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 = "8a"
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	0.77	0.85	1.00	1.00	0.63	1.00	0.60
1	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
2	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
3	0.09	0.10	0.09	0.12	0.06	0.07	0.07
4	0.09	0.10	0.09	0.12	0.06	0.07	0.07

[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    315
2.data    306
3.data    333
4.data    324
5.data    315
6.data    342
7.data    315
dtype: int64
```

[7]:
```
df
```

[7]:

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	0.77	0.85	1.00	1.00	0.63	1.00	0.60
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	0.09	0.10	0.09	0.12	0.06	0.07	0.07
4	0.09	0.10	0.09	0.12	0.06	0.07	0.07
..	…	…	…	…	…	…	…
724	1.00	1.00	NaN	NaN	1.00	NaN	1.00
725	NaN	NaN	NaN	NaN	NaN	NaN	NaN
726	0.77	0.85	0.98	1.00	0.63	0.58	0.60
727	0.77	0.85	0.98	1.00	0.63	0.58	0.60
728	0.77	0.85	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(asym_column.mean(),3)

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

    # calculate min
    min_eff = round(asym_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(asym_column.sum(),3)

    # Calculating ek sum of factors of 1 (or above if errors in calculation)
    ek = asym_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':␣
↪wwtp},ignore_index=True)
```

```
WWTP 1
Mean = 0.469 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 414 | ek = 94 | R1k = 0.227 | Sk = 194.03 | R2k = 0.313

WWTP 2
Mean = 0.483 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 423 | ek = 94 | R1k = 0.222 | Sk = 204.23 | R2k = 0.335

WWTP 3
Mean = 0.499 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 396 | ek = 109 | R1k = 0.275 | Sk = 197.73 | R2k = 0.309

WWTP 4
Mean = 0.51 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 405 | ek = 145 | R1k = 0.358 | Sk = 206.39 | R2k = 0.236

WWTP 5
Mean = 0.413 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 414 | ek = 94 | R1k = 0.227 | Sk = 170.84 | R2k = 0.24

WWTP 6
Mean = 0.433 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 387 | ek = 107 | R1k = 0.276 | Sk = 167.38 | R2k = 0.216

WWTP 7
Mean = 0.415 Maximum = 1.0 Minimum = 0.01 Amplitude (max-min)(%) = 99.0
tao = 414 | ek = 93 | R1k = 0.225 | Sk = 171.79 | R2k = 0.245
```

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	0.227	0.313
1	2	0.222	0.335
2	3	0.275	0.309
3	4	0.358	0.236
4	5	0.227	0.240

```
5    6  0.276  0.216
6    7  0.225  0.245
```

[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",␣
↪amp_str])
```

[19]:
```
stats_df
```

[19]:

	WWTP	Mean	Maximum	Minimum	Amplitude (max-min)(%)
0	1	0.469	1.0	0.01	99.0
1	2	0.483	1.0	0.01	99.0
2	3	0.499	1.0	0.01	99.0
3	4	0.510	1.0	0.01	99.0
4	5	0.413	1.0	0.01	99.0
5	6	0.433	1.0	0.01	99.0
6	7	0.415	1.0	0.01	99.0
7	Mean	0.460	1.0	0.01	99.0
8	SD	0.037	0.0	0.00	0.0

[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/8a/asymmetric"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-asymmetric.ipynb to pdf
[NbConvertApp] Writing 44346 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 47187 bytes to ../../results/7c/asymmetric/clip-max-
ignore-zeros-asymmetric.pdf
```

[ ]:
```

```