Problem Set 5

Xiaoyue Liu 28589009 xiaoyliu@umich.edu

1.Create an HDF5 file containing the TMAX values for every day in 2014 using the GHCND data. Then write an R function that calculates the annual range (maximum minus minimum) for each station, using the data from the HDF5 file. Time how long it takes this function to run. Then calculate the same ranges using the data in-memory and time this calculation. If you cannot load the entire data into memory on your machine, load a fraction, say 10%, then adjust the time by the appropriate factor (e.g. if you load 10% of the data, scale the time by 10 to make it comparable to the HDF5 time).

#First we create the hdf5 file. I did it on the server, it took some time.

```
source("https://bioconductor.org/biocLite.R")
biocLite("rhdf5")
library(rhdf5)
library(data.table)
ghcn = fread("zcat -dc 2014.csv.gz")
#ghcn = fread("C:/Users/Heathtasia/Desktop/2014.csv/2014.csv")
setnames(ghcn, c("station", "date", "vtype", "value", "x1", "x2", "x3", "x4"))
setkey(ghcn, vtype, station)
stations = unique(ghcn[, station])
H5close()
fname = "ghcn_2014.hdf5"
h5createFile(fname)
for (vt in c("TMIN", "TMAX", "PRCP")) {
   h5createGroup(fname, vt)
   for (j in 1:length(stations)) {
      st = stations[j]
      if (j %% 100 == 0) {
         print(j)
      }
      dd = ghcn[.(vt, st), .(date, value)]
      dd = as.matrix(dd)
      dname = sprintf("%s/%s", vt, st)
      h5createDataset(fname, dname, dim(dd), level=7)
      h5write(dd, fname, dname)
   }
}
H5close()
```

#Then we compute the range using the hdf5 file

```
contents = h5ls(fname)
contents = contents[2:dim(contents)[1],]
stations = unique(contents$name)
```

```
dfr = data.frame()
t_hfd5=proc.time()
for (j in 1:length(stations)) {

   dx = h5read(fname, sprintf("TMAX/%s", stations[j]))
   ii=which.max(dx[,2])
   jj=which.min(dx[,2])
   dfr[stations[j], "range"] = (dx[ii, 2]-dx[jj,2]) / 10
}
timer_hdf5=proc.time()-t_hfd5
save(as.numeric(timer_hdf5),filefile="hdf5_timer.RData")
q(save="no")
```

>timer_hdf5
[1]229.43 45.22 277.01

#Try compute the range using the data in memory

```
ghcnd2014=fread("C:/Users/Heathtasia/Desktop/2014.csv/2014.csv")
setnames(ghcnd2014, c("station", "date", "vtype", "value", "x1", "x2", "x3", "x4"))
ghcnd2014=ghcnd2014[ghcnd2014$vtype=="TMAX",]
t_mem=proc.time()
GHCN2014=ghcnd2014[,(max(value)-min(value))/10,by="station"]
timer_mem=proc.time()-t_mem
GHCN2014=GHCN2014[order(GHCN2014$station,1)]
```

> timer_mem
0.17 0.00 0.18

The range calculated is the same. Clearly using data in memory is much faster.

2. Use fread from the data.table package to load the 2014 GHCND data to memory from both a zipped file (using zcat or gzip -dc to decompress on-the-fly), and then from an uncompressed copy of the file. Time these two operations and compare.

```
library(data.table)
#Calculate the time for reading from zip file
t1=proc.time()
ghcn = fread("zcat -dc 2014.csv.gz")
timer1=as.numeric(proc.time()-t1)
#Calculate the time for reading from uncompressed file
t2=proc.time()
ghcn=fread("2014.csv")
timer2=as.numeric(proc.time()-t2)
```

Run the program on the server and I got the following result.

>timer1

```
[1] 19.284    0.637 26.685    6.296    0.432
> timer2
[1] 19.493    0.553 20.075    0.000    0.000
```

Reading from the uncompressed file require a little bit more time than reading from the zip file however it requires less time for CPU to execute the instructions.

3. Load the 2014 GHCND data into a data.table object (if you cannot load the entire data set on your machine load as large a fraction of it as possible). Then use dplyr's group_by and summarise to compute the monthly means for every station. Do the same thing using the native data.frame grouping method. Time the two methods. Also assess whether the results of the two calculations are equivalent.

Run the following program on my own laptop. I loaded all the data into memory.

```
#Prepare for dataset
setnames(ghcn, c("station", "date", "vtype", "value", "x1", "x2", "x3", "x4"))
#Get the month column
ghcn$month=as.numeric(substring(ghcn$date,5,6))
#Load the 2014 GHCND data into a data.table object
GHCN=data.table(ghcn)
#Calculate the time using dplyr for grouping and calculate means
library(dplyr)
t3=proc.time()
summarise(group_by(GHCN,station,month),meantmp=mean(value))
timer3=as.numeric(proc.time()-t3)
#Calculate the time using native data.frame grouping and calculate means
t4=proc.time()
GHCN[,mean(value),by=c("station","month")]
timer4=as.numeric(proc.time()-t4)
```

I got the following result.

```
> timer3
[1]   4.17    5.72   14.52      NA      NA
> timer4
[1]   2.19    0.35    2.56      NA      NA
```

When using data as a data.table object, the native data.frame grouping method runs faster than using group_by in the dplyr. The means are the same.

```
> summarise(group_by(GHCN,station,month),meantmp=mean(value))
Source: local data table [417,353 x 3]
Groups: station

       station month  meantmp
         (chr) (dbl)    (dbl)
1  USS0005N04S     1 17.93548
2  USS0005N04S     2 31.33214
3  USS0005N04S     3 35.56352
4  USS0005N04S     4 44.46000
5  USS0005N04S     5 56.62258
6  USS0005N04S     6 78.40000
7  USS0005N04S     7 79.07097
8  USS0005N04S     8 71.63548
9  USS0005N04S     9 66.47667
10 USS0005N04S    10 53.45484
```

```
> GHCN[,mean(value),by=c("station","month")]
            station month        V1
     1: USS0005N04S     1  17.93548
     2: USS0005N04S     2  31.33214
     3: USS0005N04S     3  35.56352
     4: USS0005N04S     4  44.46000
     5: USS0005N04S     5  56.62258
    ---
417349: FIE00144121     9  87.46667
417350: FIE00144121    10  17.24138
417351: FIE00144121    11 -10.60000
417352: FIE00144121    12  -9.75000
417353: USC00142660    12   1.00000
```