clip-max-ignore-zeros-custom

August 11, 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    1.00    0.92    1.00    1.00    0.68    1.00     1.0
1    1.02    0.92    0.00    0.00    0.68    0.00     0.0
2    1.05    0.92    0.00    0.00    0.68    0.00     0.0
3    0.99    0.84    0.96    0.91    0.63    0.96     1.0
4    0.99    0.84    0.96    0.91    0.63    0.00     0.0
```

[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      0
     2.data      0
     3.data    324
     4.data    324
     5.data      0
     6.data    297
     7.data    324
     dtype: int64
```

```
[7]: # Ignore invalid values by dropping them from the dataframe
     df = df.dropna()
```

```
[8]: df
```

```
[8]:      1.data  2.data  3.data  4.data  5.data  6.data  7.data
     0      1.00    0.92    1.00    1.00    0.68    1.00    1.00
     3      0.99    0.84    0.96    0.91    0.63    0.96    1.00
     6      0.99    0.77    0.91    0.84    0.58    0.53    1.00
     9      1.00    0.92    1.00    1.00    0.69    1.00    1.00
     12     1.00    0.85    0.97    0.92    0.64    0.97    1.00
     ..      …       …       …       …       …       …       …
     718    0.99    0.79    0.99    0.99    0.58    0.54    0.99
     719    0.99    0.79    0.99    0.99    0.58    0.54    0.99
     726    1.00    0.79    1.00    1.00    0.58    0.54    1.00
     727    1.00    0.79    1.00    1.00    0.58    0.54    1.00
     728    1.00    0.79    1.00    1.00    0.58    0.54    1.00

     [324 rows x 7 columns]
```

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

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

```
[11]: # Creating scenario quantity variable
      tao = len(df)
      tao
```

```
[11]: 324
```

0.1 Ranking of WWTP

[12]:
```
for column in df:
    wwtp = column[0]

    # TODO: get original (pre-analysis) value
    # pending

    # calculate mean
    avg_eff = round(df[column].mean(),3)

    # calculate max
    max_eff = round(df[column].max(),3)

    # calculate min
    min_eff = round(df[column].min(),3)

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

    amp_str = "Amplitude (max-min)(%)"

    # print stats results
    print("WWTP", wwtp,"Mean =",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)


    # TODO: Populate statistics dataframe using pd.df.append

    # Calculating Sk sum of factors
    Sk = round(df[column].sum(),3)

    # Calculating ek sum of factors of 1 (or above if errors in calculation)
    ek = df[column] >= 1
    ek = ek.sum()
    print("ek =",ek)

    # 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("WWTP", wwtp,"| ek =",ek,"| R1k =",R1k, "| Sk =",Sk, "| R2k =",R2k)

    # 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 = 0.994 Maximum = 1.0 Minimum = 0.98 Amplitude (max-min)(%) = 2.0
ek = 162
WWTP 1 | ek = 162 | R1k = 0.5 | Sk = 322.08 | R2k = 0.988
WWTP 2 Mean = 0.827 Maximum = 0.93 Minimum = 0.77 Amplitude (max-min)(%) = 16.0
ek = 0
WWTP 2 | ek = 0 | R1k = 0.0 | Sk = 268.02 | R2k = 0.827
WWTP 3 Mean = 0.966 Maximum = 1.0 Minimum = 0.88 Amplitude (max-min)(%) = 12.0
ek = 108
WWTP 3 | ek = 108 | R1k = 0.333 | Sk = 313.08 | R2k = 0.949
WWTP 4 Mean = 0.944 Maximum = 1.0 Minimum = 0.83 Amplitude (max-min)(%) = 17.0
ek = 108
WWTP 4 | ek = 108 | R1k = 0.333 | Sk = 305.7 | R2k = 0.915
WWTP 5 Mean = 0.617 Maximum = 0.7 Minimum = 0.57 Amplitude (max-min)(%) = 13.0
ek = 0
WWTP 5 | ek = 0 | R1k = 0.0 | Sk = 199.86 | R2k = 0.617
WWTP 6 Mean = 0.736 Maximum = 1.0 Minimum = 0.53 Amplitude (max-min)(%) = 47.0
ek = 36
WWTP 6 | ek = 36 | R1k = 0.111 | Sk = 238.31 | R2k = 0.702
WWTP 7 Mean = 0.997 Maximum = 1.0 Minimum = 0.99 Amplitude (max-min)(%) = 1.0
ek = 216
WWTP 7 | ek = 216 | R1k = 0.667 | Sk = 322.92 | R2k = 0.99
```

[13]:
```
# Reorder columns to be usable as a results table
ranked_df = ranked_df.reindex(columns=['WWTP',R1k_col, R2k_col])
```

[14]:
```
ranked_df
```

[14]:
```
   WWTP     R¹      R²
0     1  0.500   0.988
1     2  0.000   0.827
2     3  0.333   0.949
3     4  0.333   0.915
4     5  0.000   0.617
5     6  0.111   0.702
```

```
6     7  0.667  0.990
```

[15]:
```
import os

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

# 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

[16]:
```
# 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)
```

[17]:
```
# 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)
```

[18]:
```
# 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)
```

[19]:
```
# 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)
```

[20]:
```
# Reorder columns
stats_df = stats_df.reindex(columns=["WWTP", "Mean", "Maximum", "Minimum",␣
 ↪amp_str])
```

```
[21]: stats_df
```

```
[21]:    WWTP   Mean  Maximum  Minimum  Amplitude (max-min)(%)
      0     1  0.994    1.000    0.980                   2.000
      1     2  0.827    0.930    0.770                  16.000
      2     3  0.966    1.000    0.880                  12.000
      3     4  0.944    1.000    0.830                  17.000
      4     5  0.617    0.700    0.570                  13.000
      5     6  0.736    1.000    0.530                  47.000
      6     7  0.997    1.000    0.990                   1.000
      7  Mean  0.869    0.947    0.793                  15.429
      8    SD  0.136    0.104    0.170                  14.171
```

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

```
[23]: # Convert Jupyter Notebook to PDF LaTeX file
      !jupyter-nbconvert --to pdf "clip-max-ignore-zeros-custom.ipynb" --output-dir ".
       ↪./../results/8a/"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-custom.ipynb to pdf
[NbConvertApp] Writing 45170 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 47177 bytes to ../../results/8a/clip-max-ignore-zeros-
custom.pdf
```

```
[ ]:
```