# Big Data Analytics
# 732A54
# Lab 3

Karolina Ziomek, Rasmus Holm

December 4, 2016

# Exercise 1

## Output

### Question

**Minimums** (u '1966 ',-49.4) (u '1999 ',-49.0) (u '1978 ',-47.7) (u '1987 ',-47.3) (u '1967 ',-45.4) (u '2014 ',-42.5) (u '1977 ',-42.5) (u '2002 ',-42.2) (u '1976 ',-42.2) (u '1982 ',-42.2)

**Maximums** (u '2014 ',34.4) (u '2010 ',34.4) (u '1989 ',33.9) (u '1982 ',33.8) (u '1968 ',33.7) (u '1991 ',32.7) (u '2006 ',32.7) (u '1988 ',32.6) (u '2011 ',32.5) (u '1961 ',31.0)

### a)

**Minimums** (u '1966 ',(u '179950 ',-49.4)) (u '1999 ',(u '192830 ',-49.0)) (u '1978 ',(u '155940 ',-47.7)) (u '1987 ',(u '123480 ',-47.3)) (u '1967 ',(u '166870 ',-45.4)) (u '1979 ',(u '112170 ',-44.0)) (u '1965 ',(u '189780 ',-44.0)) (u '1981 ',(u '166870 ',-44.0)) (u '2001 ',(u '112530 ',-44.0)) (u '1972 ',(u '167860 ',-37.5))

**Maximums** (u '1986 ',(u '76470 ',33.2)) (u '1970 ',(u '103080 ',33.2)) (u '1956 ',(u '145340 ',33.0)) (u '2000 ',(u '62400 ',33.0)) (u '1959 ',(u '65160 ',32.8)) (u '1966 ',(u '151640 ',33.5)) (u '2002 ',(u '78290 ',33.3)) (u '1983 ',(u '98210 ',33.3)) (u '1964 ',(u '76430 ',31.2)) (u '1971 ',(u '65130 ',31.2)) (u '1972 ',(u '98200 ',31.2))

### b)

Running time for sequential execution on the temperature-readings.csv

| Real | User | Sys |
|---|---|---|
| 2m43.850s | 2m41.552s | 0m2.340s |

Running time for sequential execution on the temperatures-big.csv

| Real | User | Sys |
|---|---|---|
| 15m11.000s | 14m31.339s | 0m21.987s |

Running time for parallel execution on the temperature-readings.csv

| Real |
|---|
| 3mins, 48sec |

Running time for parallel execution on the temperatures-big.csv

| Real |
|---|
| 9mins, 7sec |

What we can see from the execution times is that the sequential program is faster with lower amount of data while you gain quite a bit by running in parallel with a larger data set. This is intuitively sound since there are a lot of overhead in parallel execution and if the calculations cannot dominate the execution time it is not worth it.

## Code

### Initialization

```
def exercise01():
data = sc.textFile("../data/temperature-readings.csv")

observations = data.map(lambda line: line.split(";")) \
.filter(lambda observation:
(int(observation[1][:4]) >= 1950 and
    int(observation[1][:4]) <= 2014)) \
.cache()

exercise01question(observations)
exercise01a(observations)
```

### Question

```
def exercise01question(observations):
temperatures = observations.map(lambda observation:
(observation[1][:4], float(observation[3])))

min_temperatures = temperatures.reduceByKey(min) \
.sortBy(ascending=True,
keyfunc=lambda (year, temp): temp)

max_temperatures = temperatures.reduceByKey(max) \
.sortBy(ascending=False,
keyfunc=lambda (year, temp): temp)

min_temperatures.repartition(1).saveAsTextFile("../nsc_result/1_qa")
max_temperatures.repartition(1).saveAsTextFile("../nsc_result/1_qb")
```

### a)

```
def exercise01a(observations):
station_temperatures = observations.map(lambda observation:
(observation[1][:4],
```

```
            (observation[0], float(observation[3])))

min_temperatures_station = station_temperatures.reduceByKey(
lambda (station1, temp1), (station2, temp2):
(station1, temp1)
if temp1 < temp2 else
(station2, temp2)) \
.sortBy(ascending=True,
keyfunc=lambda (year, (station, temp)): temp)

max_temperatures_station = station_temperatures.reduceByKey(
lambda (station1, temp1), (station2, temp2):
(station1, temp1)
if temp1 > temp2 else
(station2, temp2)) \
.sortBy(ascending=False,
keyfunc=lambda (year, (station, temp)): temp)


min_temperatures_station.repartition(1).saveAsTextFile("../nsc_result/1_aa")
max_temperatures_station.repartition(1).saveAsTextFile("../nsc_result/1_ab")
```

b)

```
from collections import defaultdict

def exercise01_seq():
with open('../data/temperature-readings-small.csv', 'rb') as infile:
year_temp = defaultdict(list)
for line in infile:
values = line.split(";")
year = int(values[1][:4])
temp = float(values[3])

year_temp[year].append(temp)

for year in year_temp:
print(year, min(year_temp[year]), max(year_temp[year]))
```

# Exercise 2

## Output

### a)

(u '2000 -09 ',63837) (u '1985 -06 ',44839) (u '2012 -11 ',255) (u '1986 -07 ',55741) (u '1958 -08 ',25613) (u '1975 -01 ',22) (u '1989 -11 ',1126) (u '1972 -08 ',53918) (u '1993 -09 ',19915) (u '1957 -09 ',12572)

### b)

(u '1982 -08 ',326) (u '1965 -07 ',349) (u '1988 -06 ',322) (u '2006 -08 ',309) (u '1986 -04 ',260) (u '1989 -01 ',23) (u '2007 -03 ',201) (u '1955 -04 ',81) (u '2008 -11 ',106) (u '1994 -03 ',89)

## Code

### Initialization

```
def exercise02():
data = sc.textFile("../data/temperature-readings.csv")

observations = data.map(lambda line: line.split(";")) \
.filter(lambda observation:
         (int(observation[1][:4]) >= 1950 and
             int(observation[1][:4]) <= 2014)) \
.cache()

exercise02a(observations)
exercise02b(observations)
```

### a)

```
def exercise02a(observations):
temperatures = observations.map(lambda observation:
(observation[1][:7], (float(observation[3]), 1))) \
.filter(lambda (month, (temp, count)): temp > 10)
reading_counts = temperatures.reduceByKey(lambda (temp1, count1), (temp2, count2):
(temp1, count1 + count2)) \
.map(lambda (month, (temp, count)):
(month, count))

reading_counts.repartition(1).saveAsTextFile("../nsc_result/2_a")
```

**b)**

```
def exercise02b(observations):
station_temperatures = observations.map(lambda observation:
(observation[1][:7],
(observation[0], float(observation[3])))) \
.filter(lambda (month, (station, temp)): temp > 10)

year_station = station_temperatures.map(
lambda (month, (station, temp)): (month, (station, 1))).distinct()
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("../nsc_result/2_b")
```

# Exercise 3

## Output

((u '1978 -08 ',u '137100 ') ,14.516129032258062) ((u '1984 -12 ',u '81060 ') ,3.285483870967741) ((u '1976 -03 ',u '162880 ') ,-7.824193548387097) ((u '1981 -10 ',u '123070 ') ,1.1741935483870969) ((u '1967 -03 ',u '92100 ') ,2.9016129032258067) ((u '1994 -12 ',u '72120 ') ,2.4387096774193546) ((u '1983 -07 ',u '105450 ') ,18.220967741935482) ((u '1963 -05 ',u '81630 ') ,11.143548387096773) ((u '1985 -08 ',u '106580 ') ,14.7741935483871) ((u '1969 -08 ',u '74240 ') ,18.375806451612902)

## Code

```
def exercise03():
data = sc.textFile("../data/temperature-readings.csv")

observations = data.map(lambda line: line.split(";"))
observations = observations.filter(lambda observation:
                                    (int(observation[1][:4]) >= 1960 and
                                     int(observation[1][:4]) <= 2014))

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

station_day_minmax_temps = station_day_temperatures.reduceByKey(
lambda
(mintemp1, maxtemp1),
(mintemp2, maxtemp2):
(min(mintemp1, mintemp2),
max(maxtemp1, maxtemp2)))

station_month_avg_temps = station_day_minmax_temps.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)))

station_month_avg_temps.repartition(1).saveAsTextFile("../nsc_result/3")
```

# Exercise 4

## Output

(97510 ,(30.0 ,103.99999999999999)) (75250 ,(30.0 ,101.8)) (52350 ,(30.0 ,101.6))
(71420 ,(30.0 ,106.3))

We filtered before finding the maximums else the result would be empty.

## Code

```
def exercise04():
temperature_data = sc.textFile("../data/temperature-readings.csv").cache()
precipitation_data = sc.textFile("../data/precipitation-readings.csv").cache()

temp_obs = temperature_data.map(lambda line: line.split(";")) \
.map(lambda obs: (int(obs[0]), float(obs[3]))) \
.filter(lambda (station, temp):
         temp >= 25 and temp <= 30 ) \
.reduceByKey(max)

precip_obs = precipitation_data.map(lambda line: line.split(";")) \
.map(lambda obs: ((obs[1], int(obs[0])), float(obs[3]))) \
.reduceByKey(lambda precip1, precip2: precip1 + precip2) \
.map(lambda ((day, station), precip):
      (station, precip)) \
.filter(lambda (station, precip):
         precip >= 100 and precip <= 200) \
.reduceByKey(max)

combined = temp_obs.join(precip_obs)

combined.repartition(1).saveAsTextFile("../nsc_result/4")
```

## Exercise 5

### Output

(u '2003 -12 ',10.087096774193547) (u '1997 -04 ',5.190000000000001) (u '1996 -12 ',7.65483870967742) (u '2014 -09 ',12.920000000000003) (u '1997 -01 ',1.1193548387096772) (u '2014 -04 ',8.469999999999999) (u '2011 -01 ',6.800000000000001) (u '2001 -12 ',6.809677419354839) (u '1999 -04 ',10.909999999999998) (u '2010 -05 ',12.999999999999998)

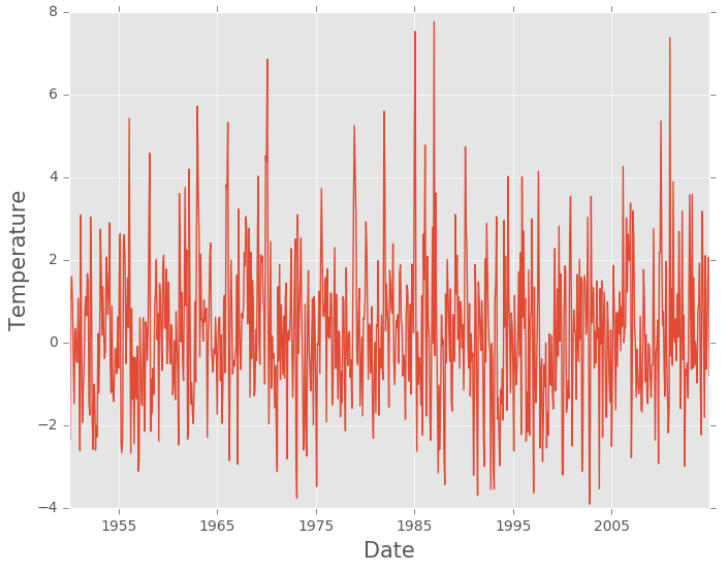
### Code

```
def exercise05():
station_data = sc.textFile("../data/stations-Ostergotland.csv")

stations = station_data.map(lambda line: line.split(";")) \
.map(lambda obs: int(obs[0])) \
.distinct().collect()
stations = {station: True for station in stations}

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

precipitation_daily = precipitation_data.map(lambda line: line.split(";")) \
.filter(lambda obs: stations.get(int(obs[0]), False)) \
.map(lambda obs: (obs[1], float(obs[3]))) \
.reduceByKey(lambda precip1, precip2:
             precip1 + precip2)

precipitation_avg_month =  precipitation_daily.map(lambda (day, precip):
                                                   (day[:7], (precip, 1))) \
.reduceByKey(lambda (precip1, count1),
             (precip2, count2):
             (precip1 + precip2,
                      count1 + count2)) \
.map(lambda (month, (precip, count)):
      (month, precip / float(count)))

precipitation_avg_month.repartition(1).saveAsTextFile("../nsc_result/5")
```

# Exercise 6

## Output

(u '1950 -01 ',-2.004831334118534) (u '1950 -02 ',2.3479898859946133) (u '1950 -03 ',2.4922107271847125) (u '1950 -04 ',1.6006931589915459) (u '1950 -05 ',0.9823519404632854) (u '1950 -06 ',-0.21623225609516084) (u '1950 -07 ',-1.4771426774150633) (u '1950 -08 ',0.24151715090320636) (u '1950 -09 ',0.3431793985584335) (u '1950 -10 ',-0.46052051524713367)




We can see from the plot that there is no pattern to be recognized and we can conclude that global warming is a hoax.

## Code

### Data

```
def exercise06():
station_data = sc.textFile("../data/stations-Ostergotland.csv")

stations = station_data.map(lambda line: line.split(";")) \
.map(lambda obs: int(obs[0])) \
.distinct().collect()
stations = {station: True for station in stations}

temperature_data = sc.textFile("../data/temperature-readings.csv")

temperature_data_filtered = temperature_data.map(
```

```
lambda line: line.split(";")) \
.filter(lambda obs:
         (stations.get(int(obs[0]), False) and
                     int(obs[1][:4]) >= 1950 and
                     int(obs[1][:4]) <= 2014))

month_avg_temp = temperature_data_filtered.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)))

month_longterm_avg_temp = month_avg_temp.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 month_longterm_avg_temp.collect()}

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

month_avg_temp.repartition(1).saveAsTextFile("../nsc_result/6")
```

### Plot

```
import matplotlib.pyplot as plt

from datetime import datetime

def exercise06_plot():
    plt.style.use('ggplot')
    chars_remove = set(["(", ")", " ", "u", "'"])

    avg_year_month = []
```

```
    avg_temp = []

    with open("../nsc_result/6/part-00000", "r") as file:
        for line in file:
            elements = ''.join([char for char in line
                                if char not in chars_remove]).split(",")
            year_month = datetime.strptime(elements[0].strip(), "%Y-%m")
            temp = float(elements[1])

            avg_year_month.append(year_month)
            avg_temp.append(temp)

    plt.plot(avg_year_month, avg_temp)
    plt.xlabel("Date", size=15)
    plt.ylabel("Temperature", size=15)
    plt.savefig('images/fig.png')
    return 'images/fig.png'
return exercise06_plot()
```