

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 = "10"
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	0.65	0.86	1.00	1.00	0.62	1.0	0.65
1	0.65	0.86	-1.00	-1.00	0.62	-1.0	0.65
2	0.65	0.86	-1.00	-1.00	0.62	-1.0	0.65
3	0.62	0.83	0.99	0.96	0.59	0.7	0.62
4	0.62	0.83	0.99	0.96	0.59	0.8	0.62

```
[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      0
     2.data      0
     3.data    342
     4.data    324
     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      0.65   0.86   1.00   1.00   0.62   1.00   0.65
1      0.65   0.86   NaN    NaN    0.62   NaN    0.65
2      0.65   0.86   NaN    NaN    0.62   NaN    0.65
3      0.62   0.83   0.99   0.96   0.59   0.70   0.62
4      0.62   0.83   0.99   0.96   0.59   0.80   0.62
..      ...     ...     ...     ...     ...     ...     ...
724    0.62   0.88   NaN    NaN    0.65   NaN    0.62
725    0.62   0.88   NaN    NaN    0.65   NaN    0.62
726    0.60   0.84   1.00   1.00   0.63   0.84   0.60
727    0.60   0.84   1.00   1.00   0.63   1.00   0.60
728    0.60   0.84   1.00   1.00   0.63   1.00   0.60
```

[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 = 0.623 Maximum = 0.65 Minimum = 0.6 Amplitude (max-min)(%) = 5.0
 tao = 729 | ek = 0 | R1k = 0.0 | Sk = 454.41 | R2k = 0.623

WWTP 2

Mean = 0.851 Maximum = 0.94 Minimum = 0.78 Amplitude (max-min)(%) = 16.0
 tao = 729 | ek = 0 | R1k = 0.0 | Sk = 620.28 | R2k = 0.851

WWTP 3

Mean = 0.99 Maximum = 1.0 Minimum = 0.97 Amplitude (max-min)(%) = 3.0
 tao = 387 | ek = 144 | R1k = 0.372 | Sk = 383.16 | R2k = 0.984

WWTP 4

Mean = 0.969 Maximum = 1.0 Minimum = 0.92 Amplitude (max-min)(%) = 8.0
 tao = 405 | ek = 162 | R1k = 0.4 | Sk = 392.4 | R2k = 0.948

WWTP 5

Mean = 0.628 Maximum = 0.68 Minimum = 0.57 Amplitude (max-min)(%) = 11.0
 tao = 729 | ek = 0 | R1k = 0.0 | Sk = 457.92 | R2k = 0.628

WWTP 6

Mean = 0.757 Maximum = 1.0 Minimum = 0.52 Amplitude (max-min)(%) = 48.0
 tao = 432 | ek = 105 | R1k = 0.243 | Sk = 326.92 | R2k = 0.679

WWTP 7

Mean = 0.623 Maximum = 0.65 Minimum = 0.6 Amplitude (max-min)(%) = 5.0
 tao = 729 | ek = 0 | R1k = 0.0 | Sk = 454.41 | R2k = 0.623

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	0.000	0.623
1	2	0.000	0.851
2	3	0.372	0.984
3	4	0.400	0.948
4	5	0.000	0.628

```

5    6    0.243    0.679
6    7    0.000    0.623

```

```

[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	0.623	0.650	0.600	5.000
1	2	0.851	0.940	0.780	16.000
2	3	0.990	1.000	0.970	3.000
3	4	0.969	1.000	0.920	8.000
4	5	0.628	0.680	0.570	11.000
5	6	0.757	1.000	0.520	48.000
6	7	0.623	0.650	0.600	5.000
7	Mean	0.777	0.846	0.709	13.714
8	SD	0.150	0.162	0.168	14.577

```
[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/10/asymmetric"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-asymmetric.ipynb to pdf
[NbConvertApp] Writing 44316 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 47176 bytes to ../../results/9c/asymmetric/clip-max-ignore-zeros-asymmetric.pdf
```

```
[ ]:
```