
Please document your code, without sufficient documentation you won't receive any points.

As a preparation for the following tasks create a personal directory in Hadoop:

```
1 $ ssh abu2
2 $ hadoop fs -mkdir hdfs://abu2/user/<yourLogin>
3 # Also try to use the NFS gateway for Hadoop on Abu2:
4 $ ls /hdfs/user/<yourLogin>
```

1 Exploration of the Chicago-Crime Dataset (R) (120 P)

We will explore the Chicago-crime dataset. You will notice that interactive analysis of the whole dataset is barely possible in R. We provide a small sample in `/home/bigdata/4/chicago_crime_sample.csv`. Create an R data frame from the CSV. Choose a suitable data type in R for each column. To document the results, we will create a "lab notebook" using R Markdown.

Also, we will use the `ggplot2` library to create plots.

1.1 Guiding questions

1. When do crimes occur?
2. Which time of day, which time of year?
3. Which type of crimes occur at which time?

1.2 Dataset: Chicago Crime

The dataset contains information about crimes reported to the police in Chicago. This covers data about the location, date, time and type of the crime in various formats. We won't describe too many details about the content to give you the chance to inspect the data yourself.

1.3 Hints

The `str()` function is helpful to inspect the content of data frames.

1.3.1 Markdown

Markdown ¹ is a simple language and format to enrich text with semantical information such as chapter information or embed code into it.

R Markdown is directly supported by RStudio or it can be manually embedded in files.

An example file is given here:

```
1 ---
2 title: "My Analysis for X"
3 documentclass: article
4 geometry: margin=1cm
```

¹<https://de.wikipedia.org/wiki/Markdown>

```

5 output:
6   pdf_document: default
7   html_document:
8     df_print: paged
9 classoption: a4paper
10 ---
11
12 # Chapter 1
13 ## Example
14
15 This embeds some **R code** to be executed:
16 ''{r}
17 print(summary(seq(1,10)))
18 ''
19
20 ## Figures, set the width appropriately
21 ''{r, fig.width=6, fig.height=4.5}
22 plot(seq(1,10), seq(2,11), main="Simple figure")
23 ''
24
25 ## Inline code:
26 We compute 'r 2*(5*4+1)'
27
28 ''{r}
29 txt = "hello world my world consists not of my big data"
30 print(table(unlist(strsplit(txt, '\\s+'))))
31 ''

```

This result of the compilation can be seen in Figure 1.

To compile the file either run with R (or RScript):

```

1 library(rmarkdown)
2 render("rmarkdown.rmd")

```

or use the knit button in RStudio.

See http://rmarkdown.rstudio.com/articles_integration.html for further description.

1.3.2 GGLOT2

GGPLOT2 is a powerful plotting package for R. *It is an implementation of "Grammar of Graphics", a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers*². It takes a data frame as input.

Please check our seminar http://wr.informatik.uni-hamburg.de/teaching/sommersemester_2016/programmierung_in_r for information about GGPLOT2.

Please also see <http://docs.ggplot2.org/current/> for further information and a sample gallery for many graph types.

Submission:

1-chicago-explore.pdf Your R lab notebook together with the code to explore the data.

2 MapReduce: Processing of Wikipedia Data (Python) (120 P)

By using MapReduce, we will parallelize the word count problem and apply it to the Wikipedia data. The output should be a list of Article id, list of tuples (words, frequency).

Instead of just outputting the words, we will normalize the words by applying alternatively 1) stemming and 2) lemmatisation (see hints).

²<https://en.wikipedia.org/wiki/Ggplot2>

Chapter 1

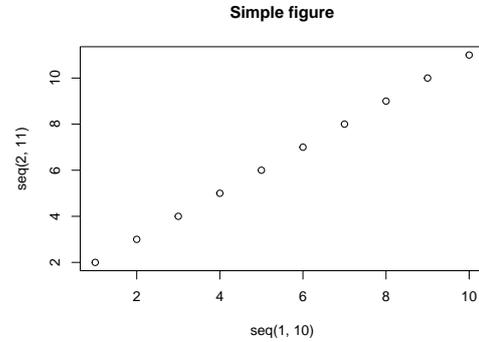
Example

This embeds some R code to be executed:

```
print(summary(seq(1,10)))
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.00   3.25   5.50   5.50   7.75   10.00
```

Figures, set the width appropriately

```
plot(seq(1,10), seq(2,11), main="Simple figure")
```



Inline code:

```
We compute 42
```

1

Figure 1: Compiled version of the rmarkdown.rmd

Specifically your output should be formatted as follows³:

```
1 "articleID", "title", wordcount, "[stemmed word1:4, stemmed word2:3,...]", "[lemma of word1:3, lemma of
  ↳ word2: ...]"
```

The last pieces are lists of word and count tuples. Wordcount is the total number of words in the article. Also, for each word, calculate the sum of all occurrences across all articles and save them as a CSV in such a format:

```
1 Wort1,4
2 Wort2,3
3 ...
```

Use `/home/bigdata/wiki-clean.csv` as input⁴. The output will be called `wiki-clean-frequency.csv`.

You may use your previous Python code and modify it for use with a MapReduce Job.

Measure your programs runtime, both with the execution using a pipe (as described below) and using Hadoop.

³You may define the escaping or choose to format the word list as JSON instead.

⁴Feel free to use the input file supplied in `/home/bigdata`.

2.1 Hints

2.1.1 Stemming and Lemmatisation

*Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form*⁵. For example, the word students is based on student and the root of studying is study. Stemming can be used to normalize words and aggregate all derived words instead of counting them separately.

We can use NLTK⁶ in Python for word stemming. There are several implementations of stemmers provided, we will use the SnowballStemmer.

```
1 from nltk.stem.snowball import SnowballStemmer
2 stemmer = SnowballStemmer("english")
3 print(stemmer.stem("studying"))
4 # prints "studi"; well the world is not perfect
```

Lemmatisation is similar but reduces the word to the word's lemma that is its dictionary form. Check the NLTK documentation for further information how to apply Lemmatisation.

2.1.2 Hadoop streaming

Hadoop streaming is available via `hadoop-streaming.jar`, it allows for running arbitrary programs as map and reduce functions. External programs are called with keys and values as input, the standard output of the program is used as output of the reducer or mapper.⁷

Embedding Python is very simple using this functionality:⁸:

```
1 yarn jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-streaming.jar \
2   -Dmapred.reduce.tasks=1 -Dmapred.map.tasks=11 -mapper $PWD/my-map.py -reducer $PWD/my-reduce.py \
3   -input <input> -output <output-directory>
```

A trivial Python example works as follows:

```
1 #!/usr/bin/python3
2 import sys
3
4 # This program outputs for every input tuple (line for mapper, key/value pair for reduce)
5 # the fixed tuple ("key", "value"). It is possible to filter and aggregate tuples (in reduce).
6 for line in sys.stdin:
7     print("key\tvalue\n")
```

For simpler debugging you can launch your program from the shell:

```
1 cat Input.csv | ./map.py | sort | ./reduce.py
```

2.1.3 Accessing Hadoop Web GUIs

We provide convenience scripts to access the Hadoop web interface in `/home/bigdata/4`. They require you to have a Unix system with a Python 3 installation.

- Append the content of the hosts file to your local hosts file in `/etc/hosts`

```
# cat hosts >> /etc/hosts
```

- Adapt the Python script with your username and execute it (on your local machine) to forward the relevant ports from the cluster.

```
$ python3 hadoop-portforwarding.py
```

⁵<https://en.wikipedia.org/wiki/Stemming>

⁶<http://www.nltk.org/>

⁷<http://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html>

⁸<http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>

- Connect to the different local ports mentioned in the Python script and have a look about the services behind. For Ambari the user/passwort is bigdata.

All that this script is doing is tunneling a list of ports to the servers. If you are not on a Unix system you will have to use a manual way of forwarding these ports.

Submission:

- 2-map.py Your mapper written in Python for processing the CSV file.
- 2-reduce.py Your reduce function for aggregating word counts.

3 MapReduce: Grouping Data (150 P)

For this task, we use the data-energy-efficiency.csv dataset. The dataset has already imported into HDFS, you can also access it at: /home/bigdata/4/data-energy-efficiency.csv. The dataset contains time series for different experiments. We want to average the values of columns. This time we use MapReduce to parallelize this reduction.

3.1 Dataset: data-energy-efficiency.csv

The dataset contains measurements of CPU hardware counters (number of operations that are performed by the CPU, e.g., floating point operations). Up to four different application kernels are run and about 260 CPU counters are measured⁹. To measure multiple counters, the experiments have been repeated and a time series has been created for each experiment.

The CSV-files are structured as follows:

```
1 App0,App1,App2,App3,Time,AGU_BYPASS_CANCEL_COUNTCPU0,(<COUNTER>)+
```

An excerpt of the file with the first eight columns :

```
1 ESM_example,ESM_example,ESM_example,ESM_example,2.750738,55.40407,705.32129,1041.9067
2 ESM_example,ESM_example,ESM_example,ESM_example,2.800752,44.90113,711.82311,1036.9053
3 ESM_example,ESM_example,ESM_example,ESM_example,2.850765,44.2288,719.9632,1030.6752
4 ESM_example,ESM_example,ESM_example,ESM_example,2.900779,45.72922,728.46558,1024.17338
5 ESM_example,ESM_example,ESM_example,ESM_example,2.950792,46.40987,731.63948,1013.98286
6 ESM_example,ESM_example,ESM_example,ESM_example,3.000805,46.91,733.64,1002.98
7 ESM_example,ESM_example,ESM_example,ESM_example,3.050819,78577.32608,26034.71632,54999.62226
8 ESM_example,ESM_example,ESM_example,ESM_example,3.100832,174338.21744,56886.73576,120846.23793
9 ESM_example,ESM_example,ESM_example,ESM_example,3.150846,121595.80666,57856.18875,96096.16269
10 ...
11 EXX_example,EXX_example,EXX_example,EXX_example,5.652014,221434.54323,211603.92431,369932.99742
12 EXX_example,EXX_example,EXX_example,EXX_example,5.702031,107861.4415,179096.3755,234573.991
13 EXX_example,EXX_example,EXX_example,EXX_example,5.752049,31701.88224,128184.82916,122750.35068
14 EXX_example,EXX_example,EXX_example,EXX_example,5.802067,13279.25248,48874.78782,47248.67986
15 EXX_example,EXX_example,EXX_example,EXX_example,5.852085,1819.22816,661.59928,1395.22376
16 EXX_example,EXX_example,EXX_example,EXX_example,5.902103,1070.95888,289.96554,1154.63718
```

When calculating the mean for each of the combination of application's you get the following results (excerpt):

```
1 ESM_example,ESM_example,ESM_example,ESM_example 112.69512266727315,186388.93997432772,118469.79527709562
2 EXX_example,EXX_example,EXX_example,EXX_example 118.44219725094219,251865.2199417397,361377.21140772087
3 VCSEXample,VCSEXample,VCSEXample,VCSEXample 13.882197396378269,409238.7635984908,286333.5696854528
4 cpu,cpu,cpu,cpu 30.69008190862511,6085.659376959866,6830.574735098203
```

Submission:

- 3-mapper.py Your mapper.
- 3-reducer.py Your reducer.

⁹Since only a few counters can be measured concurrently, the experiment has been repeated.


```
15     ...
16
17 wordcloud = WordCloud(relative_scaling=.5)
18 wordcloud.generate_from_frequencies(list(categories.items()))
19
20 image = wordcloud.to_image()
21 print(image.size)
22 image.save(searchword + ".png")
```

Submission:

4-wiki-wordcloud.py Your script for creating word clouds