RHadoop Usage Guide

0. Preparation

安裝 RHadoop 套件 rmr2 與 rhdfs，參考網頁：
http://michaelhsu.tw/2013/05/01/r-and-hadoop-%E5%88%9D%E9%AB%94%E9%A9%97/

1. Word Count

載入 RHadoop 套件：

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

以 Iris data 為例，載入測試資料並存之於 Hadoop distributed file system：

```
iris.data = read.table("iris.data", sep = ",", header = FALSE)
iris.in = to.dfs(iris.data)
```

map()：標記所得花名為 1：

```
map = function(k, v){
        keyval(v[5], 1)
}
```

k：第 k 筆資料列。
v：花名與花之數據，v[5] 存花名。

reduce()：累計同花名標記數：

```
reduce = function(k, vv){
        keyval(k, sum(vv))
}
```

vv：所有相同 key（花名）之 value 陣列。

組合以上項目，執行 Hadoop MapReduce 函數：

```
iris.out = mapreduce(input = iris.in, map = map, reduce = reduce)
```

mapreduce() 回傳值正是計算結果。

從 Hadoop distributed file system 取回結果資料並輸出：

```
from.dfs(iris.out)
```

輸出結果：

```
$key
                V5
1     Iris-setosa
2 Iris-versicolor
3  Iris-virginica

$val
[1] 50 50 50
```

原 Iris data 三種花名確實各占三分之一。

2. K Means

載入 RHadoop 套件：

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

撰寫 K Means 演算法宏觀面：

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

        # 初始化：隨機指定一中心點於各資料點
        centers = kmeans.iteration(point.data, center.count, NULL)
        centers = centers$val

        # 迴圈：K Means 演算法實作
        for(i in 1:iterations){
                newCenters = kmeans.iteration(point.data, center.count, centers)
                newCenters = newCenters $val
                # 若中心點坐標無任何改變，表示答案已定， K Means 演算法可直接結
束
                if(isTRUE(all.equal(newCenters, centers))){
                        break
                }
        }
```

```
        # 已知中心點，分群所有資料點
        # $key == center, $val == point
        kmeans.iteration(point.data, NULL, centers)
}
```

points：資料點集合，為二維矩陣，一列一筆資料。
center.count：中心點數量，事先給定。
iterations：預定迴圈次數。

撰寫 K Means 演算法微觀面：

```
kmeans.iteration = function(point.data, center.count = NULL, centers = NULL){
        from.dfs(mapreduce(input = point.data,
                map = function(k, v){
                        if(is.null(centers)){
                                # 初始化：隨機指定一中心點於資料點
                                keyval(sample(1:center.count, 1), v)
                        }
                        else{
                                # 距離公式：歐幾里得距離
                                distances = apply(centers, 1, function(c){ norm(as.matrix(c -
v), type = "F") })
                                # 指定最近中心點於資料點
                                keyval(centers[which.min(distances), ], v)
                        }
                },
                reduce = function(k, vv){
                        # mean == function(c){ mean(c) }
                        if(is.null(center.count)){
                                # 輸出屬於此中心點之所有資料點
                                keyval(k, vv)
                        }
                        else{
                                # 計算中心點新坐標：所屬資料點平均值
                                keyval(NULL, apply(rbind(vv), 2, mean))
                        }
                }
        ))
}
```

載入測試資料 Iris ，執行 K Means：

```
iris.data = read.table(“iris.data”, sep = “,”, header = FALSE)
kmeans(iris.data[, 1:4], 3, 3)
```

K Means 無須事先知曉 Iris 第五行花名，只取前四行即可。

註：我身邊只有 32-bit CPU 電腦，無任何 64-bit CPU 電腦可測試 K Means 程式。
依據下列網頁問答：
http://stackoverflow.com/questions/14273352/install-rhadoop-on-32-bit-ubuntu
知 RHadoop 實不相援 32-bit CPU 電腦，強裝雖可，執行卻可能有意料外錯誤，例如 sample() 函數執行不正常，所有 map() 均輸出同一數字；而 Word Count 或其結構簡易，得以避過錯誤。此外，我的電腦亦有 BIOS 限制，無法模擬 64-bit CPU 環境，是故我認為這份 RHadoop 應能順利執行，卻苦無印證機會。