732A54: Lab 1
Big Data Analytics

Carles Sans Fuentes

February 27, 2017

Assignment 1

Exercise 1

Part A code

```
##1 maximum temperatures descending
#max
from pyspark import SparkContext
sc1 = SparkContext(appName = "1 max temperatures descending")
reading = sc1.textFile("/user/x_carsa/data/temperature-readings.csv")
separate = reading.map(lambda a: a.split(";"))
lines = separate.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])))) ##added x[0] as it is the station column
maxTemperatures = temperatures.reduceByKey(lambda a,b: a if a[1]>b[1] else b)
maxTemperaturestogether = maxTemperatures.repartition(1)
maxTemperaturesSorted = maxTemperaturestogether.sortBy(ascending=False,keyfunc=lambda k: k[1][1]) ## the first [1] stands for k = length 1 and vector length 2, with the second [1, we acces to the second element of v]
maxTemperaturesSorted.saveAsTextFile("results/1resultsMax")
```

Output:

```

Row(_c0=u'1975', _c1=u'134520', value=36.1)
Row(_c0=u'1992', _c1=u'140360', value=35.4)
Row(_c0=u'1994', _c1=u'54300', value=34.7)
Row(_c0=u'2010', _c1=u'133250', value=34.4)
Row(_c0=u'2014', _c1=u'133250', value=34.4)
Row(_c0=u'1989', _c1=u'140200', value=33.9)
Row(_c0=u'1982', _c1=u'106100', value=33.8)
Row(_c0=u'1968', _c1=u'134520', value=33.7)
Row(_c0=u'1966', _c1=u'102190', value=33.5)
Row(_c0=u'1983', _c1=u'133260', value=33.3)
Row(_c0=u'2002', _c1=u'156770', value=33.3)
Row(_c0=u'1986', _c1=u'106100', value=33.2)
Row(_c0=u'1970', _c1=u'102190', value=33.2)
Row(_c0=u'2000', _c1=u'133470', value=33.0)
Row(_c0=u'1956', _c1=u'161670', value=33.0)
Row(_c0=u'1959', _c1=u'108640', value=32.8)
Row(_c0=u'1991', _c1=u'133260', value=32.7)
Row(_c0=u'2006', _c1=u'139120', value=32.7)
Row(_c0=u'1988', _c1=u'134460', value=32.6)
Row(_c0=u'2011', _c1=u'116430', value=32.5)
Row(_c0=u'1999', _c1=u'133470', value=32.4)
Row(_c0=u'2003', _c1=u'139120', value=32.2)
Row(_c0=u'2007', _c1=u'133250', value=32.2)
Row(_c0=u'2008', _c1=u'151280', value=32.2)
Row(_c0=u'1953', _c1=u'137030', value=32.2)
Row(_c0=u'1955', _c1=u'139340', value=32.2)
Row(_c0=u'1973', _c1=u'139200', value=32.2)
Row(_c0=u'2005', _c1=u'133470', value=32.1)
Row(_c0=u'1969', _c1=u'102190', value=32.0)
Row(_c0=u'1979', _c1=u'133470', value=32.0)
Row(_c0=u'2001', _c1=u'106160', value=31.9)
Row(_c0=u'1997', _c1=u'146070', value=31.8)
Row(_c0=u'1977', _c1=u'133470', value=31.8)
```

```
Row(_c0=u'2013', _c1=u'139120', value=31.6)
Row(_c0=u'2009', _c1=u'140460', value=31.5)
Row(_c0=u'2012', _c1=u'151240', value=31.3)
Row(_c0=u'1964', _c1=u'139200', value=31.2)
Row(_c0=u'1971', _c1=u'139200', value=31.2)
Row(_c0=u'1972', _c1=u'112080', value=31.2)
Row(_c0=u'1976', _c1=u'139200', value=31.1)
Row(_c0=u'1961', _c1=u'137030', value=31.0)
Row(_c0=u'1963', _c1=u'106270', value=31.0)
Row(_c0=u'1995', _c1=u'136420', value=30.8)
Row(_c0=u'1996', _c1=u'134410', value=30.8)
Row(_c0=u'1958', _c1=u'139340', value=30.8)
Row(_c0=u'1978', _c1=u'139200', value=30.8)
Row(_c0=u'1974', _c1=u'134520', value=30.6)
Row(_c0=u'1954', _c1=u'137030', value=30.5)
Row(_c0=u'1980', _c1=u'106100', value=30.4)
Row(_c0=u'1952', _c1=u'137030', value=30.4)
Row(_c0=u'1990', _c1=u'133260', value=30.2)
Row(_c0=u'2004', _c1=u'139120', value=30.2)
Row(_c0=u'1985', _c1=u'102200', value=29.8)
Row(_c0=u'1957', _c1=u'135440', value=29.8)
Row(_c0=u'1981', _c1=u'140480', value=29.7)
Row(_c0=u'1993', _c1=u'102210', value=29.7)
Row(_c0=u'1987', _c1=u'102200', value=29.6)
Row(_c0=u'1984', _c1=u'139200', value=29.5)
Row(_c0=u'1967', _c1=u'134520', value=29.5)
Row(_c0=u'1950', _c1=u'134110', value=29.4)
Row(_c0=u'1960', _c1=u'107400', value=29.4)
Row(_c0=u'1998', _c1=u'133260', value=29.2)
Row(_c0=u'1951', _c1=u'107440', value=28.5)
Row(_c0=u'1965', _c1=u'192710', value=28.5)
Row(_c0=u'1962', _c1=u'156730', value=27.4)
```

Part b code

```

#min
from pyspark import SparkContext
sc1 = SparkContext(appName = "1 min temperatures descending")
reading = sc1.textFile("/user/x_carsa/data/temperature-readings.csv")
separate = reading.map(lambda a: a.split(";"))
lines = separate.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])))) ##added x[0] as it is the
    station column
maxTemperatures = temperatures.reduceByKey(lambda a,b: a if a[1]<b[1] else b)
maxTemperaturestogether = maxTemperatures.repartition(1)
maxTemperaturesSorted = maxTemperaturestogether.sortBy(ascending=True,keyfunc=lambda k: k
    [1][1]) ## the first [1] stands for k = length 1 and vector length 2, with the second [1,
    we acces to the second element of v]
maxTemperaturesSorted.saveAsTextFile("results/1resultsMin")
```

Output:

```

Row(_c0=u'1990', _c1=u'133260', value=-35.0)
Row(_c0=u'1952', _c1=u'144300', value=-35.5)
Row(_c0=u'1974', _c1=u'102190', value=-35.6)
Row(_c0=u'1954', _c1=u'107440', value=-36.0)
Row(_c0=u'1992', _c1=u'116430', value=-36.1)
Row(_c0=u'1975', _c1=u'136010', value=-37.0)
Row(_c0=u'1972', _c1=u'188800', value=-37.5)
Row(_c0=u'2000', _c1=u'146350', value=-37.6)
Row(_c0=u'1995', _c1=u'107400', value=-37.6)
Row(_c0=u'1957', _c1=u'108640', value=-37.8)
Row(_c0=u'1983', _c1=u'135520', value=-38.2)
Row(_c0=u'1989', _c1=u'116430', value=-38.2)
Row(_c0=u'1953', _c1=u'134110', value=-38.4)
Row(_c0=u'2009', _c1=u'116430', value=-38.5)
Row(_c0=u'1993', _c1=u'107400', value=-39.0)
Row(_c0=u'1984', _c1=u'140480', value=-39.2)
Row(_c0=u'1973', _c1=u'112080', value=-39.3)
Row(_c0=u'2008', _c1=u'116430', value=-39.3)
Row(_c0=u'1991', _c1=u'107400', value=-39.3)
Row(_c0=u'2005', _c1=u'116430', value=-39.4)
Row(_c0=u'1961', _c1=u'102190', value=-39.5)
Row(_c0=u'1964', _c1=u'188830', value=-39.5)
Row(_c0=u'1970', _c1=u'149160', value=-39.6)
Row(_c0=u'2004', _c1=u'135460', value=-39.7)
Row(_c0=u'1988', _c1=u'158750', value=-39.9)
Row(_c0=u'1960', _c1=u'151290', value=-40.0)
Row(_c0=u'1997', _c1=u'188790', value=-40.2)
Row(_c0=u'1994', _c1=u'116430', value=-40.5)
```

```
Row(_c0=u'2006', _c1=u'102190', value=-40.6)
Row(_c0=u'2007', _c1=u'158740', value=-40.7)
Row(_c0=u'2013', _c1=u'116430', value=-40.7)
Row(_c0=u'1963', _c1=u'133630', value=-41.0)
Row(_c0=u'1955', _c1=u'139340', value=-41.2)
Row(_c0=u'2003', _c1=u'158750', value=-41.5)
Row(_c0=u'1969', _c1=u'105230', value=-41.5)
Row(_c0=u'1996', _c1=u'146070', value=-41.7)
Row(_c0=u'2010', _c1=u'146050', value=-41.7)
Row(_c0=u'2011', _c1=u'146050', value=-42.0)
Row(_c0=u'1962', _c1=u'136010', value=-42.0)
Row(_c0=u'1950', _c1=u'180750', value=-42.0)
Row(_c0=u'1951', _c1=u'188830', value=-42.0)
Row(_c0=u'1968', _c1=u'133470', value=-42.0)
Row(_c0=u'1982', _c1=u'135520', value=-42.2)
Row(_c0=u'2002', _c1=u'102190', value=-42.2)
Row(_c0=u'1976', _c1=u'136010', value=-42.2)
Row(_c0=u'2014', _c1=u'158740', value=-42.5)
Row(_c0=u'1977', _c1=u'105230', value=-42.5)
Row(_c0=u'1998', _c1=u'133260', value=-42.7)
Row(_c0=u'2012', _c1=u'116490', value=-42.7)
Row(_c0=u'1958', _c1=u'102190', value=-43.0)
Row(_c0=u'1985', _c1=u'180940', value=-43.4)
Row(_c0=u'1959', _c1=u'134110', value=-43.6)
Row(_c0=u'1981', _c1=u'158750', value=-44.0)
Row(_c0=u'2001', _c1=u'116430', value=-44.0)
Row(_c0=u'1965', _c1=u'151290', value=-44.0)
Row(_c0=u'1979', _c1=u'173960', value=-44.0)
Row(_c0=u'1986', _c1=u'140480', value=-44.2)
Row(_c0=u'1971', _c1=u'112080', value=-44.3)
Row(_c0=u'1980', _c1=u'140480', value=-45.0)
Row(_c0=u'1956', _c1=u'144300', value=-45.0)
Row(_c0=u'1967', _c1=u'173960', value=-45.4)
Row(_c0=u'1987', _c1=u'158750', value=-47.3)
Row(_c0=u'1978', _c1=u'133470', value=-47.7)
Row(_c0=u'1999', _c1=u'158750', value=-49.0)
Row(_c0=u'1966', _c1=u'108640', value=-49.4)
```

Part c code

```
from functools import reduce
from itertools import groupby


file = open("/nfshome/x_carsa/Desktop/data/temperatures-big.csv", mode = "r")

working_data = []
for line in file:
    temp = line.split(";")
    if int(temp[1][0:4]) >= 1950 and int(temp[1][0:4]) <= 2014:
        working_data.append((temp[1][0:4], (temp[0],float(temp[3]))))
    else:
        continue



def reduceByKey(func, iterable):
    first = lambda p: p[0]
    second = lambda p: p[1]
    return map(lambda l: (l[0], reduce(func, map(second, l[1]))), groupby(sorted(iterable, key=
        first), first))


maxData = reduceByKey(lambda x ,y: x if x[1] > y[1] else y, working_data)

for line in maxData:
    print line
```

Output: It last for about 40 minutes whereas in the previous case it last much less (about a minute or two). Such differences are related to the non-usage of parallel execution of the algorithm rather than for a single computer with its nodes.

Exercise 2

Count the number of readings for each month in the period of 1950-2014 which are higher than 10 degrees.

```
from pyspark import SparkContext
sc = SparkContext(appName = "2 number of readings each month")
reading = sc.textFile("/user/x_carsa/data/temperature-readings.csv")
separate = reading.map(lambda a: a.split(";"))
lines = separate.filter(lambda x: int(x[1][0:4]) >= 1950 and int(x[1][0:4])<=2014 and float(x
    [3])>10)
higher10temperatures = lines.map(lambda x: (x[1][0:7], 1)) ##added as it is the station column
counts = higher10temperatures.reduceByKey(lambda v1,v2: v1 + v2) ##summing counts of every one
maxTemperaturestogether = counts.repartition(1) ## putting everything in one partition

maxTemperaturestogether.sortBy(ascending = False, keyfunc= lambda x : x[1]) \
            .saveAsTextFile("results/2resultshigher10")
print maxTemperaturestogether.take(20)
```

Output:

```
(u'2014-07', 147681)
(u'2011-07', 146656)
(u'2010-07', 143419)
(u'2012-07', 137477)
(u'2013-07', 133657)
(u'2009-07', 133008)
(u'2011-08', 132734)
(u'2009-08', 128349)
(u'2013-08', 128235)
(u'2003-07', 128133)
(u'2002-07', 127956)
(u'2006-08', 127622)
(u'2008-07', 126973)
(u'2002-08', 126073)
(u'2005-07', 125294)
(u'2011-06', 125193)
(u'2012-08', 125037)
(u'2006-07', 124794)
(u'2010-08', 124417)
(u'2014-08', 124045)
(u'1997-07', 123496)
(u'2007-07', 123218)
(u'2013-06', 122181)
(u'1997-08', 121154)
(u'2001-07', 120529)
(u'1998-07', 120230)
(u'2000-07', 119769)
(u'2004-07', 119536)
(u'1999-07', 116385)
(u'2008-08', 114272)
(u'2004-08', 114168)
(u'2002-06', 114034)
(u'2005-08', 113950)
(u'2001-08', 113937)
(u'2007-08', 110428)
(u'2000-08', 109201)
(u'2003-08', 108501)
(u'1996-08', 107758)
(u'1997-06', 104696)
(u'1999-06', 103227)
(u'2007-06', 103046)
(u'2008-06', 102900)
(u'2010-06', 102716)
(u'2006-06', 102588)
(u'2014-06', 101711)
(u'1998-08', 101387)
(u'1996-07', 99916)
(u'2003-06', 99693)
(u'2011-09', 99335)
(u'1999-08', 97437)
(u'2006-09', 97181)
(u'2012-06', 94513)
(u'2001-06', 93375)
(u'2005-06', 90724)
(u'2004-06', 89628)
(u'1999-09', 89418)
(u'2009-09', 89106)
(u'2009-06', 87787)
(u'2000-06', 86592)
(u'2014-09', 86090)
(u'1998-06', 82608)
(u'2013-05', 81996)
(u'2013-09', 81960)
(u'1996-06', 80440)
(u'2001-09', 79657)
(u'1998-09', 76535)
(u'1988-07', 75521)
(u'2005-09', 75494)
(u'2010-09', 74816)
```

```
(u'1997-09', 74472)
(u'1991-07', 73385)
(u'2004-09', 73334)
(u'1973-07', 71522)
(u'1991-08', 71185)
(u'2003-09', 70459)
(u'2012-09', 70427)
(u'1990-07', 70031)
(u'1988-08', 69913)
(u'1987-07', 68135)
(u'1989-07', 67880)
(u'1989-08', 67793)
(u'1990-08', 67604)
(u'1995-08', 66920)
(u'1974-07', 66277)
(u'2002-05', 66116)
(u'2002-09', 65928)
(u'1974-08', 64470)
(u'1975-07', 64408)
(u'1976-07', 64109)
(u'2000-09', 63837)
(u'1988-06', 63572)
(u'1992-07', 62911)
(u'1975-08', 62565)
(u'2007-09', 61346)
(u'1978-07', 60998)
(u'2008-09', 60989)
(u'1976-08', 60898)
(u'2009-05', 60867)
(u'1989-06', 60822)
(u'1979-07', 60719)
(u'1994-07', 60691)
```

Part b: 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.

```
from pyspark import SparkContext
sc = SparkContext(appName = "2 number of readings each month one per station max")
reading = sc.textFile("/user/x_carsa/data/temperature-readings.csv")
separate = reading.map(lambda a: a.split(";"))
lines = separate.filter(lambda x: int(x[1][0:4]) >= 1950 and int(x[1][0:4])<=2014 and float(x
    [3])>10)
higher10temperatures = lines.map(lambda x: (x[1][0:7],(x[0],1))).distinct() ##DISTINCT = one
    count for each station

counts = higher10temperatures.reduceByKey(lambda v1,v2: (v1[0],v1[1] + v2[1])) ##summing counts
     of every one
counts= counts.map(lambda x: (x[0], x[1][1]))
maxTemperaturestogether = counts.repartition(1) ## putting everything in one partition
 ## putting everything in one partition

maxTemperaturestogether.sortBy(ascending= False, keyfunc= lambda x: x[1])\
        .saveAsTextFile("results/2resultshigher10one_eachmonth")
print maxTemperaturestogether.take(20)
```

Output:

```
(u'1972-10', 378)
(u'1973-06', 377)
(u'1973-05', 377)
(u'1973-09', 376)
(u'1972-08', 376)
(u'1972-05', 375)
(u'1971-08', 375)
(u'1972-06', 375)
(u'1972-09', 375)
(u'1971-09', 374)
(u'1972-07', 374)
(u'1971-06', 374)
(u'1973-08', 373)
(u'1971-05', 373)
(u'1974-06', 372)
(u'1974-08', 372)
(u'1974-05', 370)
(u'1970-08', 370)
(u'1971-07', 370)
(u'1973-07', 370)
(u'1974-09', 370)
(u'1975-09', 369)
(u'1970-09', 369)
```

```
(u'1976-05', 369)
(u'1970-06', 369)
(u'1976-06', 368)
(u'1975-06', 368)
(u'1975-08', 367)
(u'1975-05', 367)
(u'1970-05', 366)
(u'1976-09', 365)
(u'1977-06', 364)
(u'1967-05', 363)
(u'1976-08', 363)
(u'1974-07', 362)
(u'1970-07', 362)
(u'1967-09', 361)
(u'1966-09', 360)
(u'1966-06', 360)
(u'1966-08', 359)
(u'1969-09', 359)
(u'1967-06', 359)
(u'1965-09', 358)
(u'1978-09', 358)
(u'1967-08', 358)
(u'1975-07', 358)
(u'1969-08', 357)
(u'1968-06', 357)
(u'1968-08', 357)
(u'1976-07', 356)
(u'1968-09', 356)
(u'1968-05', 355)
(u'1965-06', 355)
(u'1979-05', 354)
(u'1978-06', 354)
(u'1965-08', 354)
(u'1966-05', 354)
(u'1977-08', 354)
(u'1968-07', 353)
(u'1977-09', 353)
(u'1978-05', 352)
(u'1969-06', 352)
(u'1966-07', 352)
(u'1967-07', 351)
(u'1979-06', 351)
(u'1977-05', 351)
(u'1979-09', 351)
(u'1977-07', 350)
(u'1978-08', 350)
(u'1965-07', 349)
(u'1973-10', 349)
(u'1969-07', 349)
(u'1971-10', 347)
(u'1969-10', 346)
(u'1979-07', 345)
(u'1996-06', 345)
(u'1970-10', 345)
(u'1974-04', 344)
(u'1965-05', 344)
(u'1978-07', 343)
(u'1996-07', 342)
(u'1996-05', 342)
(u'1996-08', 341)
(u'1978-10', 340)
(u'1996-09', 340)
(u'1975-10', 340)
(u'1979-08', 340)
(u'1997-09', 340)
(u'1982-06', 339)
(u'1997-06', 338)
(u'1980-09', 338)
(u'1980-05', 337)
(u'1981-05', 337)
(u'1997-08', 337)
(u'1983-06', 337)
(u'1983-05', 336)
(u'1965-10', 335)
(u'1981-09', 335)
(u'1969-05', 335)
(u'1981-08', 334)
(u'1982-09', 334)
(u'1997-07', 333)
(u'1984-05', 333)
(u'1983-09', 332)
(u'1980-06', 332)
(u'1981-06', 331)
(u'1999-06', 330)
(u'1983-08', 330)
(u'1982-05', 330)
(u'1980-08', 330)
```

```
(u'1999-07', 329)
(u'1981-07', 329)
(u'1999-09', 328)
(u'1985-09', 327)
(u'1984-09', 327)
(u'1999-08', 327)
(u'1998-09', 326)
(u'1998-08', 326)
(u'2002-06', 326)
(u'1998-07', 326)
(u'1982-08', 326)
(u'1998-06', 326)
(u'1981-10', 325)
(u'1999-05', 325)
(u'2000-08', 325)
(u'1985-05', 325)
(u'1980-07', 324)
(u'1967-10', 324)
(u'1984-06', 324)
(u'2001-07', 324)
(u'2002-07', 324)
(u'2001-06', 324)
(u'1985-06', 324)
(u'2002-05', 324)
(u'1987-06', 323)
(u'2003-06', 323)
(u'2000-05', 323)
(u'2002-09', 323)
(u'2001-08', 323)
(u'1986-09', 323)
(u'1987-09', 323)
(u'2002-08', 322)
(u'2001-09', 322)
(u'1968-04', 322)
(u'1998-05', 322)
(u'2000-09', 322)
(u'1988-06', 322)
(u'2003-05', 321)
(u'2004-05', 321)
(u'2003-07', 321)
(u'1984-10', 321)
(u'1982-07', 321)
(u'2000-06', 321)
(u'1991-06', 321)
(u'2004-09', 321)
(u'1987-05', 320)
(u'2010-06', 320)
(u'2000-07', 320)
(u'1988-05', 320)
(u'2003-09', 320)
(u'2004-08', 320)
(u'1987-08', 320)
(u'2003-08', 320)
(u'1997-05', 319)
(u'1987-07', 319)
(u'2004-06', 319)
(u'2004-07', 319)
(u'2010-05', 319)
(u'2011-07', 319)
(u'1983-07', 319)
(u'2010-07', 318)
```

Exercise 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.

```
from pyspark import SparkContext

sc = SparkContext(appName = "3 average temperature for each month")
reading = sc.textFile("/user/x_carsa/data/temperature-readings.csv")
separate = reading.map(lambda a: a.split(";"))
lines = separate.filter(lambda x: int(x[1][0:4]) >= 1960 and int(x[1][0:4])<=2014)
temperatures = lines.map(lambda x: ((x[1], x[0]), (float(x[3]), float(x[3])))) ##added x[0] as it is the station column, and one to divide it at then end by the averages
minmax = temperatures.reduceByKey(lambda v1,v2: (v1[0] if v1[0]>v2[0] else v2[0], v1[1] if v1[1]<v2[1] else v2[1])) # first is max, second is min, done just to get the max and min per day in case there is more than one
```

```
calculations= minmax.map(lambda x: ((x[0][0][0:7], x[0][1]), (x[1][0], x[1][1], 1))) # getting together by index and adding 1 to be able to divide by counts
averageperday = calculations.reduceByKey(lambda v1,v2: (v1[0] + v2[0],v1[1] + v2[1], v1[2]+ v2[2])) #
averageperday = averageperday.map(lambda x: ((x[0]), ((x[1][0]+x[1][1])/(2*x[1][2]))))
temperaturestogether = averageperday.repartition(1)

print temperaturestogether.take(20)

temperaturestogether.sortBy(ascending = False, keyfunc= lambda x : x[1]) \
            .saveAsTextFile("results/3average_temperature_eachmonth")
```

Output:

```
((u'2014-07', u'96000'), 26.3)
((u'1994-07', u'96550'), 23.071052631578944)
((u'1983-08', u'54550'), 23.0)
((u'1994-07', u'78140'), 22.970967741935482)
((u'1994-07', u'85280'), 22.872580645161293)
((u'1994-07', u'75120'), 22.85806451612903)
((u'1994-07', u'65450'), 22.856451612903225)
((u'1994-07', u'96000'), 22.80806451612903)
((u'1994-07', u'95160'), 22.76451612903226)
((u'1994-07', u'86200'), 22.711290322580645)
((u'2002-08', u'78140'), 22.700000000000003)
((u'1994-07', u'76000'), 22.698387096774198)
((u'1997-08', u'78140'), 22.666129032258066)
((u'1994-07', u'105260'), 22.65967741935484)
((u'1975-08', u'54550'), 22.642857142857142)
((u'2006-07', u'76530'), 22.598387096774193)
((u'1994-07', u'86330'), 22.54838709677419)
((u'2006-07', u'75120'), 22.52741935483871)
((u'1994-07', u'54300'), 22.469354838709677)
((u'2006-07', u'78140'), 22.45806451612903)
((u'2001-07', u'96550'), 22.408333333333335)
((u'2010-07', u'98180'), 22.37903225806452)
((u'2006-07', u'65450'), 22.37741935483871)
((u'1994-07', u'85210'), 22.375806451612902)
((u'1994-07', u'98180'), 22.367741935483874)
((u'2014-07', u'98180'), 22.367741935483874)
((u'2002-08', u'98180'), 22.366129032258062)
((u'1994-07', u'92100'), 22.31774193548387)
((u'1994-07', u'86470'), 22.30806451612903)
((u'1994-07', u'83230'), 22.272580645161288)
((u'1994-07', u'64290'), 22.259677419354837)
((u'1994-07', u'97490'), 22.258064516129032)
((u'1994-07', u'94180'), 22.25322580645161)
((u'1972-07', u'173960'), 22.244999999999997)
((u'1994-07', u'74080'), 22.241935483870968)
((u'2006-07', u'54300'), 22.23709677419355)
((u'2002-08', u'98210'), 22.23548387096774)
((u'1994-07', u'106070'), 22.232258064516135)
((u'1994-07', u'75100'), 22.229032258064517)
((u'1994-07', u'53440'), 22.197499999999998)
((u'1994-07', u'83270'), 22.177419354838705)
((u'1994-07', u'103080'), 22.16451612903226)
((u'1994-07', u'82110'), 22.161290322580644)
((u'1994-07', u'97120'), 22.135483870967743)
((u'2010-07', u'98210'), 22.111290322580647)
((u'1994-07', u'53430'), 22.096774193548388)
((u'1997-08', u'86330'), 22.07903225806452)
((u'2006-07', u'66500'), 22.05483870967742)
((u'1994-07', u'76530'), 22.033870967741933)
((u'1997-08', u'98210'), 21.983870967741936)
((u'2014-07', u'98210'), 21.962903225806453)
((u'1994-07', u'62400'), 21.951612903225808)
((u'1997-08', u'62400'), 21.938709677419357)
((u'1994-07', u'108110'), 21.90806451612903)
((u'1994-07', u'83130'), 21.900000000000002)
((u'1997-08', u'98180'), 21.887096774193544)
((u'2006-07', u'98210'), 21.872580645161293)
((u'1997-08', u'98290'), 21.864516129032257)
((u'1991-08', u'78040'), 21.85)
((u'2010-07', u'78140'), 21.830645161290324)
((u'1994-07', u'63340'), 21.7758064516129)
((u'1994-07', u'91130'), 21.76290322580645)
((u'1994-07', u'105370'), 21.761290322580642)
((u'2008-07', u'83420'), 21.75)
((u'1994-07', u'64130'), 21.72741935483871)
((u'1997-08', u'96550'), 21.725806451612904)
((u'1994-07', u'83440'), 21.716129032258067)
((u'2006-07', u'85210'), 21.706451612903226)
((u'1994-07', u'74420'), 21.690322580645162)
((u'2003-07', u'98180'), 21.68548387096774)
```

```
71 ((u'1997-08', u'52230'), 21.68548387096774)
```

Exercise 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 file.

```
1  from pyspark import SparkContext
2
3  sc = SparkContext(appName = "4 average temperature for each month")
4  filetemp = sc.textFile("/user/x_carsa/data/temperature-readings.csv")
5
6  temp = filetemp.map(lambda a: a.split(";"))
7
8
9  calculatedTemp = temp.map(lambda x: ((x[0]), (x[3])))\
10                              .filter(lambda (station_number, temp): (temp >= 25 and temp <=30))\
11                              .reduceByKey(max)\
12                              .repartition(1)\
13                              .sortByKey()
14
15
16 fileprec = sc.textFile("/user/x_carsa/data/precipitation-readings.csv")
17 prec = fileprec.map(lambda a: a.split(";"))
18
19 calculatedPrec = prec.map(lambda x: ((x[0],x[1]), float(x[3])))\
20                              .reduceByKey(lambda x1, x2: (x1+x2))\
21                              .filter(lambda x: (x >= 100 and x <=200))\
22                              .reduceByKey(max)\
23                              .map(lambda ((station_number, date), prec): (station_number, prec))
                                    \
24                              .repartition(1)\
25                              .sortByKey()
26 print calculatedPrec.take(20)
27
28 together = calculatedPrec.join(calculatedTemp)\
29           .repartition(1)\
30           .sortBy(ascending= False, keyfunc= lambda x: (x[0]))
31
32 print together.take(20)
33 together.saveAsTextFile("results/4stations_with_max_temperatures_precipitations")
```

Output:

```
1 No output
```

Exercise 5

Calculate the average monthly precipitation for the Östergotland region (list of stations is provided in the separate file). In order to do this, you will first need to calculate the total daily precipitation before calculating the monthly average. In this exercise you will use the precipitation-readings.csv and stations-Ostergotland.csv files.

```
1  from pyspark import SparkContext
2  sc = SparkContext(appName = "5 average monthly precipitation for the Ostergotland")
3
4  reading2 = sc.textFile("/user/x_carsa/data/stations-Ostergotland.csv")
5  separate2 = reading2.map(lambda a: a.split(";"))
6  stations = separate2.map(lambda observation: int(observation[0]))
7  stations = stations.distinct().collect() #collect transforms the rdd to a python list object
8  stations = {station: True for station in stations}
9
10
11 reading1 = sc.textFile("/user/x_carsa/data/precipitation-readings.csv")#x[0] = station number
12
13
14 percip_daily = reading1.map(lambda line: line.split(";")) \
```

```
                        .filter(lambda obs: stations.get(int(obs[0]), False)) \
                        .map(lambda obs: (obs[1], float(obs[3]))) \
                        .reduceByKey(lambda value1, value2: (value1 + value2))#total per day

counting = percip_daily.map(lambda x: (x[0][0:7],(float(x[1]), 1)))
average = counting.reduceByKey(lambda v1, v2: (v1[0]+v2[0], v1[1] + v2[1])) #total
    precipitation and number of days
average = average.map(lambda x: ((x[0]),(x[1][0]/x[1][1])))
print average.take(20)
results = average.repartition(1) \
          .sortBy(ascending= False, keyfunc= lambda x: x[0])\
        .saveAsTextFile("results/5_average_monthly_precipitation")
```

Output:

```
(u'2016-07', 0.0)
(u'2016-06', 12.710000000000004)
(u'2016-05', 7.548387096774194)
(u'2016-04', 7.173333333333335)
(u'2016-03', 5.151612903225806)
(u'2016-02', 5.948275862068965)
(u'2016-01', 5.761290322580644)
(u'2015-12', 7.4645161290322575)
(u'2015-11', 17.036666666666665)
(u'2015-10', 0.5838709677419355)
(u'2015-09', 27.013333333333335)
(u'2015-08', 6.9645161290322575)
(u'2015-07', 30.73548387096774)
(u'2015-06', 20.97666666666667)
(u'2015-05', 24.058064516129033)
(u'2015-04', 4.09)
(u'2015-03', 10.99677419354839)
(u'2015-02', 7.092857142857143)
(u'2015-01', 15.254838709677419)
(u'2014-12', 9.151612903225804)
(u'2014-11', 13.979999999999999)
(u'2014-10', 18.616129032258065)
(u'2014-09', 12.920000000000003)
(u'2014-08', 23.43548387096774)
(u'2014-07', 5.932258064516129)
(u'2014-06', 20.036666666666672)
(u'2014-05', 14.96774193548387)
(u'2014-04', 8.469999999999999)
(u'2014-03', 9.435483870967742)
(u'2014-02', 12.489285714285716)
(u'2014-01', 16.148387096774194)
(u'2013-12', 10.90645161290323)
(u'2013-11', 12.366666666666667)
(u'2013-10', 13.903225806451612)
(u'2013-09', 6.983333333333336)
(u'2013-08', 13.954838709677418)
(u'2013-07', 14.080645161290322)
(u'2013-06', 16.353333333333335)
(u'2013-05', 12.36774193548387)
(u'2013-04', 10.209999999999999)
(u'2013-03', 1.9064516129032258)
(u'2013-02', 7.292857142857143)
(u'2013-01', 5.554838709677419)
(u'2012-12', 12.954838709677418)
(u'2012-11', 13.73)
(u'2012-10', 12.693548387096774)
(u'2012-09', 14.55)
(u'2012-08', 13.319354838709677)
(u'2012-07', 11.432258064516128)
(u'2012-06', 26.44)
(u'2012-05', 4.445161290322581)
(u'2012-04', 12.556666666666668)
(u'2012-03', 1.6548387096774195)
(u'2012-02', 5.931034482758621)
(u'2012-01', 8.429032258064517)
(u'2011-12', 8.154838709677417)
(u'2011-11', 2.6933333333333334)
(u'2011-10', 8.46774193548387)
(u'2011-09', 10.513333333333334)
(u'2011-08', 16.69677419354839)
(u'2011-07', 18.370967741935484)
(u'2011-06', 17.669999999999998)
(u'2011-05', 7.325806451612904)
(u'2011-04', 2.983333333333333)
(u'2011-03', 3.838709677419355)
(u'2011-02', 5.253571428571428)
(u'2011-01', 6.800000000000001)
(u'2010-12', 7.196774193548386)
(u'2010-11', 18.71)
(u'2010-10', 10.167741935483871)
```

```
(u'2010-09', 8.616666666666667)
(u'2010-08', 20.91290322580645)
(u'2010-07', 17.883870967741938)
(u'2010-06', 9.729999999999999)
(u'2010-05', 12.999999999999998)
(u'2010-04', 4.756666666666666)
(u'2010-03', 4.622580645161291)
(u'2010-02', 11.30357142857143)
(u'2010-01', 6.964516129032259)
(u'2009-12', 10.345161290322583)
(u'2009-11', 12.843333333333334)
(u'2009-10', 11.0)
(u'2009-09', 5.989999999999999)
(u'2009-08', 11.916129032258064)
(u'2009-07', 21.903225806451616)
(u'2009-06', 9.953333333333335)
(u'2009-05', 10.483870967741936)
(u'2009-04', 0.56)
(u'2009-03', 6.674193548387096)
(u'2009-02', 5.310714285714288)
(u'2009-01', 3.074193548387097)
(u'2008-12', 8.416129032258064)
(u'2008-11', 9.350000000000001)
(u'2008-10', 11.52903225806452)
(u'2008-09', 9.473333333333333)
(u'2008-08', 26.809677419354838)
(u'2008-07', 16.490322580645163)
(u'2008-06', 8.586666666666668)
(u'2008-05', 4.47741935483871)
(u'2008-04', 4.05)
(u'2008-03', 8.167741935483871)
(u'2008-02', 5.844827586206897)
(u'2008-01', 8.703225806451613)
(u'2007-12', 10.587096774193547)
(u'2007-11', 10.136666666666667)
(u'2007-10', 5.441935483870968)
(u'2007-09', 12.373333333333335)
(u'2007-08', 10.483870967741936)
(u'2007-07', 18.5741935483871)
(u'2007-06', 21.790000000000003)
(u'2007-05', 7.8419354838709685)
(u'2007-04', 4.250000000000001)
(u'2007-03', 7.8419354838709685)
(u'2007-02', 7.085714285714286)
(u'2007-01', 13.283870967741937)
(u'2006-12', 5.75483870967742)
(u'2006-11', 14.343333333333334)
(u'2006-10', 22.870967741935484)
(u'2006-09', 3.8533333333333335)
(u'2006-08', 28.661290322580644)
(u'2006-07', 5.6096774193548375)
(u'2006-06', 6.2266666666666675)
(u'2006-05', 10.129032258064516)
(u'2006-04', 8.873333333333333)
(u'2006-03', 5.393548387096776)
(u'2006-02', 7.44642857142857)
(u'2006-01', 4.08076923076923)
(u'2005-12', 10.961290322580645)
(u'2005-11', 6.52)
(u'2005-10', 7.3645161290322605)
(u'2005-09', 2.79)
```

Exercise 6

Compare the average monthly temperature (find the difference) in the period 1950-2014 for each station in Östergotland with long-term monthly averages in the period of 1950-1980. Make a plot of your results.

```
from pyspark import SparkContext
sc = SparkContext(appName = "6 Comparison Ostergotland average monthly temperature 1950-2014
     with long-term monthly averages in the period of 1950-1980 ")

reading2 = sc.textFile("/user/x_carsa/data/stations-Ostergotland.csv")

separate2 = reading2.map(lambda a: a.split(";"))
stations = separate2.map(lambda x: int(x[0]))
stations = stations.distinct().collect() # Taken different stations as in previous step
stations = {station: True for station in stations}

#Extracting temperatures
```

```
12
13
14 reading2 = sc.textFile("/user/x_carsa/data/temperature-readings.csv")
15 myfilter = reading2.map(lambda a: a.split(";")) \
16                 .filter(lambda x: (int(x[1][0:4]) >= 1950 and int(x[1][0:4])<=2014 and stations
                        .get(int(x[0], False))))
17
18 average_short_by_station = myfilter.map(lambda x: ((x[1], int(x[0])),(float(x[3]), float(x[3]))
       )) \
19                         .reduceByKey(lambda (min1,max1), (min2, max2): (min(min1, min2), max(
                               max1, max2))) \
20                         .map(lambda ((date, station),(tmin, tmax)): ((date[0:7], station), (
                               tmin+tmax, 2)))\
21                         .reduceByKey(lambda (temp1, count1), (temp2, count2): (temp1 + temp2,
                               count1 + count2))\
22                         .map(lambda ((date, station), (ttemp, tcount)): (date, station, ttemp/
                               float(tcount)))
23
24 average_short = average_short_by_station.map(lambda (date, station, avgtemp): (date, (avgtemp,
       1))) \
25                         .reduceByKey(lambda (temp1, count1), (temp2, count2): (temp1 + temp2,
                               count1 + count2)) \
26                         .map(lambda (date,(ttemp, tcount)): (date, ttemp/float(tcount)))
27
28 avglong = average_short.filter(lambda (date, avgtemp): (int(date[0:4])>= 1950 and int(date
       [0:4])<= 1980)) \
29                         .map(lambda (date, avgtemp): (date[5:7], (avgtemp,1))) \
30                         .reduceByKey(lambda (temp1, count1), (temp2, count2): (temp1 + temp2,
                               count1 + count2)) \
31                         .map(lambda (date,(ttemp, tcount)): (date, ttemp/float(tcount)))
32
33
34 mylong = avglong.collect()
35 avgMonth = {month: temp for (month,temp) in mylong} # first part (month: temp, = structure type
       , for = old structure type)
36
37 together = average_short.map(lambda x: (x[0], abs(x[1]) - abs(avgMonth.get(x[0][5:7], 0))))
38
39 together.sortBy(ascending= False, keyfunc= lambda x: x[0], numPartitions = 1)\
40             .saveAsTextFile("results/6_average_diff")
```

Output:

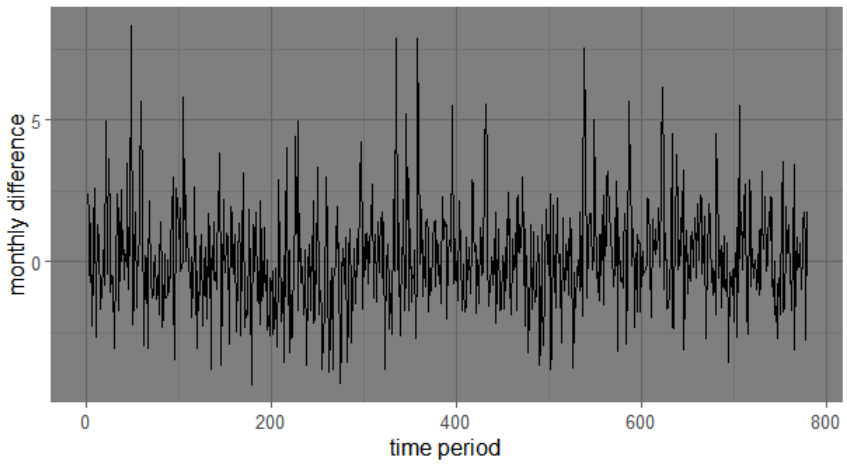

Figure 1: Average monthly difference on temperature