Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, H. Hatzel

Exercise Sheet 4
Big Data Analytics
WiSe 2016/2017
Due Date: 2016-11-18, 435 P

Please document your code, without sufficent documentation you won't recive any points.

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

```
hadoop fs -mkdir hdfs://abu1.cluster/user/<yourLogin>

# Alternatively, use the NFS gateway on Abu1:
mkdir /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 policy 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:

```
---
title: "My Analysis for X"
output: pdf_document
documentclass: article
```

[1] `https://de.wikipedia.org/wiki/Markdown`

```
classoption: a4paper
geometry: margin=1cm
---

# Chapter 1
## Example

This embeds some **R code** to be executed:
```{r}
print(summary(seq(1,10)))
```

Figures, set the width appropriately
```{r, fig.width=6, fig.height=4.5}
plot(seq(1,10), seq(2,11), main="Simple figure")
```

Inline code:
We compute `r 2*(5*4+1)`
```

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

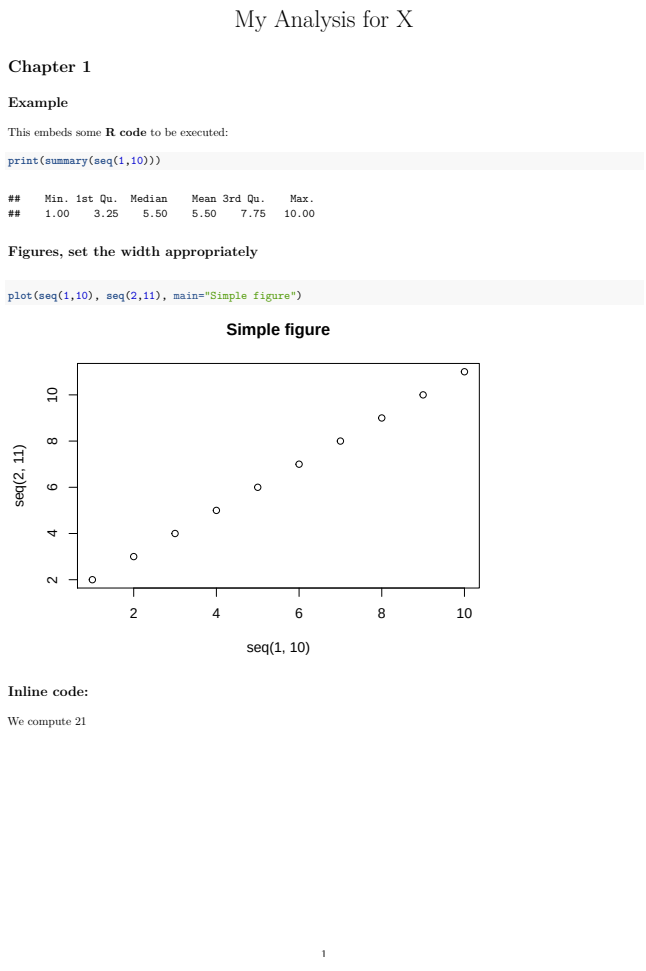

Figure 1: Compiled version of the rmarkdown.rmd

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

```
library(rmarkdown)
render("rmarkdown.rmd")
```

1.3.2 GGPLOT2

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 stemming (see hints).
Specifically your output should be formatted as follows[3]:

```
"articleID","title",wordcount,"[word1:4,word2:3,...]"
```

Where `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:

```
Wort1,4
Wort2,3
...
```

Use *wiki-clean.csv* as input[4]. 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.

2.1 Hints

2.1.1 Stemming

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 NTLK[6] in Python for word stemming. There are several implementations of stemmers provided, we will use the SnowballStemmer.

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

[2] `https://en.wikipedia.org/wiki/Ggplot2`
[3] You may define the escaping or choose to format the word list as JSON instead.
[4] Feel free to use the input file supplied in `/home/bigdata`.
[5] `https://en.wikipedia.org/wiki/Stemming`
[6] `http://www.nltk.org/`

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]:

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

A trivial Python example works as follows:

```
#!/usr/bin/python3
import sys

# This program outputs for every input tuple (line for mapper, key/value pair for reduce)
# the fixed tuple ("key", "value"). It is possible to filter and aggregate tuples (in reduce).
for line in sys.stdin:
   print("key\tvalue\n")
```

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

```
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. 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`

- Execute the Python script (on your local machine) to forward the relevant ports from the cluster.

  `$ python3 hadoop-portfowarding.py`

All that this script is doing is tunneling a list of ports to the abu1 host. 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-efficency.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.

[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/

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 measured[9]. 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:

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

An excerpt of the file with the first eight columns :

```
ESM_example,ESM_example,ESM_example,ESM_example,2.750738,55.40407,705.32129,1041.9067
ESM_example,ESM_example,ESM_example,ESM_example,2.800752,44.90113,711.82311,1036.9053
ESM_example,ESM_example,ESM_example,ESM_example,2.850765,44.2288,719.9632,1030.6752
ESM_example,ESM_example,ESM_example,ESM_example,2.900779,45.72922,728.46558,1024.17338
ESM_example,ESM_example,ESM_example,ESM_example,2.950792,46.40987,731.63948,1013.98286
ESM_example,ESM_example,ESM_example,ESM_example,3.000805,46.91,733.64,1002.98
ESM_example,ESM_example,ESM_example,ESM_example,3.050819,78577.32608,26034.71632,54999.62226
ESM_example,ESM_example,ESM_example,ESM_example,3.100832,174338.21744,56886.73576,120846.23793
ESM_example,ESM_example,ESM_example,ESM_example,3.150846,121595.80666,57856.18875,96096.16269
...
EXX_example,EXX_example,EXX_example,EXX_example,5.652014,221434.54323,211603.92431,369932.99742
EXX_example,EXX_example,EXX_example,EXX_example,5.702031,107861.4415,179096.3755,234573.991
EXX_example,EXX_example,EXX_example,EXX_example,5.752049,31701.88224,128184.82916,122750.35068
EXX_example,EXX_example,EXX_example,EXX_example,5.802067,13279.25248,48874.78782,47248.67986
EXX_example,EXX_example,EXX_example,EXX_example,5.852085,1819.22816,661.59928,1395.22376
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):

```
ESM_example,ESM_example,ESM_example,ESM_example 112.69512266727315,186388.93997432772,118469.79527709562
EXX_example,EXX_example,EXX_example,EXX_example 118.44219725094219,251865.2199417397,361377.21140772087
VCSexample,VCSexample,VCSexample,VCSexample     13.882197396378269,409238.7635984908,286333.5696854528
cpu,cpu,cpu,cpu                                 30.69008190862511,6085.659376959866,6830.574735098203
```

Submission:

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

4 Visualizing Wikipedia Categories as a Word Cloud (Python) (45 P)

For each word, we would like to find associated Wikipedia categories. We want to visualize these as a Word Cloud[10]. Find relevant articles for a given search term and visualize the names of the primary categories of those articles. Categories that occur frequently should be larger in the visualization that is generated. Relevant articles should be those where the word occurs more frequent then a certain threshold value (as supplied via the command line).
You can find the combined results in `/home/bigdata/4/enwiki-clean-all.csv`.

An example: Search term is "Apple". "Apple" appears in many articles of the category "Apple Inc", it does however also appear in the category "Isaac Newton". An example word cloud for the term Apple is shown in Figure 2

4.1 Hints

We will use the package `wordcloud` to generate the graphic.

[9] Since only a few counters can be measured concurrently, the experiment has been repeated.
[10] Word-clouds also known as Tag-clouds: `https://en.wikipedia.org/wiki/Tag_cloud`

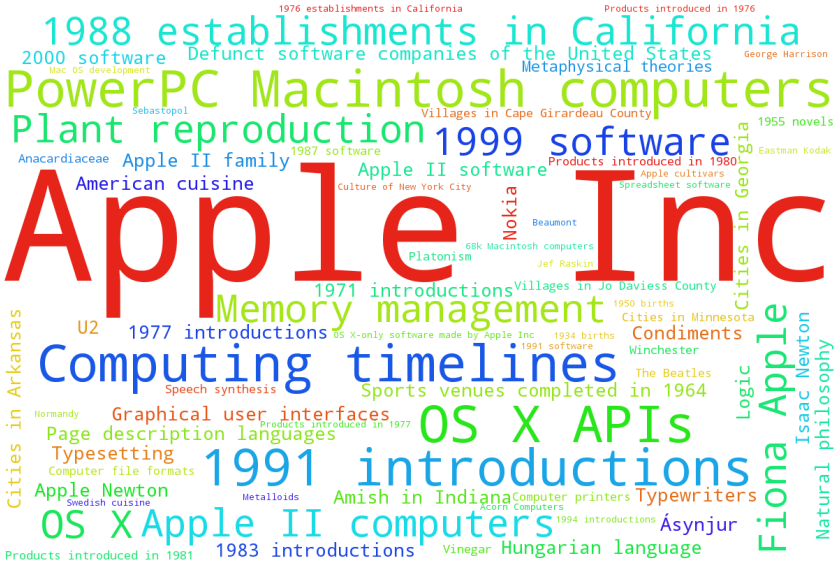

Figure 2: Categories the word ”apple” appears in

If you want to install the package locally for testing don’t use the version from PyPi but instead the following git branch, as otherwise incompatibilities might arise:

```
pip install git+git://github.com/amueller/word_cloud.git@b35ed472b5b93
```

4.1.1 Python Code-Skeleton

```
#!/usr/bin/env python3
from wordcloud import WordCloud
import csv
import re
import sys

csv.field_size_limit(sys.maxsize)

categories = {}
searchword = "apple"
def get_category(text):
    ...

csv_file = open("wikipedia-text.csv", "r")
reader = csv.reader(csv_file)
for _, title, _, text in reader:
    ...

wordcloud = WordCloud(relative_scaling=.5)
wordcloud.generate_from_frequencies(list(categories.items()))

image = wordcloud.to_image()
print(image.size)
image.save(searchword + ".png")
```

Submission:

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