RHadoop Usage Guide

0. Preparation

安裝 RHadoop 相關套件，這些都是前置處理：

```
R
install.packages("RJSONIO")
install.packages("itertools")
 install.packages("digest")
q()

R CMD javareconf
R
install.packages("rJava")
install.packages("Rcpp")
install.packages("functional")
install.packages("stringr")
install.packages("reshape2")
q()
```

資料來源：徐承志同學的網頁
http://michaelhsu.tw/2013/05/01/r-and-hadoop-%E5%88%9D%E9%AB%94%E9%A9%97/

自官方網頁下載最新版 rmr2 、 rhdfs，在本次專題中不會用到 rhbase。
https://github.com/RevolutionAnalytics/RHadoop/wiki/Downloads

安裝程式：

```
R CMD INSTALL rmr2_2.2.1.tar.gz
R CMD INSTALL rhdfs_1.0.6.tar.gz
```

若發現無法安裝，可能是缺少前置套件，請依錯誤提示安裝之；或權限不是 root ，建議以上指令加上 sudo 。

1. K Means Algorithm

載入 RHadoop 套件：

```
library(rmr2)
library(rhdfs)
hdfs.init()
```

K Means 演算法宏觀面：

```
kmeans = function(points, center.count, iterations){
        point.data = to.dfs(points)
        point.count = dim(points)[1]

        # 初始化：隨意指定中心點予資料點
        centers = kmeans.iteration(point.data, point.count, NULL, center.count)
        centers = centers$val
        # 迴圈：K Means 演算法執行處
        for(i in 1:iterations){
        newCenters = kmeans.iteration(point.data, point.count, centers, center.count)
        newCenters = newCenters$val

        # 若新中心點坐標全然不變，表示已達穩定狀態
        # 可跳過之後所有迴圈
        if(isTRUE(all.equal(centers, newCenters))){
                        break
        }
        centers = newCenters
        }
        # 給定中心點，實際分群資料點
        # $key == 中心點，$val == 資料點
        kmeans.iteration(point.data, point.count, centers, NULL)
}
```

K Means 演算法微觀面（迴圈內部）：

```
kmeans.iteration = function(point.data, point.count, centers = NULL, center.count =
NULL){
        from.dfs(mapreduce(input = point.data,
        map = function(k, v){
                if(is.null(centers)){
                        # 隨意指定中心點予資料點
                        keyval(sample(1:center.count, point.count, replace = TRUE), v)
                }
                else{
                        # 距離量測：歐幾里德距離
                        distances = apply(v, 1, function(i){
                        apply(centers, 1, function(c){
                                norm(as.matrix(c - i), type = "F")
                        })
                        })
                        # 指定最近中心點予資料點
```

```
                        indices = apply(distances, 2, which.min)
                        newCenters = t(apply(as.matrix(indices), 1, function(i){ t(centers[i, ]
}))
                        keyval(newCenters, v)
                        #keyval(apply(distances, 2, which.min), v)
                }
        },
        reduce = function(k, vv){
                if(is.null(center.count)){
                        # 輸出最近某中心點的所有資料點
                        keyval(k, vv)
                }
                else{
                          # mean == function(c){ mean(c) }
                        # 計算中心點新坐標：資料點坐標平均
                        keyval(k, t(as.matrix(apply(vv, 2, mean))))
                }
        }
        ))
}
```

載入資料，開始測試程式：

```
# 載入資料：以 Iris 為例
data(iris)
iris.data = iris
# 輸出資料至 iris.out
sink("iris.out")
kmeans(iris.data[, 1:4], 3, 5)
sink()
```

2. Upload Files to GitHub

至 GitHub 官方網站申請帳號，申請後記得按右上角「帳號設定」認證電子郵件信箱。
https://github.com/
依官方網頁指示建立新 repository 。記得不要選取「Initialize this repository with a README」，否則可能此後步驟會失效。

安裝程式 git：

```
yum install git
```

設定 SSH Key 。無此步驟，GitHub 不會隨意讓你上傳檔案：
https://help.github.com/articles/generating-ssh-keys#platform-linux
其中 passphrase 很重要，每次上傳時 GitHub 會問 private key password ，即指這項。這裡可以輸入任意英數組合，但請務必記住。

選定上傳資料夾，開始初始化：

```
git init
```

加入與認證檔案：

```
git add k_means.R
git commit -m “first commit”
```

此時螢幕輸出當有「master」字樣。commit 後方文字用以確認版本，每次應輸入一致。

設定用戶名與電子郵件信箱，同 GitHub 申請帳號與電子郵件信箱：

```
git config --gobal user.name “xxxxx”
git config --gobal user.email “xxxxx@xxx.xxx.xx”
```

建立連線，開始上傳：

```
git remote add origin git@github.com:XXXX/xxxx.git
git push origin master
```

網址可在 repository 網頁中看見。