

clip-max-ignore-zeros-asymmetric

August 13, 2020

```
[1]: # rank the obtained results using the *.log files
import os
import pandas as pd
import numpy as np
```

```
[2]: source = "7a"
targetdir = '../../data/' + source + "/"
filelist = sorted(os.listdir(targetdir))
```

```
[3]: filelist
```

```
[3]: ['1.data', '2.data', '3.data', '4.data', '5.data', '6.data', '7.data']
```

```
[4]: # Create dataframe from files
df = pd.DataFrame()

for file in filelist:
    filename = targetdir+file
    col_name = [file]
    temp_df = pd.read_csv(filename,names=col_name)
    df = pd.concat([df, temp_df], axis=1)

# Look at the data
df.head()
```

```
[4]:
```

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	1.0	1.00	1.00	1.00	0.71	1.00	1.0
1	1.0	0.95	-1.00	-1.00	0.69	-1.00	1.0
2	1.0	0.92	-1.00	-1.00	0.68	-1.00	1.0
3	1.0	1.00	0.98	1.00	0.70	0.55	1.0
4	1.0	0.92	0.98	0.99	0.68	0.54	1.0

```
[5]: # Clip values > 1 with 1 and ignore 0s
df.mask(df > 1, 1, inplace=True)
df.mask(df <= 0, np.NaN, inplace=True)
```

```
[6]: # Count NaN values
df.isna().sum()
```

```
[6]: 1.data    162
     2.data     0
     3.data   441
     4.data   486
     5.data     0
     6.data   297
     7.data     0
     dtype: int64
```

```
[7]: df
```

```
[7]:
```

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	1.0	1.00	1.00	1.00	0.71	1.00	1.0
1	1.0	0.95	NaN	NaN	0.69	NaN	1.0
2	1.0	0.92	NaN	NaN	0.68	NaN	1.0
3	1.0	1.00	0.98	1.00	0.70	0.55	1.0
4	1.0	0.92	0.98	0.99	0.68	0.54	1.0
..
724	NaN	1.00	NaN	NaN	0.72	NaN	1.0
725	NaN	1.00	NaN	NaN	0.72	NaN	1.0
726	1.0	1.00	NaN	NaN	0.73	0.86	1.0
727	1.0	1.00	NaN	NaN	0.71	1.00	1.0
728	1.0	1.00	1.00	1.00	0.70	1.00	1.0

[729 rows x 7 columns]

```
[8]: # Save processed dataframe as csv file
df.to_csv("../data/processed/asymmetric/" + source + ".csv", index=False)
```

```
[9]: # Creating ranked dataframe
ranked_df = pd.DataFrame()
stats_df = pd.DataFrame()
```

```
[10]: # going through every column
for column in df:
    wwtp = column[0]

    # In every column, drop na values
    asym_column = df[column].dropna()

    # and calculate individual tao
    tao = len(asym_column)

    # calculate mean
```

```

avg_eff = round(asm_column.mean(),3)

# calculate max
max_eff = round(asm_column.max(),3)

# calculate min
min_eff = round(asm_column.min(),3)

# calculate amplitude
amplitude = round((max_eff - min_eff)*100,2)

amp_str = "Amplitude (max-min)%"

# print stats results
print("WWTP",wwtp,
      "\nMean =",avg_eff,"Maximum =",max_eff,"Minimum =",min_eff,
↪amp_str,"=",amplitude)

stats_df = stats_df.append({ 'WWTP': wwtp,
                             "Mean": avg_eff, "Maximum" : max_eff, "Minimum": min_eff, amp_str:
↪amplitude},ignore_index=True)

# Calculating Sk sum of factors
Sk = round(asm_column.sum(),3)

# Calculating ek sum of factors of 1 (or above if errors in calculation)
ek = asm_column >= 1
ek = ek.sum()

# Calculating R1k ek/tao
R1k = round(ek/tao,3)

# Calculate R2k
if tao != ek:
    R2k = (Sk - ek)/(tao - ek)
elif R1k == 1:
    R2k = 0

R2k = round(R2k,3)

# Printing results
print("tao =",tao,"| ek =",ek,"| R1k =",R1k, "| Sk =",Sk, "| R2k
↪=",R2k,"\n")

# Populate ranking dataframe using pd.df.append
# Using unicode to name columns with super and subscripts
R1k_col = 'R\u00B9\u2096\u2080'

```

```
R2k_col = 'R\u00B2\u2096\u2080'
ranked_df = ranked_df.append({ R2k_col:R2k, R1k_col: R1k, 'WWTP':\u2192wwtp},ignore_index=True)
```

WWTP 1

Mean = 1.0 Maximum = 1.0 Minimum = 1.0 Amplitude (max-min)(%) = 0.0
 tao = 567 | ek = 567 | R1k = 1.0 | Sk = 567.0 | R2k = 0

WWTP 2

Mean = 0.979 Maximum = 1.0 Minimum = 0.84 Amplitude (max-min)(%) = 16.0
 tao = 729 | ek = 447 | R1k = 0.613 | Sk = 713.97 | R2k = 0.947

WWTP 3

Mean = 0.974 Maximum = 1.0 Minimum = 0.93 Amplitude (max-min)(%) = 7.0
 tao = 288 | ek = 54 | R1k = 0.188 | Sk = 280.57 | R2k = 0.968

WWTP 4

Mean = 0.995 Maximum = 1.0 Minimum = 0.97 Amplitude (max-min)(%) = 3.0
 tao = 243 | ek = 150 | R1k = 0.617 | Sk = 241.89 | R2k = 0.988

WWTP 5

Mean = 0.712 Maximum = 0.78 Minimum = 0.63 Amplitude (max-min)(%) = 15.0
 tao = 729 | ek = 0 | R1k = 0.0 | Sk = 518.88 | R2k = 0.712

WWTP 6

Mean = 0.722 Maximum = 1.0 Minimum = 0.51 Amplitude (max-min)(%) = 49.0
 tao = 432 | ek = 105 | R1k = 0.243 | Sk = 311.7 | R2k = 0.632

WWTP 7

Mean = 1.0 Maximum = 1.0 Minimum = 1.0 Amplitude (max-min)(%) = 0.0
 tao = 729 | ek = 729 | R1k = 1.0 | Sk = 729.0 | R2k = 0

0.1 Ranking of WWTP

```
[11]: # Reorder columns to be usable as a results table
ranked_df = ranked_df.reindex(columns=['WWTP',R1k_col, R2k_col])
```

```
[12]: ranked_df
```

```
[12]:
```

	WWTP	R ¹	R ²
0	1	1.000	0.000
1	2	0.613	0.947
2	3	0.188	0.968
3	4	0.617	0.988
4	5	0.000	0.712

```

5    6    0.243    0.632
6    7    1.000    0.000

```

```

[13]: import os

path = "../../results/" + source + "/asymmetric"

# Save rankings dataframe as csv file

try:
    ranked_df.to_csv(path + "/ranking.csv",index=False)
    print("Save succesful")
except:
    print("Creating folder and saving")
    os.mkdir(path)
    ranked_df.to_csv(path + "/ranking.csv",index=False)

```

Save succesful

0.2 Calculate Descriptive Statistics

```

[14]: # Calculate the mean of every column
mean_mean = round(stats_df.Mean.mean(),3)
mean_max = round(stats_df.Maximum.mean(),3)
mean_min = round(stats_df.Minimum.mean(),3)
mean_amp = round(stats_df[amp_str].mean(),3)

```

```

[15]: # Add means to stats dataframe
stats_df = stats_df.append({ 'WWTP': "Mean", "Mean" : mean_mean, "Maximum" : ↵
↵mean_max,
                             "Minimum" : mean_min, amp_str : ↵
↵mean_amp},ignore_index=True)

```

```

[16]: # Calculate the standard deviation of every column
sd_mean = round(stats_df.Mean.std(),3)
sd_max = round(stats_df.Maximum.std(),3)
sd_min = round(stats_df.Minimum.std(),3)
sd_amp = round(stats_df[amp_str].std(),3)

```

```

[17]: # Add means to stats dataframe
stats_df = stats_df.append({ 'WWTP': "SD", "Mean" : sd_mean, "Maximum" : sd_max,
                             "Minimum" : sd_min, amp_str : ↵
↵sd_amp},ignore_index=True)

```

```

[18]: # Reorder columns

```

```
stats_df = stats_df.reindex(columns=["WWTP", "Mean", "Maximum", "Minimum", "Amplitude (max-min) (%)", "SD"])
```

```
[19]: stats_df
```

```
[19]:
```

	WWTP	Mean	Maximum	Minimum	Amplitude (max-min) (%)
0	1	1.000	1.000	1.000	0.000
1	2	0.979	1.000	0.840	16.000
2	3	0.974	1.000	0.930	7.000
3	4	0.995	1.000	0.970	3.000
4	5	0.712	0.780	0.630	15.000
5	6	0.722	1.000	0.510	49.000
6	7	1.000	1.000	1.000	0.000
7	Mean	0.912	0.969	0.840	12.857
8	SD	0.124	0.077	0.181	15.959

```
[20]: # Save statistics dataframe as csv file
stats_df.to_csv(path + "/statistics.csv", index=False)
```

```
[21]: # Convert Jupyter Notebook to PDF LaTeX file
!jupyter-nbconvert --to pdf "clip-max-ignore-zeros-asymmetric" --output-dir "../results/7a/asymmetric"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-asymmetric.ipynb to pdf
[NbConvertApp] Writing 44340 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 47286 bytes to ../../results/6d/asymmetric/clip-max-ignore-zeros-asymmetric.pdf
```

```
[ ]:
```