Big Data Analytics Lab 1

Omkar Bhutra (omkbh878) and Vinay Bengaluru (vinbe289)

7 May 2019

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. a) Extend the program to include the station number (not the station name) where the maximum/minimum temperature was measured. b) (not for the SparkSQL lab) Write the 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 This program will read the local file (not from HDFS). The local file is available under /nfshome/hadoop__examples/shared__data/temperatures-big.csv. How does the runtime compare to the Spark version? Use logging (add the –conf spark.eventLog.enabled=true flag) to check the execution of the Spark program. Repeat the exercise, this time using temperatures-big.csv file available on hdfs. Explain the differences and try to reason why such runtimes were observed.

1.a) Minimum and Maximum temperature by year in Spark

```
from pyspark import SparkContext
sc = SparkContext(appName="MinMaxTempExtractorSparkJob")
lines = sc.textFile("/user/x_omkbh/data/temperature-readings.csv")
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]))))
min_temperatures = temperatures.reduceByKey(lambda x, y: x if x[1] < y[1] else y)
max_temperatures = temperatures.reduceByKey(lambda x, y: x if x[1] > y[1] else y)
temperaturesminmax = min_temperatures.union(max_temperatures).reduceByKey(lambda x,y:
                                                    (x[0],x[1],y[0],y[1]))
temperaturesminmaxsorted = temperaturesminmax.sortBy(ascending=False, keyfunc=lambda a: a[1][3])
temperaturesminmaxsortedcsv = temperaturesminmaxsorted.map(lambda a: '%s,%s,%s,%s,%s' %
                                                    (a[0], a[1][0], a[1][1], a[1][2], a[1][3]))
temperaturesminmaxsortedcsv.coalesce(1).saveAsTextFile("sparkparallelminmax")
```

Execution of code

```
[x_omkbh@heffa1 ~]$ ./runYarn.sh sparkparallel.py

spark-submit --conf spark.eventLog.enabled=true --deploy-mode cluster --master yarn
--num-executors 9 --driver-memory 2g --executor-memory 2g --executor-cores 4 sparkparallel.py

./runYarn-withHistory.sh sparkparallel.py
```

output contents of the file in format (year, station number, minimum temperature,station number, maximum temperature)

```
1975,157860,-37.0,86200,36.1
1992,179960,-36.1,63600,35.4
1994,179960,-40.5,117160,34.7
2014,192840,-42.5,96560,34.4
2010,191910,-41.7,75250,34.4
1989,166870,-38.2,63050,33.9
1982,113410,-42.2,94050,33.8
1968,179950,-42.0,137100,33.7
1966,179950,-49.4,151640,33.5
1983,191900,-38.2,98210,33.3
2002,169860,-42.2,78290,33.3
1986,167860,-44.2,76470,33.2
1970,179950,-39.6,103080,33.2
2000,169860,-37.6,62400,33.0
1956,160790,-45.0,145340,33.0
1959,159970,-43.6,65160,32.8
1991,179960,-39.3,137040,32.7
2006,169860,-40.6,75240,32.7
1988,170790,-39.9,102540,32.6
2011,179960,-42.0,172770,32.5
1999,192830,-49.0,98210,32.4
1953,183760,-38.4,65160,32.2
2007,169860,-40.7,86420,32.2
1973,166870,-39.3,71470,32.2
1955,160790,-41.2,97260,32.2
2003,179960,-41.5,136420,32.2
2008,179960,-39.3,102390,32.2
2005,155790,-39.4,107140,32.1
1979,112170,-44.0,63600,32.0
1969,181900,-41.5,71470,32.0
2001,112530,-44.0,62400,31.9
1997,179960,-40.2,76420,31.8
1977,179950,-42.5,94180,31.8
2013,179960,-40.7,98210,31.6
2009,179960,-38.5,81370,31.5
2012,191910,-42.7,54990,31.3
1972,167860,-37.5,98200,31.2
1971,166870,-44.3,65130,31.2
1964,166810,-39.5,76430,31.2
1976,192830,-42.2,75040,31.1
1963,181900,-41.0,62410,31.0
1961,181900,-39.5,76000,31.0
1978,155940,-47.7,71470,30.8
1996,155790,-41.7,98210,30.8
1995,182910,-37.6,76420,30.8
1958,159970,-43.0,117160,30.8
1974,166870,-35.6,183980,30.6
1954,113410,-36.0,53650,30.5
1952,192830,-35.5,106100,30.4
1980,191900,-45.0,92400,30.4
1990,166870,-35.0,54330,30.2
2004,166940,-39.7,117160,30.2
1985,166870,-43.4,85450,29.8
```

```
1957,159970,-37.8,75040,29.8
1993,191900,-39.0,132030,29.7
1981,166870,-44.0,65160,29.7
1987,123480,-47.3,105410,29.6
1984,123480,-39.2,105370,29.5
1967,166870,-45.4,75040,29.5
1960,160790,-40.0,173810,29.4
1950,155910,-42.0,75040,29.4
1998,169860,-42.7,63600,29.2
1951,155910,-42.0,75040,28.5
1965,189780,-44.0,116500,28.5
1962,181900,-42.0,76380,27.4
```

1.b) Non-Prallelized code

```
import time, argparse
argumentparseobj = argparse.ArgumentParser(description='Non-parallelized')
argumentparseobj.add_argument('-o', '--output', help='Output file', default='out_temp.csv')
requiredNamed = argumentparseobj.add_argument_group('Required arguments')
requiredNamed.add_argument('-t', '--time', help='time range in years, ex: 1975:2000', required=True)
requiredNamed.add_argument('-i', '--input', help='Input file', required=True)
argumentparse = argumentparseobj.parse_args()
inputfile = argumentparse.input
outputfile = argumentparse.output
years =  argumentparse.time.split(":")
from_year = int(years[0])
to_year = int(years[1])

start_time = time.time()
print('Running Non-Parallelized Python code to find minimum and maximum temperature in each year:\n \
From %s To %s\ninput file: %s\noutput file: %s'
      % (from_year, to_year, inputfile, outputfile))
temperatures_year_dict = dict()
with open(inputfile) as i_file:
    for l in i_file:
        line = l.split(";")
        year = int(line[1].split("-")[0])
        if year >= from_year and year <= to_year:
            temperature = temperatures_year_dict.get(year)
            station_number = line[0];
            measured_temperature = float(line[3]);
            if not temperature:
                #format is in the form: list(list(min temperature),list(max temperature))
                temperatures_year_dict[year] = [[station_number, measured_temperature],
                                                [station_number, measured_temperature]]
            else:
                min = float(temperature[0][1])
                max = float(temperature[1][1])
                if measured_temperature < min:
                    temperature[0][0] = station_number
                    temperature[0][1] = measured_temperature
                if measured_temperature > max:
                    temperature[1][0] = station_number
                    temperature[1][1] = measured_temperature
i_file.close()
#temperatures in descending order by maximum temperature
sortedtemperature = temperatures_year_dict.items()
sortedtemperature.sort(key=lambda x: x[1][1][1], reverse=True)
with open(outputfile,'wb+') as i_file:
    for i in sortedtemperature:
        i_file.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]))
i_file.close()
end_time = time.time()
print('Runtime was %s minutes' % ((end_time - start_time)/60))
```

Execution of code

```
[x_omkbh@heffa2 ~]$ python nonparallel.py -t 1950:2014 -i /nfshome/hadoop_examples/shared_data/temperatu
Running Non-Parallelized Python code to find minimum and maximum temperature in each year:
From 1950 To 2014
Input file: /nfshome/hadoop_examples/shared_data/temperature-readings.csv
Output file: nonparallelminmax
Runtime was 4.95247233311 minutes
```

output contents of the file in format (year, station number, minimum temperature,station number, maximum temperature)

```
1975,157860,-37.0,86200,36.1
1992,179960,-36.1,63600,35.4
1994,179960,-40.5,117160,34.7
2010,191910,-41.7,75250,34.4
2014,192840,-42.5,96560,34.4
1989,166870,-38.2,63050,33.9
1982,113410,-42.2,94050,33.8
1968,179950,-42.0,137100,33.7
1966,179950,-49.4,151640,33.5
1983,191900,-38.2,98210,33.3
2002,169860,-42.2,78290,33.3
1970,179950,-39.6,103080,33.2
1986,167860,-44.2,76470,33.2
1956,160790,-45.0,145340,33.0
2000,169860,-37.6,62400,33.0
1959,159970,-43.6,65160,32.8
1991,179960,-39.3,137040,32.7
2006,169860,-40.6,75240,32.7
1988,170790,-39.9,102540,32.6
2011,179960,-42.0,172770,32.5
1999,192830,-49.0,98210,32.4
1953,183760,-38.4,65160,32.2
1955,160790,-41.2,97260,32.2
1973,166870,-39.3,71470,32.2
2003,179960,-41.5,136420,32.2
2007,169860,-40.7,86420,32.2
```

```
2008,179960,-39.3,102390,32.2
2005,155790,-39.4,107140,32.1
1969,181900,-41.5,71470,32.0
1979,112170,-44.0,63600,32.0
2001,112530,-44.0,62400,31.9
1977,179950,-42.5,94180,31.8
1997,179960,-40.2,74180,31.8
2013,179960,-40.7,98210,31.6
2009,179960,-38.5,81370,31.5
2012,191910,-42.7,54990,31.3
1964,166810,-39.5,76430,31.2
1971,166870,-44.3,65130,31.2
1972,167860,-37.5,137100,31.2
1976,192830,-42.2,75040,31.1
1961,181900,-39.5,76000,31.0
1963,181900,-41.0,62410,31.0
1958,159970,-43.0,117160,30.8
1978,155940,-47.7,71470,30.8
1995,182910,-37.6,76420,30.8
1996,155790,-41.7,96140,30.8
1974,166870,-35.6,167710,30.6
1954,113410,-36.0,53650,30.5
1952,192830,-35.5,106100,30.4
1980,191900,-45.0,161790,30.4
1990,147270,-35.0,54330,30.2
2004,166940,-39.7,117160,30.2
1957,159970,-37.8,75040,29.8
1985,159970,-43.4,85450,29.8
1981,166870,-44.0,65160,29.7
1993,191900,-39.0,132030,29.7
1987,123480,-47.3,105410,29.6
1967,166870,-45.4,75040,29.5
1984,123480,-39.2,105370,29.5
1950,155910,-42.0,75040,29.4
1960,155910,-40.0,173810,29.4
1998,169860,-42.7,63600,29.2
1951,155910,-42.0,75040,28.5
1965,189780,-44.0,116500,28.5
1962,181900,-42.0,76380,27.4
```

The Spark version has a runtime of 2.49 minutes and the non-parallelised version has a runtime of 4.95 minutes on the temperature-readings.csv file. The temperature-big.csv file is read from the shared data folder and both the parallelized as well as the non-parallelized python code is run on it. We find that the non-parallelized code took over 16 minutes to produce results, while the parralelized pySpark code produces a result in 7.5 minutes.

parallelized pySpark code on temperatures-big.csv

```
[x_omkbh@heffa1 ~]$ ./runYarn.sh sparkparallelbig.py

spark-submit --conf spark.eventLog.enabled=true --deploy-mode cluster --master yarn
--num-executors 9 --driver-memory 2g --executor-memory 2g --executor-cores 4 sparkparallelbig.py
```

```
./runYarn-withHistory.sh sparkparallelbig.py
```

The output remains the same as we see above.

non-parallelized pySpark code on temperatures-big.csv

```
[x_omkbh@heffa2 ~]$ python nonparallel.py -t 1950:2014 -i /nfshome/hadoop_examples/shared_data/temperatu
Running Non-Parallelized Python code to find minimum and maximum temperature in each year:
From 1950 To 2014
Input file: /nfshome/hadoop_examples/shared_data/temperatures-big.csv
Output file: nonparallelminmaxbig
Runtime was 15.95247233311 minutes
```

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. In this exercise you will use the temperature-readings.csv file. The output should contain the following information: Year, month, count

```
from pyspark import SparkContext
sc = SparkContext(appName="CounterSpark")
lines = sc.textFile('/user/x_omkbh/data/temperature-readings.csv')
lines = lines.map(lambda a: a.split(";"))
obs = lines.filter(lambda ob:(int(ob[1][:4]) >= 1950 and
                              int(ob[1][:4]) <= 2014)).cache()
# Q2a) Year-month, count
temperatures = obs.map(lambda ob:(ob[1][:7], (float(ob[3]), 1))) \
                              .filter(lambda (month, (temp, count)): temp > 10)
counter = temperatures.reduceByKey(lambda (temp1, count1), (temp2, count2):
                                              (temp1, count1 + count2)) \
                                .map(lambda (month, (temp, count)):(month, count))
counter.repartition(1).saveAsTextFile('/user/x_omkbh/countaboveten')
# Q2b) Year,month, distinct count
station_temperatures = obs.map(lambda ob:(ob[1][:7],(ob[0], float(ob[3])))) \
                         .filter(lambda (month, (station, temp)):temp > target_temp)
year_station = station_temperatures.map(lambda (month, (station, temp)):
                                          (month, (station, 1))).distinct()
counter = year_station.reduceByKey(lambda (station1, count1), (station2, count2):
                                              (station1, count1 + count2)) \
                                .map(lambda (month, (station, count)): (month, count))
counter.repartition(1).saveAsTextFile('/user/x_omkbh/distinctcountaboveten')
```

Number of the temperature readings above 10C for each month:

```
print counter1.take(15)
(u'2010-07', 143419)
(u'1957-06', 18956)
(u'1967-12', 6)
(u'2002-08', 126073)
(u'2000-06', 86592)
(u'1985-09', 27922)
(u'1993-06', 39261)
(u'1958-07', 24709)
(u'1996-05', 21527)
(u'1992-09', 39772)
(u'1995-04', 3037)
(u'1979-11', 33)
(u'1953-11', 120)
(u'2001-09', 79657)
(u'1954-03', 77)
```

Distinct Number of the readings above 10C for each month:

```
print counter2.take(15)
(u'1979-01', 1)
(u'1982-10', 315)
(u'1957-05', 126)
(u'2014-08', 296)
(u'1998-08', 326)
(u'1982-07', 321)
(u'1958-04', 81)
(u'1981-06', 331)
(u'1953-12', 58)
(u'1962-09', 300)
(u'1978-06', 354)
(u'1972-04', 245)
(u'2006-10', 276)
(u'1993-05', 292)
(u'1951-10', 113)
```

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. In this exercise you will use the temperature-readings.csv file. The output should contain the following information: Year, month, station number, average monthly temperature

```
from pyspark import SparkContext
sc = SparkContext(appName="AvgMonthlyTemp")
lines = sc.textFile('/user/x_vinbe/data/temperature-readings.csv')
lines = lines.map(lambda a: a.split(";"))
rows = lines.filter(lambda row: (int(row[1][:4]) >= 1960 and int(row[1][:4]) <= 2014))
```

```
dailyTempsStation = rows.map(lambda row:((row[1], row[0]),(float(row[3]),float(row[3]))))
dailyMinMaxTempsStation = dailyTempsStation.reduceByKey(lambda
                                                          (mintemp1, maxtemp1),
                                                          (mintemp2, maxtemp2):
                                                          (min(mintemp1, mintemp2),
                                                          max(maxtemp1, maxtemp2)))

monthlyAvgTempsStation = dailyMinMaxTempsStation.map(lambda ((day, station),
                                                              (mintemp, maxtemp)):
                                                            ((day[:7], station),
                                                             (sum((mintemp, maxtemp)), 2))) \
                                                       .reduceByKey(lambda (temp1, count1),
                                                                    (temp2, count2):
                                                                    (temp1 + temp2,
                                                                     count1 + count2)) \
                                                       .map(lambda ((month, station),
                                                                    (temp, count)):
                                                                    ((month, station),
                                                                    temp / float(count)))

monthlyAvgTempsStation.repartition(1).saveAsTextFile('stationAvgMonthTemp_vinbe')
```

Output for question 3

```
print monthlyAvgTempsStation.take(15)
((u'2001-06', u'137110'), 15.471666666666664)
((u'2004-12', u'62390'), 2.9903225806451617)
((u'1985-11', u'138270'), -6.583333333333333)
((u'2002-02', u'178970'), -10.655357142857142)
((u'2010-01', u'64030'), -4.441935483870968)
((u'1966-04', u'182720'), -3.3416666666666663)
((u'1967-11', u'108170'), 5.213333333333334)
((u'1961-09', u'92400'), 11.986666666666666)
((u'1983-03', u'160890'), -5.962903225806452)
((u'1963-01', u'79580'), -2.5709677419354837)
((u'1972-06', u'84020'), 16.326666666666668)
((u'1992-07', u'65020'), 18.00161290322581)
((u'1983-07', u'105370'), 18.343548387096774)
((u'1979-05', u'136360'), 8.11774193548387)
((u'1982-06', u'136420'), 9.688333333333333)
```

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. In this exercise you will use the temperature-readings.csv and precipitation-readings.csv files. The output should contain the following information: Station number, maximum measured temperature, maximum daily precipitation HINT: The correct result for this question should be empty.

```
#Station number, maximum measured temperature, maximum daily precipitation
from pyspark import SparkContext
sc = SparkContext(appName="temp_ppt_max")
#Maximum Temperatures
temperaturereadings = sc.textFile('/user/x_omkbh/data/temperature-readings.csv')
temperaturereadings = temperaturereadings.map(lambda temp1: temp1.split(";"))
temperaturereadings = temperaturereadings.map(lambda temp2: (temp2[0], float(temp2[3])))
max_temp = temperaturereadings.reduceByKey(max)
max_temp = max_temp.filter(lambda temp1: temp1[1] > 25 and temp1[1] < 30)
#Maximum Precipitation
precipitationreadings = sc.textFile('/user/x_omkbh/data/precipitation-readings.csv')
precipitationreadings = precipitationreadings.map(lambda ppt1: ppt1.split(";"))
precipitationreadings = precipitationreadings.map(lambda ppt2: (ppt2[0]+','+ppt2[1], float(ppt2[3])))
max_ppt = precipitationreadings.reduceByKey(lambda p1, p2: p1+p2)
max_ppt = max_ppt.filter(lambda ppt1: ppt1[1] >= 100 and ppt1[1] <= 200) \
.map(lambda ppt2: (ppt2[0].split(",")[0], ppt2[1])).reduceByKey(max)
#Join of Max Temperature and Precipitation
max_tempppt = max_temp.leftOuterJoin(max_ppt)
max_temppptcsv = max_tempppt.map(lambda a: '%s,%s,%s' % (a[0], a[1][0], a[1][1]))
max_tempppt.coalesce(1).saveAsTextFile('/user/x_omkbh/tempandppt_max1')
```

```
print max_tempppt.take(15)
#Station number, maximum measured temperature, maximum daily precipitation
(u'128510', (29.5, None))
(u'192830', (29.5, None))
(u'84660', (27.6, None))
(u'139110', (29.0, None))
(u'161670', (25.7, None))
(u'166940', (27.9, None))
(u'77180', (29.3, None))
(u'180740', (29.0, None))
(u'72340', (29.8, None))
(u'147560', (29.9, None))
(u'180750', (29.3, None))
(u'83460', (28.0, None))
(u'83620', (29.4, None))
(u'159680', (26.2, None))
(u'139340', (28.9, None))
```

Question 5:

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file) for the period 1993-2016. 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). In this exercise you will use the precipitation-readings.csv and stations-Ostergotland.csv files. HINT (not for the SparkSQL lab): Avoid using joins here! stations-Ostergotland.csv is small and if distributed will cause a number of unnecessary shuffles when joined with precipitation RDD. If you distribute precipitation-readings.csv then either repartition your stations RDD to 1 partition or make use of the collect to acquire a python list and broadcast function to broadcast the list to all nodes. The output should contain the following information: Year, month, average monthly precipitation

```
from pyspark import SparkContext
sc = SparkContext(appName ="question5")

precipitation = sc.textFile("/user/x_vinbe/data/precipitation-readings.csv")

stationOst = sc.textFile("/user/x_vinbe/data/stations-Ostergotland.csv")
stationOst = stationOst.map(lambda x: x.split(";"))
stationOst = stationOst.map(lambda line: line[0])
stationOst = stationOst.coalesce(1)
stationOst = stationOst.collect()
stationOst = sc.broadcast(stationOst)

precip = precipitation.map(lambda x: x.split(";"))
precip = precip.map(lambda x: ((x[0], x[1][0:4], x[1][5:7]), (float(x[3]))))
precip = precip.filter(lambda x: x[0][0] in stationOst.value)

precip_2016 = precip.filter(lambda x: int(x[0][1]) >= 1993 and int(x[0][1]) <= 2016)
precip_2016 = precip_2016.reduceByKey(lambda x, y: x + y)
precip_2016 = precip_2016.map(lambda line: ((line[0][1], line[0][2]), (line[1], 1)))
precip_2016 = precip_2016.reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1]))
precip_2016 = precip_2016.map(lambda line: (line[0][0], line[0][1], (line[1][0] / line[1][1])))
precip_2016 = precip_2016.coalesce(1)
precip_2016.saveAsTextFile("question5")
```

```
print precip_2016.take(15)
(u'2012', u'09', 72.75)
(u'1995', u'05', 26.000000000000002)
(u'2015', u'04', 15.337499999999997)
(u'2007', u'04', 21.249999999999996)
(u'2007', u'06', 108.95)
(u'2011', u'06', 88.35000000000001)
(u'2011', u'10', 43.75000000000001)
(u'2014', u'10', 72.13749999999999)
(u'1996', u'09', 57.466666666666676)
(u'1995', u'07', 43.6)
(u'2002', u'05', 72.13333333333334)
(u'2010', u'04', 23.783333333333335)
(u'1999', u'01', 61.933333333333394)
(u'2013', u'11', 46.37500000000002)
(u'2010', u'03', 23.88333333333334)
```

Question 6:

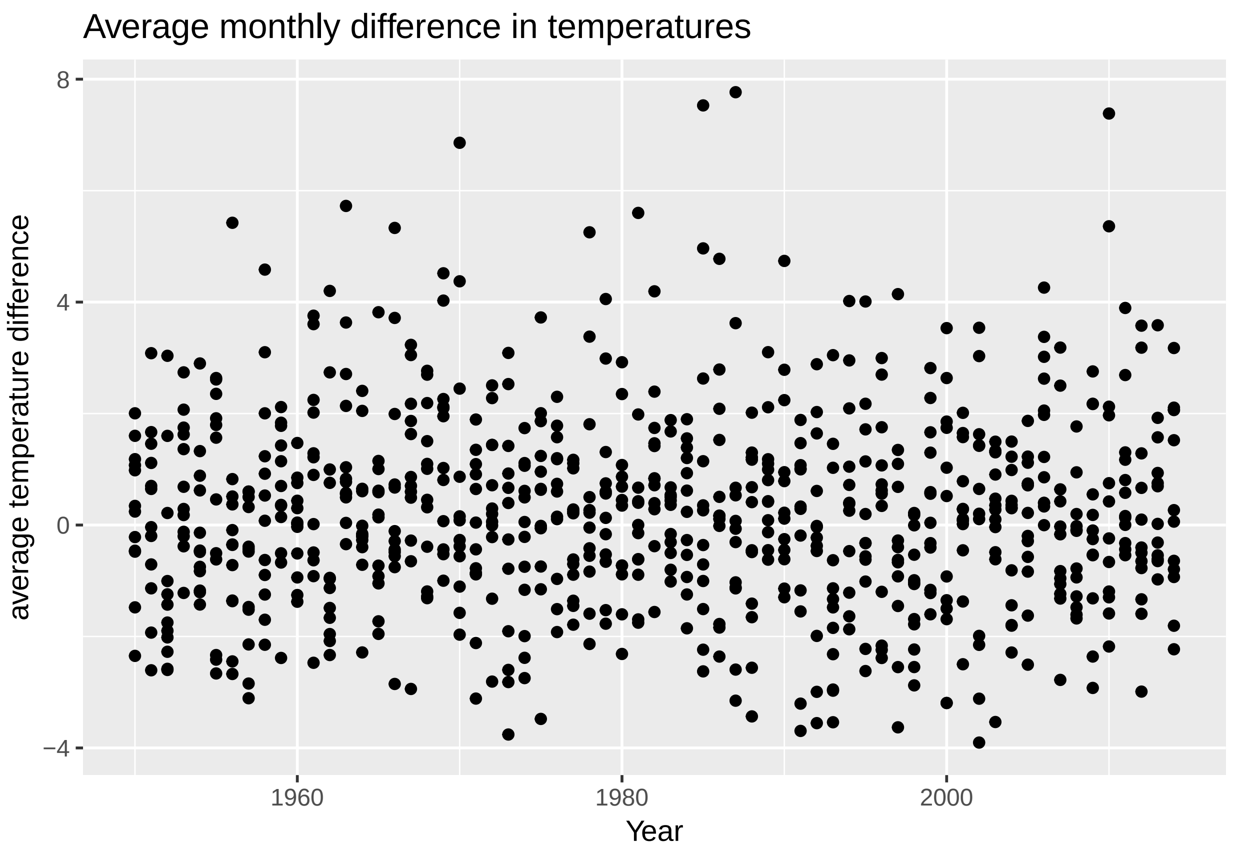
Compare the average monthly temperature (find the difference) in the period 1950-2014 for all stations in Östergotland with long-term monthly averages in the period of 1950-1980. Make a plot of your results. HINT: The first step is to find the monthly averages for each station. In the next step, you can average over all stations to acquire the average temperature for a specific year and month. This RDD/Data Frame can be used to compute the long-term average by averaging over all the years in the interval. The output should contain the following information: Year, month, difference

```
#Year, month, difference
from pyspark import SparkContext
sc = SparkContext(appName="temp_avg_monthlyostergotland")
oststations = sc.textFile('/user/x_omkbh/data/stations-Ostergotland.csv')
oststations = oststations.map(lambda line: line.split(";"))
oststations = oststations.map(lambda x: int(x[0])).distinct().collect()
oststations_flag = (lambda s: s in oststations)

temperatures = sc.textFile('/user/x_omkbh/data/temperature-readings.csv') \
                            .map(lambda line: line.split(";")) \
                            .filter(lambda x: oststations_flag(int(x[0]))
                             and int(x[1][0:4])>= 1950 and int(x[1][0:4])<= 2014)
monthly_avg_temperature = 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)))

monthly_avg_temperature_lt = monthly_avg_temperature.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 monthly_avg_temperature_lt.collect()}

monthly_avg_temperature = monthly_avg_temperature.map(lambda (month, temp):
                          (month, abs(temp) - abs(month_temp[month[-2:]]))) \
                          .sortBy(ascending=True, keyfunc=lambda (month, temp): month)
monthly_avg_temperaturecsv = monthly_avg_temperature.map(lambda a: '%s,%s,%s' %
                             (a[0].split('-')[0], a[0].split('-')[1], a[1]))
monthly_avg_temperaturecsv.repartition(1).saveAsTextFile('/user/x_omkbh/avg_monthly_temp')
```

```
1950,01,2.00483133412
1950,02,-2.34798988599
1950,03,1.1819828212
1950,04,1.60069315899
1950,05,0.982351940463
1950,06,-0.216232256095
1950,07,-1.47714267742
1950,08,0.241517150903
1950,09,0.343179398558
1950,10,-0.460520515247
1950,11,-0.47779366064
1950,12,1.07259158462
1951,01,-0.19629769814
1951,02,-2.60656131457
1951,03,3.08359572443
1951,04,-0.0381401743418
1951,05,-1.93038999502
```

```
1951,06,-1.13606558943
1951,07,-0.707303967738
```

```
part.000006 <- read.csv("C:/Users/Omkar/Downloads/Big Data Analytics/lab1/part-000006", header=FALSE)
part.000006 %>% ggplot(aes(x=V1, y=V3))+geom_point()+
  ggtitle("Average monthly difference in temperatures")+
  labs(x="Year",y="average temperature difference")
```


Average monthly difference in temperatures
average temperature difference
Year