Introduction to Web Science

Assignment 10

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Olga Zagovora
zagovora@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: January 25, 2016, 10:00 a.m.
Tutorial on: January 27, 2016, 12:00 p.m.

For all the assignment questions that require you to write code, **make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.**

Team Name: hotel

Andrea Mildes - mildes@uni-koblenz.de

Sebastian Blei - sblei@uni-koblenz.de

Johannes Kirchner - jkirchner@uni-koblenz.de

Abdul Afghan - abdul.afghan@outlook.de

1 Modeling Twitter data (10 points)

In the meme paper[1] by Weng et al., in Figure 2[2] you find a plot, comparing the system entropy with the average user entropy. Your task is to reproduce the plot and corresponding calculations.

1. We provide you with the file 'onlyhashtag.data', containing a collection of hashtags from tweets. Use this data to reproduce the plot from the paper. Once you have the values for average user entropy and system entropy calculated per day create a scatter plot to display the values.
2. Interpret the scatter plot and compare it with the authors interpretation from the graph showed in the paper. Will the interpretations be compatible to each other or will they contradict each other? Do not write more than 5 sentences.

1.1 Hints

1. Use formulas from the lecture to calculate the entropy for one user and the system entropy.
2. Do not forget to give proper names of plot axes.

Code:

```
1: # assignment 10 task 1
2: # Andrea Mildes - mildes@uni-koblenz.de
3: # Sebastian Blei - sblei@uni-koblenz.de
4: # Johannes Kirchner - jkirchner@uni-koblenz.de
5: # Abdul Afghan - abdul.afghan@outlook.de
6:
7: import logging
8: import time
9: import numpy as np
10: import matplotlib.pyplot as plt
11:
12: # Configure logging and set start time
13: logging.basicConfig(filename='system_entropy.log', level=logging.DEBUG)
14: start_time = 0
15:
16:
17: # Read the given file into a list
18: def read_file(file):
19:     l = []
20:     with open(file) as f:
21:         for line in f:
```

[1] http://www.nature.com/articles/srep00335
[2] Slide 27, Lecture Meme spreading on the Web

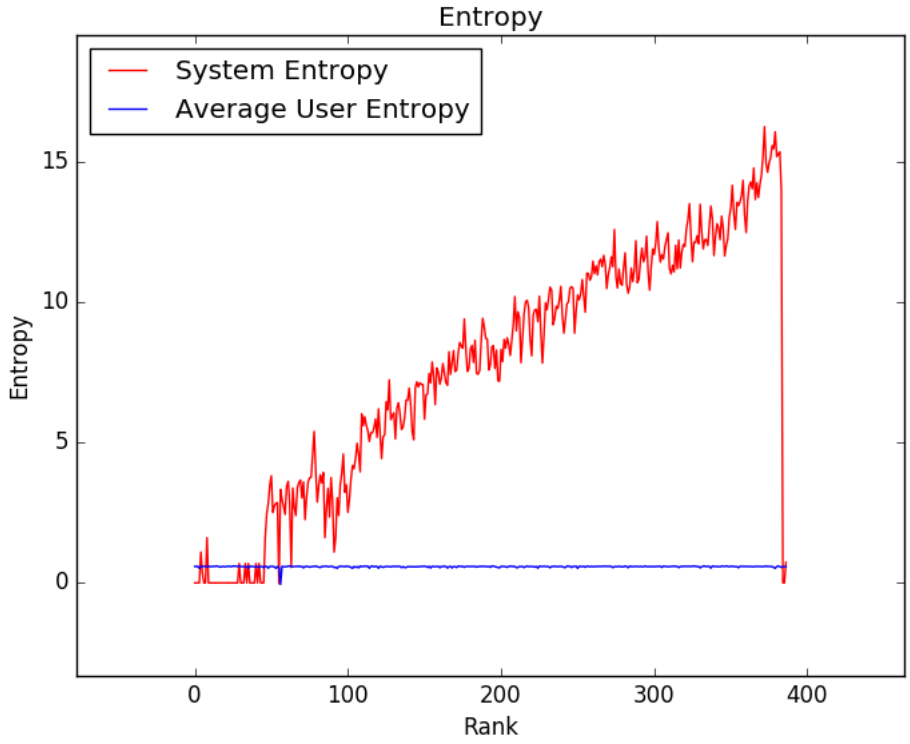
```
22:             l.append(tuple(line.strip().split('\t')))
23:     return l
24:
25:
26: # Creates a dictionary with the date as key and another dictionary
27: # containing the memes and count of memes as value
28: def get_dict_of_memes_by_date(l):
29:     dic = {}
30:     # Sep by date
31:     for line in l:
32:         date = line[1].strip()
33:         hashtag = line[2].lower().strip()
34:         # Check if Date is in dic yet
35:         # If not, add the date and the current hashtag
36:         # If it is, check if the hashtag is already in its set
37:         # If it is, increase count by one
38:         # If its not, add it
39:         if date not in dic:
40:             dic[date] = {}
41:             dic[date][hashtag] = 1
42:         else:
43:             if hashtag in dic[date]:
44:                 dic[date][hashtag] += 1
45:             else:
46:                 dic[date][hashtag] = 1
47:     return dic
48:
49:
50: # Creates a nested dictionary with the following structure:
51: # dic[date][user][hashtag] -> count of the hashtag
52: def get_dict_of_memes_by_date_and_user(l):
53:     dic = {}
54:     # Sep by date
55:     for line in l:
56:         date = line[1].strip()
57:         hashtag = line[2].lower().strip()
58:         user = line[0].lower().strip()
59:         if date not in dic:
60:             dic[date] = {}
61:             dic[date][user] = {}
62:             dic[date][user][hashtag] = 1
63:         else:
64:             if user in dic[date]:
65:                 if hashtag in dic[date][user]:
66:                     dic[date][user][hashtag] += 1
67:                 else:
68:                     dic[date][user][hashtag] = 1
69:             else:
70:                 dic[date][user] = {}
```

```
71:                 dic[date][user][hashtag] = 1
72:     return dic
73:
74:
75: # Calculates the system entropy
76: def calc_system_entropy_per_day(l):
77:     dic = get_dict_of_memes_by_date(l)
78:     entropy_per_day = []
79:
80:     for k, v in dic.items():
81:         dic_len = len(v)
82:         system_entropy = 0
83:         for i, j in v.items():
84:             system_entropy += (j / dic_len) * np.log((j / dic_len))
85:         system_entropy *= -1
86:         entropy_per_day.append((k, system_entropy))
87:
88:     return entropy_per_day
89:
90:
91: def calc_average_user_entropy_per_day(l):
92:     dic = get_dict_of_memes_by_date_and_user(l)
93:     avg_user_entropy_per_day = []
94:     user_entropy = 0
95:     user_entropy_list = []
96:     avg_user_entropy = 0
97:
98:     for k, v in dic.items():
99:         date = k
100:        user_dic = v
101:        for i, j in user_dic.items():
102:            user = i
103:            hashtag_dic = j
104:
105:            # Calculate user entropy
106:            user_entropy = 0
107:            len_hash_dic = len(hashtag_dic)
108:            for x, y in hashtag_dic.items():
109:                user_entropy += (y / len_hash_dic) * np.log((y / len_hash_dic))
110:            user_entropy_list.append(user_entropy)
111:
112:        # calculate average user entropy:
113:        for ue in user_entropy_list:
114:            avg_user_entropy += ue
115:        avg_user_entropy /= len(user_entropy_list)
116:
117:        avg_user_entropy_per_day.append((date, avg_user_entropy))
118:
119:    return avg_user_entropy_per_day
```

```
120:
121:
122: # draw the plot
123: def draw_plot(system_entropy, user_entropy):
124:     temp_system_entropy = []
125:     temp_user_entropy = []
126:
127:     # Sort both lists by date
128:     system_entropy.sort(key=lambda tup: tup[0])
129:     user_entropy.sort(key=lambda tup: tup[0])
130:
131:     for i in system_entropy:
132:         temp_system_entropy.append(i[1])
133:     for i in user_entropy:
134:         temp_user_entropy.append(i[1])
135:
136:     plt.plot(temp_system_entropy, 'r',
137:                 label='System Entropy')
138:     plt.plot(temp_user_entropy, 'b',
139:                 label='Average User Entropy')
140:     plt.title("Entropy")
141:     plt.xlabel("Rank")
142:     plt.ylabel("Entropy")
143:     plt.legend(loc=0)
144:     plt.margins(0.2)
145:     plt.show()
146:
147:     return
148:
149:
150: def main():
151:     global start_time
152:
153:     # Delete content of log file
154:     with open('system_entropy.log', 'w'):
155:         pass
156:
157:     start_time = time.time()
158:
159:     t = str(round((time.time() - start_time), 2))
160:     logging.info("[" + t + "]  Starting :) \n\n")
161:
162:     l = read_file('onlyhash.data')
163:
164:     # Calculate the system entropy per day
165:     system_entropy = calc_system_entropy_per_day(l)
166:
167:     # Calculate the average user entropy per day
168:     user_entropy = calc_average_user_entropy_per_day(l)
```

```
169:
170:     # Draw the plot
171:     draw_plot(system_entropy, user_entropy)
172:
173:     t = str(round((time.time() - start_time), 2))
174:     logging.info("[" + t + "]   " + "FINISHED \n\n")
175:
176:
177: if __name__ == '__main__':
178:     main()
```

Output:



Interpretation:
Comparing our generated plot with the plot provided in the paper, it catches someones eye that both plots are different. While the system entropy in the provided plot rises steadily, the increase over time is much more rapid in our generated graph. However, while the system entropy differs, the average user entropy is roughly the same. This corresponds with the papers observation, that the users breadth of attention remains constant irrespective of the system diversity. So this theory is backed by our plot, because although the system entropy rises over time, the average user entropy stays the same.

2 Measuring inequality (10 points)

We provide you with a sample implementation of the Chinese Restaurant Process[3].

Assume there is a restaurant with an infinite number of tables. When a new customer enters a restaurant he chooses an occupied table or the next empty table with some probabilities.

According to the process first customer always sits at the first table. Probability of the next customer to sit down at an occupied table i equals ratio of guests sitting at the table (c_i/n), where n is the number of guests in the restaurant and c_i is the number of guests sitting at table i.
Probability of customer to choose an empty table equals : $1 - \sum_{i=1}^{S} p_i$, where S is the number of occupied tables and $p_i = c_i/n$.

Provided script simulates the process and returns number of people sitting at each table. We will study restaurants for 1000 customers. Now you should modify the code and evaluate how unequal were the customers' choices of tables.

Calculate the Gini- coefficient measuring the inequality between the tables, until the coefficient stabilizes. Do five different runs and plot your results in a similar way that plots in the lecture slides are done, cf. Slide 32 and Slide 33.

Code:

```
 1: # assignment 10 task 2
 2: # Andrea Mildes - mildes@uni-koblenz.de
 3: # Sebastian Blei - sblei@uni-koblenz.de
 4: # Johannes Kirchner - jkirchner@uni-koblenz.de
 5: # Abdul Afghan - abdul.afghan@outlook.de
 6:
 7: import random
 8: import json
 9: import matplotlib.pyplot as plt
10: import scipy.integrate as integrate
11:
12: def gini_coefficient(network):
13:     network.sort()
14:     sum = 0
15:     sumList = []
16:     equal = [0]
17:     # This is just a helper-loop in order to get the
18:     # Line of Equality
19:     for k in range(1,len(network)):
20:         # this prevents division by 0, if k is 1
21:         if k is 1:
22:             bot = 1
```

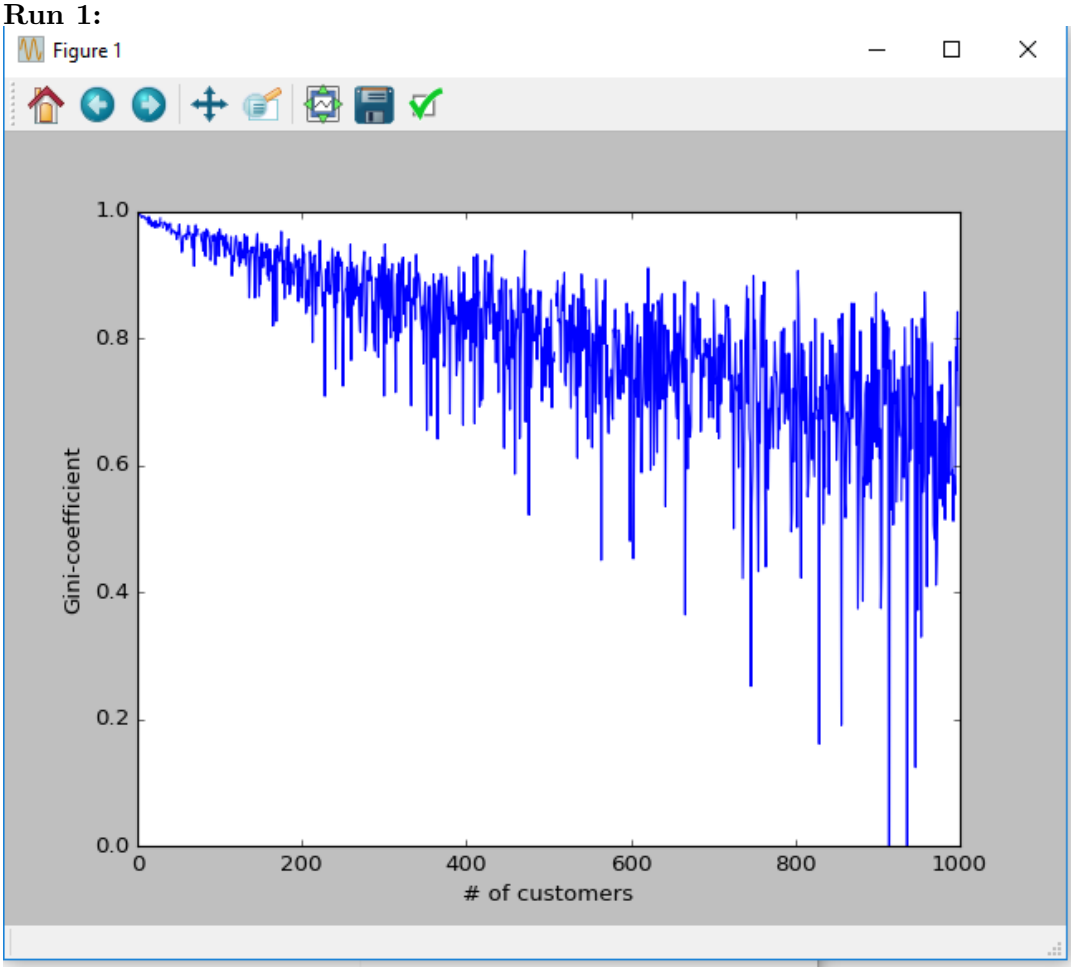
[3]File "chinese_restaurant.py"; Additional information can be found here: https://en.wikipedia.org/wiki/Chinese_restaurant_process

```
23:         else:
24:             bot = len(network) -1
25:         equal.append((k * 1000) / (len(network) -1))
26:     # This loop calculates the Lorenz Curve
27:     for i in range(0,len(network)):
28:         sum += network[i]
29:         sumList.append(sum)
30:     # We calculate the Gini coefficient as: G = A / 0 where
31:     # A is the area of the Line of equality minus the area
32:     # of the Lorenz Curve and 0 the total area beneath the Line
33:     # of equality. The area beneath the lines are calculated
34:     # by using the integrate function of scipy
35:     0 = integrate.simps(equal)
36:     B = integrate.simps(sumList)
37:     A = 0-B
38:     G = A / 0
39:     return G
40:
41: def generateChineseRestaurant(customers):
42:     # First customer always sits at the first table
43:     tables = [1]
44:     # for all other customers do
45:     for cust in range(2, customers+1):
46:             # rand between 0 and 1
47:             rand = random.random()
48:             # Total probability to sit at a table
49:             prob = 0
50:             # No table found yet
51:             table_found = False
52:             # Iterate over tables
53:             for table, guests in enumerate(tables):
54:                 # calc probability for actual table an add it to
55:                 # total probability
56:                 prob += guests / (cust)
57:                 # If rand is smaller than the current total prob.,
58:                 # customer will sit down at current table
59:                 if rand < prob:
60:                     # incr. #customers for that table
61:                     tables[table] += 1
62:                     # customer has found table
63:                     table_found = True
64:                     # no more tables need to be iterated,
65:                     # break out for loop
66:                     break
67:             # If table iteration is over and no table was found,
68:             # open new table
69:             if not table_found:
70:                 tables.append(1)
71:     return tables
```

```
72:
73: ginis = [[],[],[],[],[]]
74:
75: for i in range(0,5):
76:     for j in range(1,1000):
77:         ginis[i].append(gini_coefficient(generateChineseRestaurant(j)))
78:     plt.plot(ginis[i])
79:     plt.xlabel('# of customers')
80:     plt.ylabel('Gini-coefficient')
81:     plt.axis([0,1000,0,1])
82:     plt.show()
```

Output:

Run 1:



Run 2:

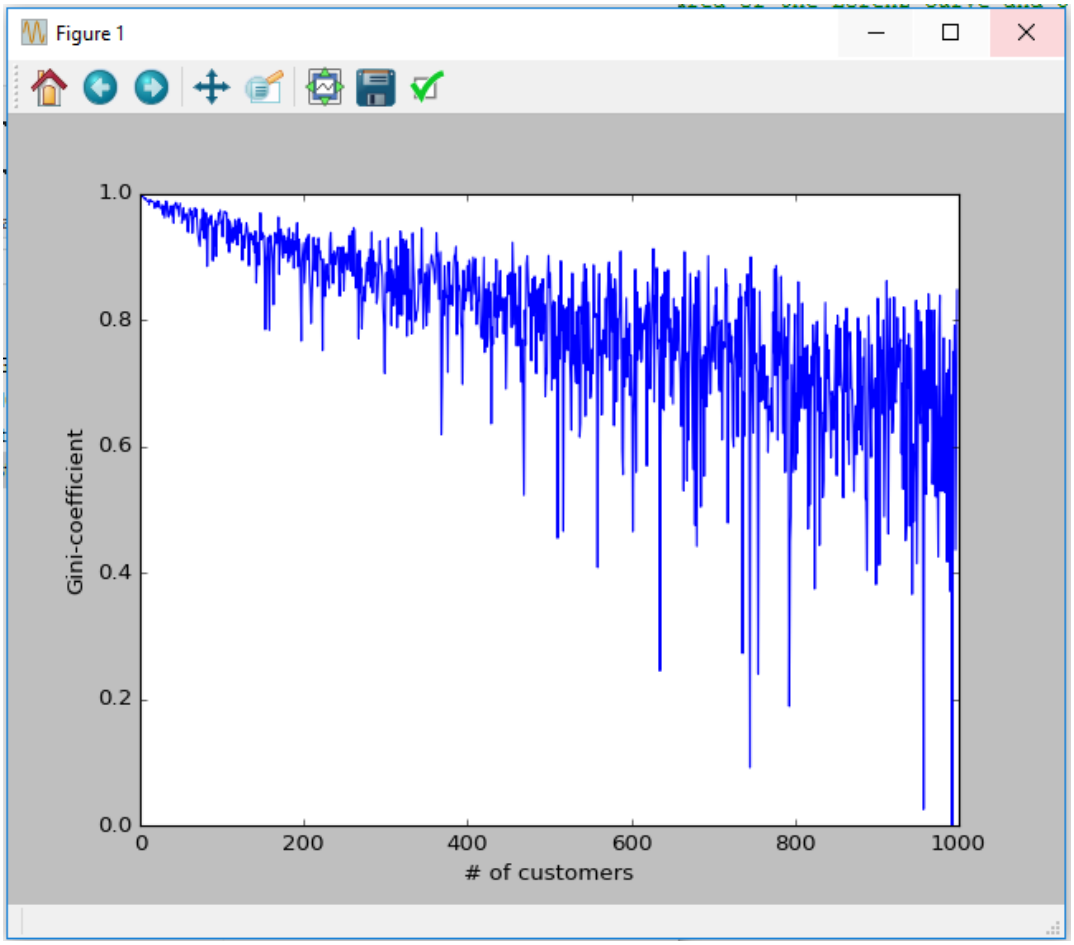


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Gini-coefficient
0
200
400
600
800
1000
# of customers

Run 3:

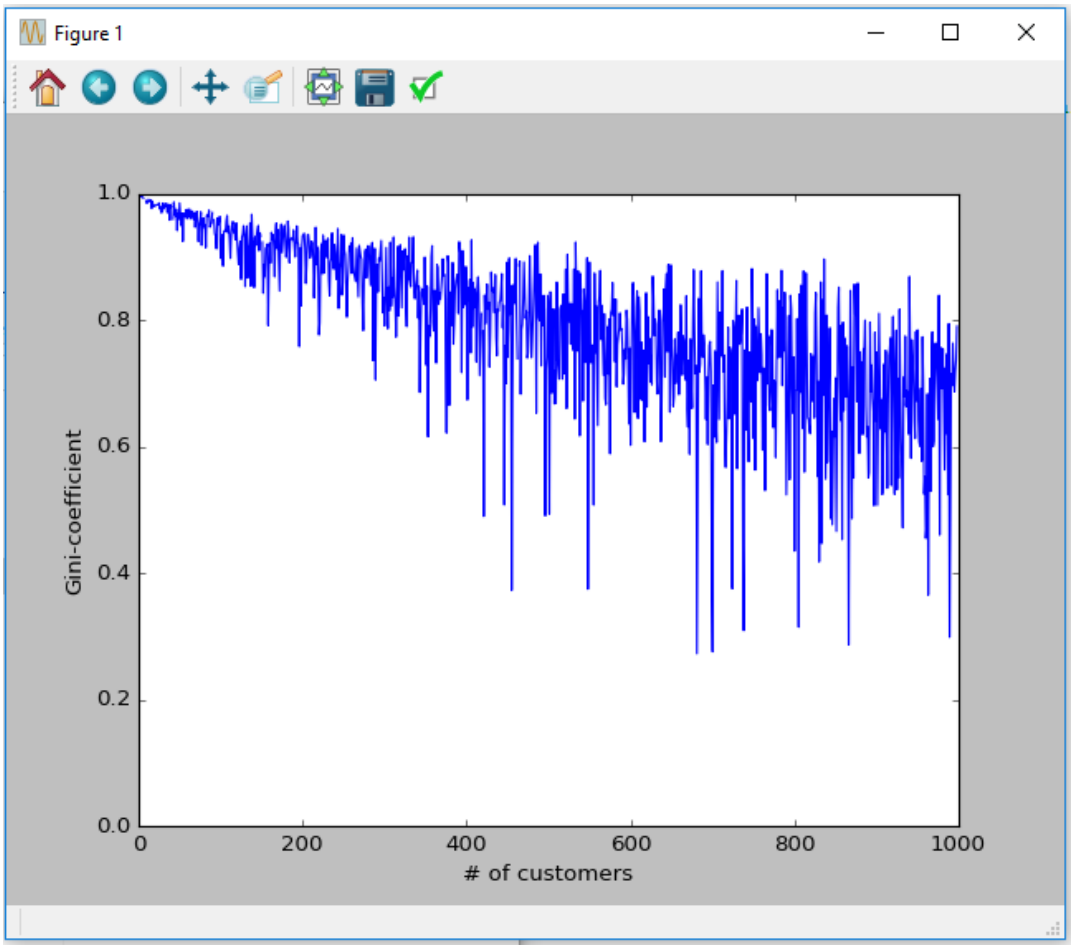


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Gini-coefficient
0
200
400
600
800
1000
# of customers

Run 4:

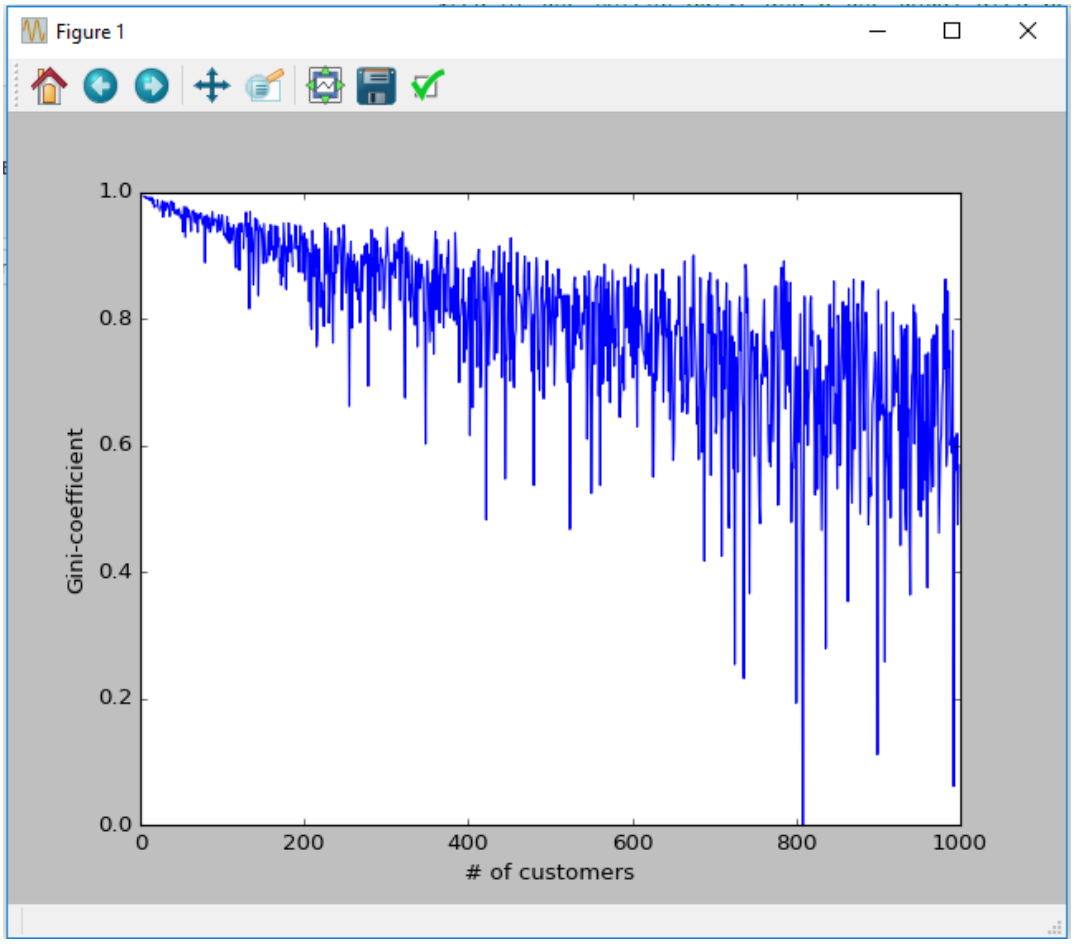


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Gini-coefficient
0
200
400
600
800
1000
# of customers

Run 5:

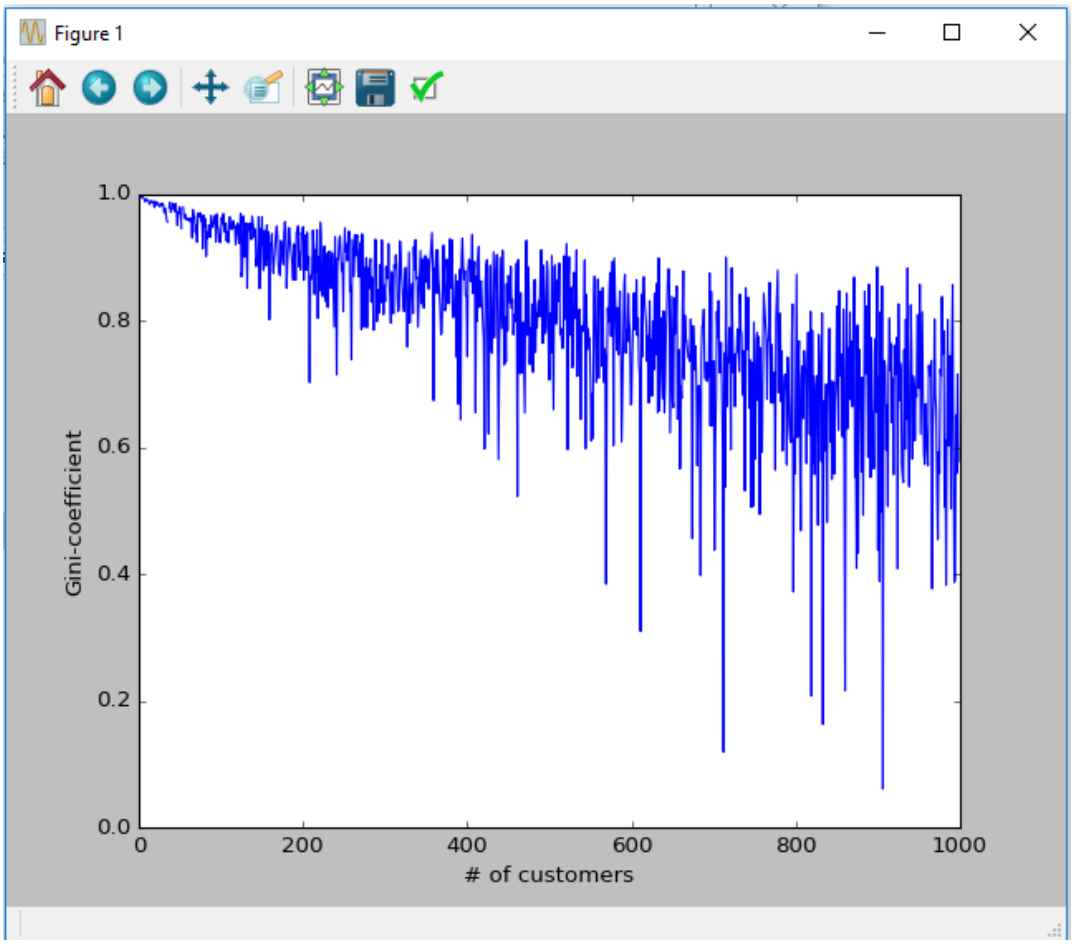


Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
Gini-coefficient
0
200
400
600
800
1000
# of customers

3 Herding (10 points)

Let us consider the altitude of Koblenz to be 74 m above sea level. You are asked to figure out the height of the Ehrenbreitstein Fortress and the Fernmeldeturm Koblenz without googling.
The exercise is split in two parts:

Part 1 : The Secret
In *complete secrecy*, each member of the team will write down their estimated height of the Ehrenbreitstein Fortress without any form of discussion. Please keep in mind that you need to have reasons for your assumption. Once you are done, then openly discuss in the group and present you values in a tabulated format with the reasons each one assumed to arrive at that value.

Part II : The Discussion
Discuss amongst yourself with valid reasoning what could be the height of the Fernmeldeturm Koblenz. Only after discussing, each member of the group is asked to arrive at a value and present this value in a tabulated format as was done in Part I.

Calculate the Mean, Standard Deviation and Variance of your noted results for both the cases and explain briefly what you infer from it.

Note: This exercise is for you to understand the concepts of herding and not to get the perfect height by googling information. There is in fact no point associated with the height but with the complete reasoning that you provide for your answers.

Solution:

Part 1:

Sebastian:	189
Johannes:	274
Andrea:	249

Sebastian: 115 + 74; Sebastian recalled his ride with the ropeway to Ehrenbreitstein and estimated the height by taking the duration and speed of the ride into consideration.

Johannes: 200 + 74; Johannes estimated the height of Ehrenbreitstein by knowing that most reflector posts have a distance of 50 meters to each other. He than stacked this distance between the posts vertically in his mind and thought that 4 times that distance could be as high as the Fortress, which equals 200 meters.

Andrea: 175 + 74; Andrea remembered the view from the fortress down to Koblenz and estimated based on the view.

Mean: $(189 + 274 + 249)/3 = 237,333$

Variance: $1/2*((189-237,333)^2+(274-237,333)^2+(249-237,333)^2) = 1/2*(2336,079+1344,469+136,118889) = 1908,3334445$
Standard Deviation: $sqrt(Variance) = 43,684$

Part 2:
To be honest: After the first discussion we were curious how high Ehrenbreitstein is, so we googled it.
In the second discussion we all agreed that the Karthause (the district where the Fernmeldeturm is located) is roughly as high as Ehrenbreitstein, which is about 118 meters. However, we couldn't agree on the height of the Fernmeldetum itself, so our numbers differ anyways.

Sebastian:	150
Johannes:	160
Andrea:	210

Mean: $(150+160+210)/3 = 173,333$
Variance: $1/2*((150-173,333)^2+(160-173,333)^2+(210-173,333)^2) = 1/2*(544,429+177,769+1344,469) = 1033,333$
Standard Deviation: $sqrt(Variance) = 32,146$

As calculated, the variance and standard deviation of part 1 are higher than the variance and deviation of part 2. Most probably this was caused by our discussion, which led to more similar hight-assumptions. For example, all of us agreed that the height of the Karthause is about 120 meters, which would not have been the case without a discussion.

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment10/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the LaTeXengine to `LuaLaTeX`.