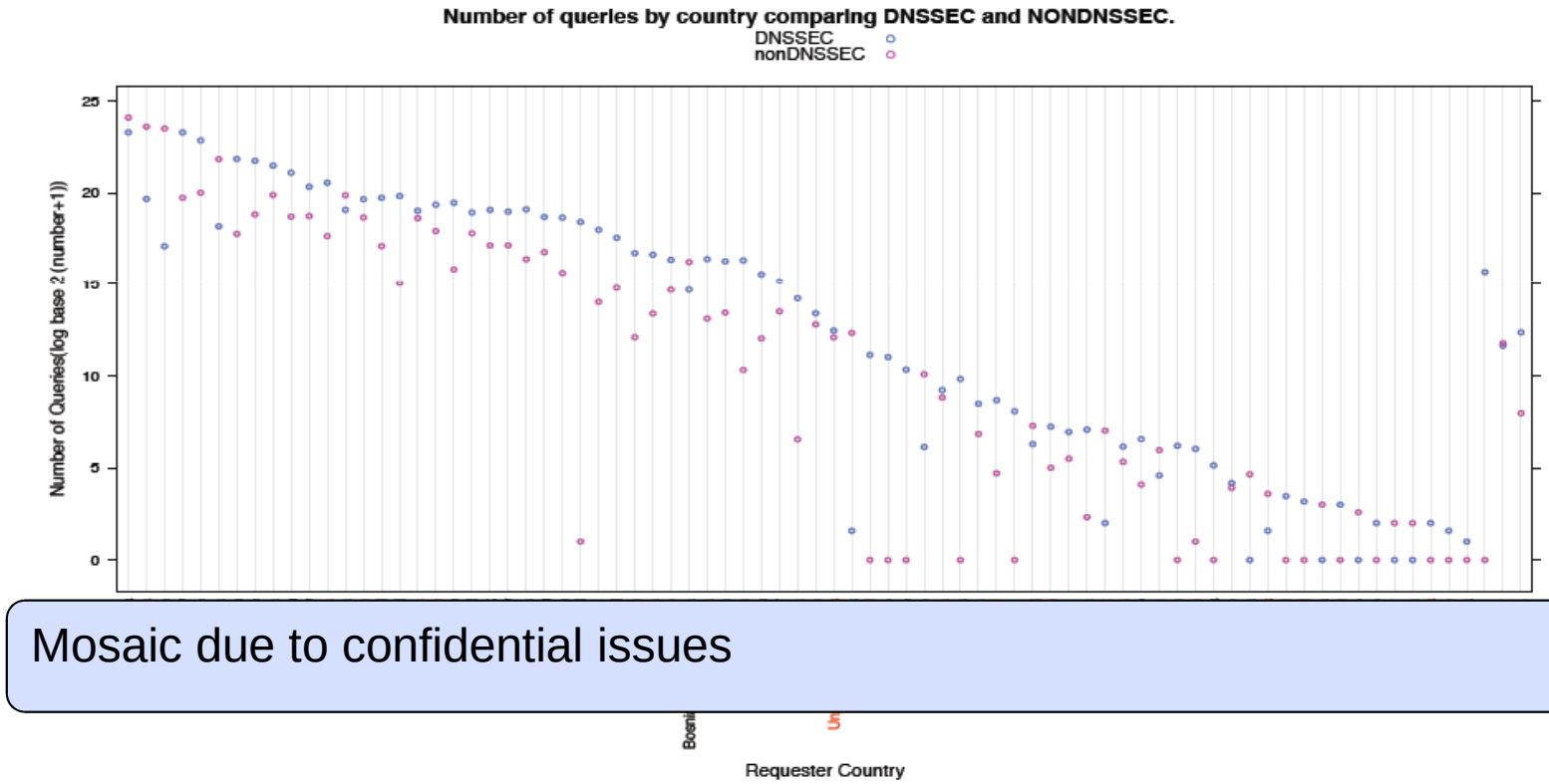
Steps to do data analysis in Rhipe for very large data set(3 steps):

1. Read raw text files into HDFS and create R data base using Rhipe

2. Grab key-value pairs from data base and do map-reduce jobs to get summary results

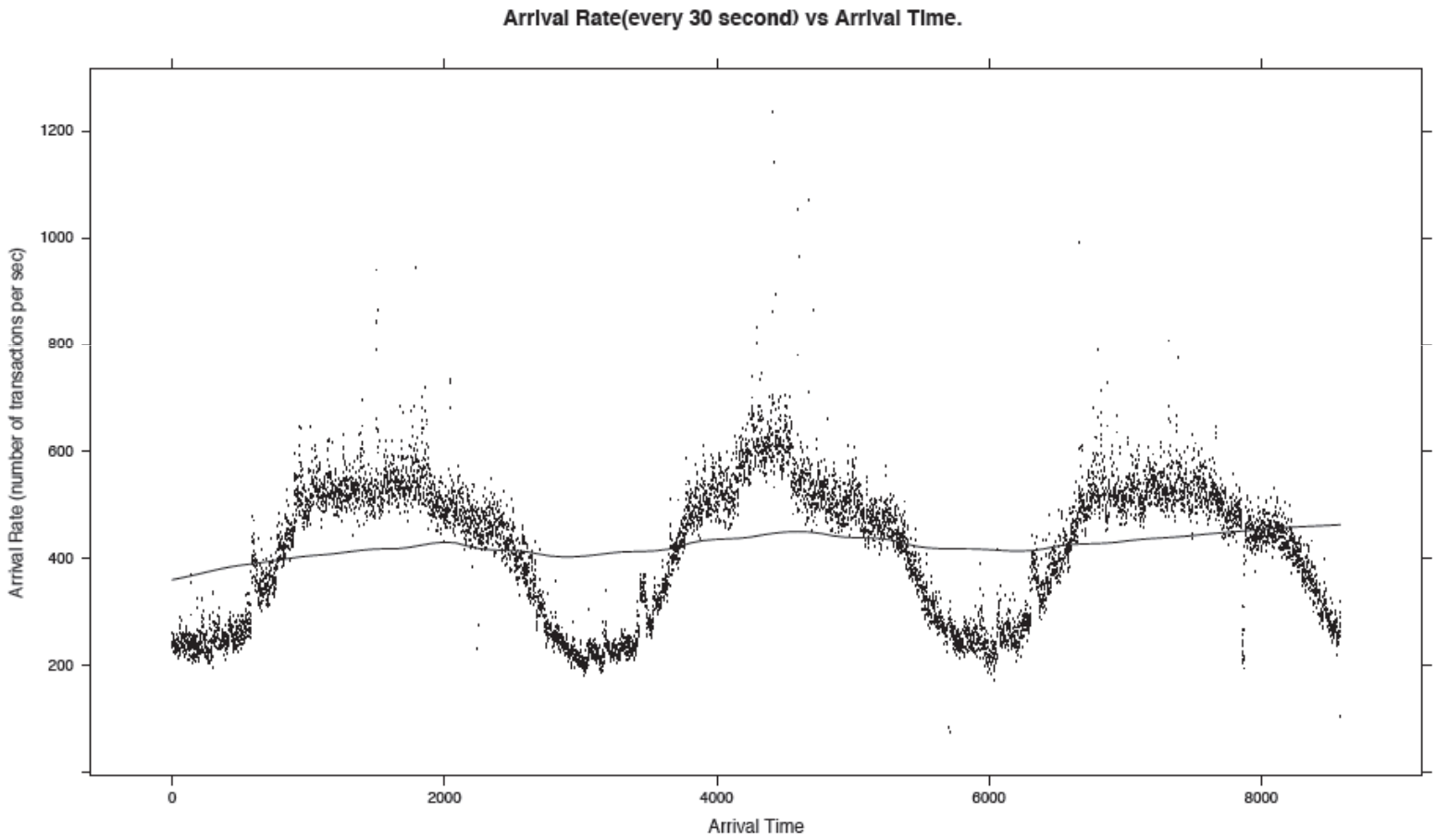
3. Data visualization

More complex case to use Rhipe



Mosaic due to confidential issues

My previous projects using Rhipe

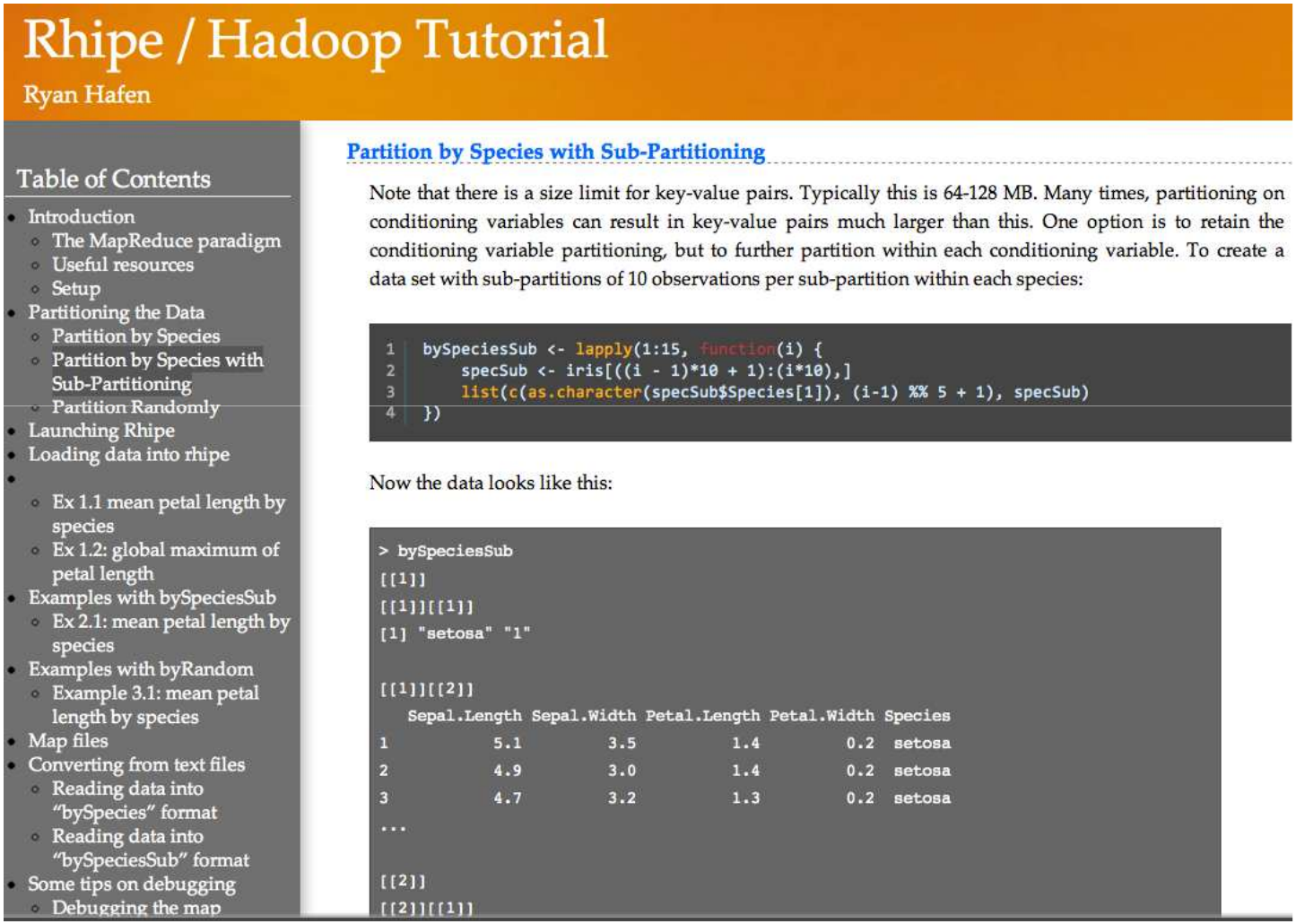


4. How to learn Rhipe more efficiently?

How to learn R+Hadoop more efficiently

1. start with Ryan Hafen's manual(easy to interpret):

http://ml.stat.purdue.edu/rhafen/rhipe/



Rhipe / Hadoop Tutorial

Ryan Hafen

Table of Contents

- Introduction
  - The MapReduce paradigm
  - Useful resources
  - Setup
- Partitioning the Data
  - Partition by Species
  - Partition by Species with Sub-Partitioning
  - Partition Randomly
- Launching Rhipe
- Loading data into rhipe
  - Ex 1.1 mean petal length by species
  - Ex 1.2: global maximum of petal length
- Examples with bySpeciesSub
  - Ex 2.1: mean petal length by species
- Examples with byRandom
  - Example 3.1: mean petal length by species
- Map files
- Converting from text files
  - Reading data into "bySpecies" format
  - Reading data into "bySpeciesSub" format
- Some tips on debugging
  - Debugging the map

Partition by Species with Sub-Partitioning

Note that there is a size limit for key-value pairs. Typically this is 64-128 MB. Many times, partitioning on conditioning variables can result in key-value pairs much larger than this. One option is to retain the conditioning variable partitioning, but to further partition within each conditioning variable. To create a data set with sub-partitions of 10 observations per sub-partition within each species:

```
1  bySpeciesSub <- lapply(1:15, function(i) {
2      specSub <- iris[((i - 1)*10 + 1):(i*10),]
3      list(c(as.character(specSub$Species[1]), (i-1) %% 5 + 1), specSub)
4  })
```

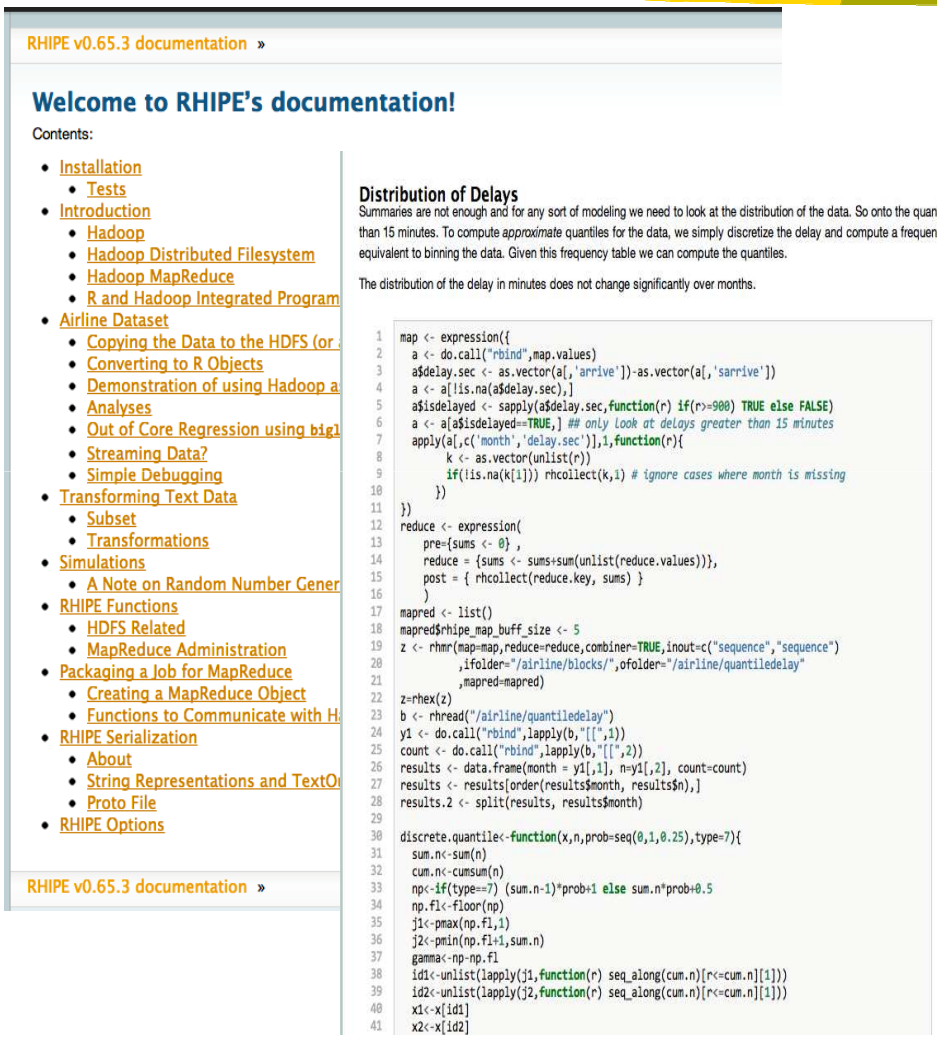
Now the data looks like this:

```
> bySpeciesSub
[[1]]
[[1]][[1]]
[1] "setosa" "1"

[[1]][[2]]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
...

[[2]]
[[2]][[1]]
```

How to learn R+Hadoop more efficiently

2. Go through examples from Rhipe manual written by Saptarshi Guha

http://www.datadr.org/doc/index.html



RHIPE v0.65.3 documentation »

Welcome to RHIPE's documentation!

Contents:

- Installation
  - Tests
- Introduction
  - Hadoop
  - Hadoop Distributed Filesystem
  - Hadoop MapReduce
  - R and Hadoop Integrated Program
- Airline Dataset
  - Copying the Data to the HDFS (or
  - Converting to R Objects
  - Demonstration of using Hadoop a
  - Analyses
  - Out of Core Regression using bigl
  - Streaming Data?
  - Simple Debugging
- Transforming Text Data
  - Subset
  - Transformations
- Simulations
  - A Note on Random Number Gener
- RHIPE Functions
  - HDFS Related
  - MapReduce Administration
- Packaging a Job for MapReduce
  - Creating a MapReduce Object
  - Functions to Communicate with H
- RHIPE Serialization
  - About
  - String Representations and TextO
  - Proto File
- RHIPE Options

RHIPE v0.65.3 documentation »

Distribution of Delays

Summaries are not enough and for any sort of modeling we need to look at the distribution of the data. So onto the quantiles of the delays. We will look at delays greater than 15 minutes. To compute *approximate* quantiles for the data, we simply discretize the delay and compute a frequency count for the unique values of delay. This is equivalent to binning the data. Given this frequency table we can compute the quantiles.

The distribution of the delay in minutes does not change significantly over months.

```
map <- expression({
  a <- do.call("rbind",map.values)
  a$delay.sec <- as.vector(a[,'arrive'])-as.vector(a[,'sarrive'])
  a <- a[!is.na(a$delay.sec),]
  a$isdelayed <- sapply(a$delay.sec,function(r) if(r>=900) TRUE else FALSE)
  a <- a[a$isdelayed==TRUE,] ## only look at delays greater than 15 minutes
  apply(a[,c('month','delay.sec')],1,function(r){
        k <- as.vector(unlist(r))
        if(!is.na(k[1])) rhcollect(k,1) # ignore cases where month is missing
      })
})
reduce <- expression(
    pre={sums <- 0} ,
    reduce = {sums <- sums+sum(unlist(reduce.values))},
    post = { rhcollect(reduce.key, sums) }
    )
mapred <- list()
mapred$rhipe_map_buff_size <- 5
z <- rhmr(map=map,reduce=reduce,combiner=TRUE,inout=c("sequence","sequence")
          ,ifolder="/airline/blocks/",ofolder="/airline/quantiledelay"
          ,mapred=mapred)
z=rhex(z)
b <- rhread("/airline/quantiledelay")
y1 <- do.call("rbind",lapply(b,"[[",1))
count <- do.call("rbind",lapply(b,"[[",2))
results <- data.frame(month = y1[,1], n=y1[,2], count=count)
results <- results[order(results$month, results$n),]
results.2 <- split(results, results$month)

discrete.quantile<-function(x,n,prob=seq(0,1,0.25),type=7){
  sum.n<-sum(n)
  cum.n<-cumsum(n)
  np<-if(type==7) (sum.n-1)*prob+1 else sum.n*prob+0.5
  np.fl<-floor(np)
  j1<-pmax(np.fl,1)
  j2<-pmin(np.fl+1,sum.n)
  gamma<-np-np.fl
  id1<-unlist(lapply(j1,function(r) seq_along(cum.n)[r<=cum.n][1]))
  id2<-unlist(lapply(j2,function(r) seq_along(cum.n)[r<=cum.n][1]))
  x1<-x[id1]
  x2<-x[id2]
```

How to learn R+Hadoop more efficiently

3. Use Rhipe on your own project.

Thanks