732A54 - Big Data Analytics

BDA1 Lab

Sridhar Adhikarla (sriad858), Obaid Ur Rehman (obaur539)

Question 1:

- What are the lowest and highest temperatures measured each year for the period 1950-2014. Provide the lists sorted in the descending order with respect to the maximum temperature. In this exercise you will use the `temperature-readings.csv` file.
- Non-parallelized program in Python to find the maximum temperatures for each year without using Spark. In this case you will run the program using: `python script.py`

Code (Non-parallelized):

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

import time

iFile = '../data/temperature-readings.csv'
oFile = 'outputs/out1_b.csv'
fromYear = 1950
toYear = 2014

start = time.time()
print('Running Python MinMaxTempExtractor:\nFrom %s To %s\nInput file: %s\nOutput file: %s'
      % (fromYear, toYear, iFile, oFile))
temp_dict = dict()

first_line = False
with open(iFile) as f:
    for l in f:
        if first_line == False:
            print(l)
            first_line = True

        line = l.split(";")
        year = int(line[1].split("-")[0])
        if year >= fromYear and year <= toYear:
            temp = temp_dict.get(year)
            station = line[0];
            curr_temp = float(line[3]);
            if not temp:
                #1st list for min temp and 2nd list for max temp
                temp_dict[year] = [[station, curr_temp], [station, curr_temp]]
            else:
                min_temp = float(temp[0][1])
```

```
                max_temp = float(temp[1][1])
                if curr_temp < min_temp:
                    temp[0][0] = station
                    temp[0][1] = curr_temp
                if curr_temp > max_temp:
                    temp[1][0] = station
                    temp[1][1] = curr_temp

#sort temperatures descending by max temp
sorted_temp = temp_dict.items()
sorted_temp.sort(key=lambda x: x[1][1][1], reverse=True)

#write the output to file.
with open(oFile,'wb+') as f:
    f.write('%s,%s,%s,%s,%s\n' % ("Year", "Station_Min", "Min_Temp", "Station_Max", "Max_Temp"))
    for i in sorted_temp:
        f.write('%s,%s,%s,%s,%s\n' % (i[0], i[1][0][0], i[1][0][1], i[1][1][0], i[1][1][1]))

end = time.time()
print('Done in %s seconds' % (end - start))
```

The lowest and highest temperatures:

```
out1_b = read.csv("outputs/out1_b.csv")
head(out1_b)
```

```
##   Year Station_Min Min_Temp Station_Max Max_Temp
## 1 1975      157860    -37.0       86200     36.1
## 2 1992      179960    -36.1       63600     35.4
## 3 1994      179960    -40.5      117160     34.7
## 4 2010      191910    -41.7       75250     34.4
## 5 2014      192840    -42.5       96560     34.4
## 6 1989      166870    -38.2       63050     33.9
```

Code (Spark):

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

import time
from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'outputs/out1_a'
fromYear = 1950
toYear = 2014

sc = SparkContext(appName="Lab1_Q1A_SparkJob")
```

```
start = time.time()

lines = sc.textFile(iFile)
lines = lines.map(lambda a: a.split(";"))
lines = lines.filter(lambda x: int(x[1][0:4]) >= 1950 and int(x[1][0:4]) <= 2014)
temperatures = lines.map(lambda x: (x[1][0:4], (x[0], float(x[3]))))

#Custom MIN/MAX Function
my_min = (lambda x, y: x if x[1] < y[1] else y)
my_max = (lambda x, y: x if x[1] > y[1] else y)

minTemperatures = temperatures.reduceByKey(my_min)
maxTemperatures = temperatures.reduceByKey(my_max)
minMaxTemp = minTemperatures.union(maxTemperatures).reduceByKey(lambda x,y: (x[0],x[1],y[0],y[1]))

sortedMinMaxTemp = minMaxTemp.sortBy(ascending=False, keyfunc=lambda a: a[1][3])
sortedMinMaxTempCsv = sortedMinMaxTemp.map(lambda a: '%s,%s,%s,%s,%s' % (a[0], a[1][0], a[1][1], a[1][2
sortedMinMaxTempCsv.coalesce(1).saveAsTextFile(oFile)

end = time.time()
print('Script Lab1_Q1_A Done in %s seconds' % (end - start))
```

The lowest and highest temperatures:

```
out1_a = read.csv("outputs/out1_a/part-00000", header = FALSE)
colnames(out1_a) = c("year","station","min","station","max")
head(out1_a)
```

```
##   year station   min station  max
## 1 1975  157860 -37.0   86200 36.1
## 2 1992  179960 -36.1   63600 35.4
## 3 1994  179960 -40.5  117160 34.7
## 4 2014  192840 -42.5   96560 34.4
## 5 2010  191910 -41.7   75250 34.4
## 6 1989  166870 -38.2   63050 33.9
```

The spark parallel version takes much less time than the non-parallel(serial execution) implemetation for the same calculation.

Question 2:

- Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees. Repeat the exercise, this time taking only distinct readings from each station. That is, if a station reported a reading above 10 degrees in some month, then it appears only once in the count for that month.

Code:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-


from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'outputs/out2_a'
oFile2 = 'outputs/out2_b'
fromYear = 1950
toYear = 2014
target_temp = 10

sc = SparkContext(appName="Lab1_Q2_SparkJob")

lines = sc.textFile(iFile)
lines = lines.map(lambda a: a.split(";"))
observations = lines.filter(lambda observation:(int(observation[1][:4]) >= fromYear and int(observation

#Q2a. Year-month, number
temperatures = observations.map(lambda observation:
                                    (observation[1][:7], (float(observation[3]), 1))) \
                                .filter(lambda (month, (temp, count)): temp > target_temp)

reading_counts = temperatures.reduceByKey(lambda (temp1, count1), (temp2, count2):
                                            (temp1, count1 + count2)) \
                                  .map(lambda (month, (temp, count)):(month, count))

reading_counts.repartition(1).saveAsTextFile(oFile)

#Q2b. Year-month, distinct number
station_temperatures = observations.map(lambda observation:
                                            (observation[1][:7],
                                             (observation[0], float(observation[3])))) \
                                        .filter(lambda (month, (station, temp)): temp > target_temp)

year_station = station_temperatures.map(lambda (month, (station, temp)): (month, (station, 1))).distinc

reading_counts = year_station.reduceByKey(lambda (station1, count1), (station2, count2):
                                            (station1, count1 + count2)) \
                                  .map(lambda (month, (station, count)): (month, count))

reading_counts.repartition(1).saveAsTextFile(oFile2)
```

Count of the readings above 10 degrees for each month:

```
out2_a = read.csv("outputs/out2_a/part-00000", header = FALSE)
head(out2_a)
```

```
##            V1      V2
## 1 (u'1981-03'    395)
## 2 (u'1974-07'  66277)
## 3 (u'2003-05'  48264)
## 4 (u'1981-10'   9882)
## 5 (u'1983-09'  38692)
## 6 (u'1987-05'  17191)
```

Distinct count of the readings above 10 degrees for each month:

```
out2_b = read.csv("outputs/out2_b/part-00000", header = FALSE)
head(out2_b)
```

```
##            V1    V2
## 1 (u'1999-07'  329)
## 2 (u'1997-05'  319)
## 3 (u'1995-03'   59)
## 4 (u'1981-11'   45)
## 5 (u'1957-01'    2)
## 6 (u'1954-04'   65)
```

Question 3:

- Find the average monthly temperature for each available station in Sweden. Your result should include average temperature for each station for each month in the period of 1960-2014. Bear in mind that not every station has the readings for each month in this timeframe.

Code:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
oFile = 'outputs/out3'
fromYear = 1960
toYear = 2014

sc = SparkContext(appName="Lab1_Q3_SparkJob")

lines = sc.textFile(iFile)
lines = lines.map(lambda a: a.split(";"))

observations = lines.filter(lambda observation:
                                    (int(observation[1][:4]) >= fromYear and
                                     int(observation[1][:4]) <= toYear))

stationDailyTemps = observations.map(lambda observation:
                                            ((observation[1], observation[0]),
                                             (float(observation[3]), float(observation[3]))))

stationDailyMinMaxTemps = stationDailyTemps.reduceByKey(lambda
                                                        (mintemp1, maxtemp1),
                                                        (mintemp2, maxtemp2):
                                                        (min(mintemp1, mintemp2),
                                                         max(maxtemp1, maxtemp2)))

stationMonthlyAvgTemps = stationDailyMinMaxTemps.map(lambda ((day, station), (mintemp, maxtemp)):
                                                        ((day[:7], station), (sum((mintemp, maxtemp))
                                                    .reduceByKey(lambda (temp1, count1), (temp2, coun
                                                                (temp1 + temp2, count1 + count2)) \
                                                    .map(lambda ((month, station), (temp, count)):
                                                        ((month, station), temp / float(count)))

stationMonthlyAvgTemps.repartition(1).saveAsTextFile(oFile)
```

Station average monthly temperature:

```r
out3 = read.csv("outputs/out3/part-00000", header = FALSE)
colnames(out3) = c("Month","Station","Avg Temp")
head(out3)
```

```
##          Month     Station              Avg Temp
## 1 ((u'1987-02'  u'166870')  -15.850000000000003)
## 2 ((u'2012-09'   u'77220')    13.26333333333333)
## 3 ((u'2002-06'   u'62530')   15.078947368421055)
## 4 ((u'1995-12'  u'104580')   -10.32258064516129)
## 5 ((u'1976-05'   u'72290')   10.500000000000002)
## 6 ((u'2005-12'   u'71190')    3.141935483870968)
```

Question 4:

- Provide a list of stations with their associated maximum measured temperatures and maximum measured daily precipitation. Show only those stations where the maximum temperature is between 25 and 30 degrees and maximum daily precipitation is between 100 mm and 200 mm.

Code:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pyspark import SparkContext

iFile = 'data/temperature-readings.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'outputs/out4'

sc = SparkContext(appName="Lab1_Q4_SparkJob")

#Max Temperatures
temperatures = sc.textFile(iFile)
temperatures = temperatures.map(lambda a: a.split(";"))
temperatures = temperatures.map(lambda x: (x[0], float(x[3])))
maxTemperatures = temperatures.reduceByKey(max)
maxTemperatures = maxTemperatures.filter(lambda a: a[1] > 25 and a[1] < 30)

#Max Precipitations
precipitations = sc.textFile(iFile2)
precipitations = precipitations.map(lambda a: a.split(";"))
precipitations = precipitations.map(lambda x: (x[0]+','+x[1], float(x[3])))
maxPrecipitations = precipitations.reduceByKey(lambda v1, v2: v1+v2)
maxPrecipitations = maxPrecipitations.filter(lambda a: a[1] >= 100 and a[1] <= 200) \
.map(lambda x: (x[0].split(",")[0], x[1])).reduceByKey(max)

#Merged Max Temperatures/Precipitations
maxTempPrec = maxTemperatures.leftOuterJoin(maxPrecipitations)

maxTempPrecCsv = maxTempPrec.map(lambda a: '%s,%s,%s' % (a[0], a[1][0], a[1][1]))

maxTempPrec.coalesce(1).saveAsTextFile(oFile)
```

stations maximum measured daily temperatures/precipitation:

```
out4 = read.csv("outputs/out4/part-00000", header = FALSE)
colnames(out4) = c("station","maxTemp","maxPrec")
head(out4)
```

```
##      station maxTemp maxPrec
## 1 (u'128510'   (29.5  None))
## 2 (u'192830'   (29.5  None))
```

```
## 3  (u'84660'   (27.6  None))
## 4 (u'139110'   (29.0  None))
## 5 (u'161670'   (25.7  None))
## 6 (u'166940'   (27.9  None))
```

Question 5:

- Calculate the average monthly precipitation for the Ostergotland region (list of stations is provided in the separate file). In order to do this, you will first need to calculate the total monthly precipitation for each station before calculating the monthly average (by averaging over stations).

Code:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pyspark import SparkContext

iFile = 'data/stations-Ostergotland.csv'
iFile2 = 'data/precipitation-readings.csv'
oFile = 'outputs/out5'

sc = SparkContext(appName="Lab1_Q5_SparkJob")

ostergotlandStations = sc.textFile(iFile)
ostergotlandStations = ostergotlandStations \
.map(lambda line: line.split(";")).map(lambda x: int(x[0])).distinct().collect()

isOstergotlandStation = (lambda s: s in ostergotlandStations)

precipitations = sc.textFile(iFile2)

daily_precipitations = precipitations.map(lambda line: line.split(";")) \
.filter(lambda x: isOstergotlandStation(int(x[0])))

daily_precipitations = daily_precipitations \
.map(lambda x: (x[0]+','+x[1], float(x[3]))).reduceByKey(lambda a, b: a + b)

st_monthly_precipitations = daily_precipitations \
.map(lambda x:(x[0].split("-")[0]+','+x[0].split("-")[1], (x[1], 1)))

st_monthly_precipitations = st_monthly_precipitations \
.reduceByKey(lambda v1, v2: (v1[0] + v2[0], v1[1] + v2[1]))

st_monthly_precipitations = st_monthly_precipitations\
.map(lambda x: (x[0].split(",")[1] + "," + x[0].split(",")[2], (x[1][0], 1)))

monthly_precipitations = st_monthly_precipitations \
.reduceByKey(lambda v1, v2: (v1[0] + v2[0], v1[1] + v2[1]))

monthly_precipitations = monthly_precipitations \
.map(lambda x: (x[0], x[1][0] / x[1][1]))
print(monthly_precipitations.take(5))

monthly_precipitations_csv = monthly_precipitations \
.map(lambda a: '%s,%s' % (a[0], a[1]))
```

```
monthly_precipitations_csv.coalesce(1).saveAsTextFile(oFile)
```

Ostergotland average monthly precipitation:

```
out5 = read.csv("outputs/out5/part-00000", header = FALSE)
colnames(out5) = c("year","month","avgPrec")
head(out5)
```

```
##   year month  avgPrec
## 1 2011     3 19.83333
## 2 2011     6 88.35000
## 3 2012    12 66.93333
## 4 2001    11 26.38333
## 5 1995     2 31.80000
## 6 2011     9 52.56667
```

Question 6:

- Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in Ostergotland with long-term monthly averages in the period of 1950-1980.

Code:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from pyspark import SparkContext

iFile = 'data/stations-Ostergotland.csv'
iFile2 = 'data/temperature-readings.csv'
oFile = 'outputs/out6'
fromYear = 1960
toYear = 2014

sc = SparkContext(appName="Lab1_Q6_SparkJob")

ostergotlandStations = sc.textFile(iFile)
ostergotlandStations = ostergotlandStations.map(lambda line: line.split(";")).map(lambda x: int(x[0])).c

isOstergotlandStation = (lambda s: s in ostergotlandStations)

temperatures = sc.textFile(iFile2).map(lambda line: line.split(";")). \
filter(lambda x: isOstergotlandStation(int(x[0])) and int(x[1][0:4]) >= fromYear and int(x[1][0:4]) <= t

monthlyAvgTemps = temperatures.map(lambda obs:
                                          ((obs[1], int(obs[0])),
                                                 (float(obs[3]), float(obs[3])))) \
                                           .reduceByKey(lambda (mint1, maxt1), (mint2, maxt2):
                                                        (min(mint1, mint2), max(maxt1, maxt2))) \
                                           .map(lambda ((day, station), (mint, maxt)):
                                                (day[:7], (mint + maxt, 2))) \
                                           .reduceByKey(lambda (temp1, count1), (temp2, count2):
                                                        (temp1 + temp2, count1 + count2)) \
                                           .map(lambda (month, (temp, count)):
                                                (month, temp / float(count)))

monthlyLongtermAvgTemps = monthlyAvgTemps.filter(lambda (month, temp):
                                                         int(month[:4]) <= 1980) \
                                         .map(lambda (month, temp):
                                              (month[-2:], (temp, 1))) \
                                         .reduceByKey(lambda (temp1, count1), (temp2, count2):
                                                      (temp1 + temp2, count1 + count2)) \
                                         .map(lambda (month, (temp, count)):
                                              (month, temp / float(count)))

month_temp = {month: temp for month, temp in monthlyLongtermAvgTemps.collect()}

monthlyAvgTemps = monthlyAvgTemps.map(lambda (month, temp):
```

```
                                        (month, abs(temp) - abs(month_temp[month[-2:]]))) \
                                    .sortBy(ascending=True, keyfunc=lambda (month, temp): month)

monthlyAvgTempsCsv = monthlyAvgTemps.map(lambda a: '%s,%s,%s' % (a[0].split('-')[0], a[0].split('-')[1]
monthlyAvgTempsCsv.repartition(1).saveAsTextFile(oFile)
```

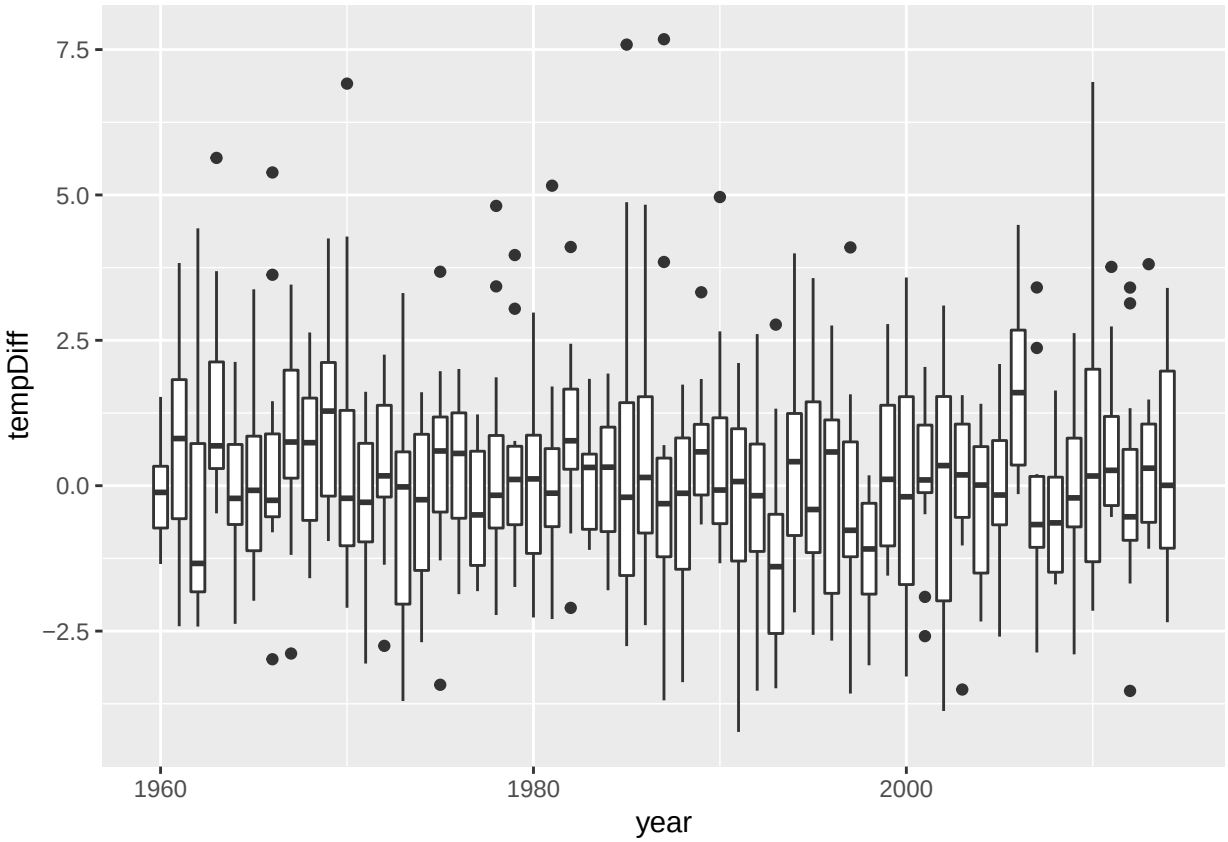
Ostergotland average monthly temperature compared with long-term monthly averages:

```
out6 = read.csv("outputs/out6/part-00000", header = FALSE)
colnames(out6) = c('year', 'month', 'tempDiff')
head(out6)
```

```
##   year month   tempDiff
## 1 1960     1  0.7614638
## 2 1960     2  1.5272002
## 3 1960     3  0.1920504
## 4 1960     4 -0.6420244
## 5 1960     5  0.1577730
## 6 1960     6 -0.2373111
```

plot:

```
library(ggplot2)
ggplot(out6) +
  geom_boxplot(aes(x = year, y = tempDiff, group = year))
```


7.5
5.0
2.5
0.0
−2.5
tempDiff
1960
1980
2000
year