clip-max-ignore-zeros-custom

August 6, 2020

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

```python
[2]: source = "9c"
     targetdir = '../../data/' + source + "/"
     filelist = sorted(os.listdir(targetdir))
```

```python
[3]: filelist
```

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

```python
[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    0.87    1.00    1.00    0.64    1.00     1.0
     1     1.0    0.87   -1.00   -1.00    0.64   -1.00     1.0
     2     1.0    0.87   -1.00   -1.00    0.64   -1.00     1.0
     3     1.0    0.84    0.99    0.96    0.61    0.72     1.0
     4     1.0    0.84    0.99    0.96    0.61    0.82     1.0
```

```python
[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    0
     4.data    0
     5.data    0
     6.data    0
     7.data    0
     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         1.0    0.87    1.00    1.00    0.64    1.00     1.0
     1         1.0    0.87   -1.00   -1.00    0.64   -1.00     1.0
     2         1.0    0.87   -1.00   -1.00    0.64   -1.00     1.0
     3         1.0    0.84    0.99    0.96    0.61    0.72     1.0
     4         1.0    0.84    0.99    0.96    0.61    0.82     1.0
     ..        ...     ...     ...     ...     ...     ...     ...
     724       1.0    0.89   -1.00   -1.00    0.67   -1.00     1.0
     725       1.0    0.89   -1.00   -1.00    0.67   -1.00     1.0
     726       1.0    0.85    1.00    1.00    0.65    0.86     1.0
     727       1.0    0.85    1.00    1.00    0.65    1.00     1.0
     728       1.0    0.85    1.00    1.00    0.65    1.00     1.0

     [729 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]: 729
```

0.1 Ranking of WWTP

```python
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
```

```python
    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 = 1.0 Maximum = 1.0 Minimum = 1.0 Amplitude (max-min)(%) = 0.0
ek = 729
WWTP 1 | ek = 729 | R1k = 1.0 | Sk = 729.0 | R2k = 0
WWTP 2 Mean = 0.86 Maximum = 0.95 Minimum = 0.79 Amplitude (max-min)(%) = 16.0
ek = 0
WWTP 2 | ek = 0 | R1k = 0.0 | Sk = 627.3 | R2k = 0.86
WWTP 3 Mean = 0.056 Maximum = 1.0 Minimum = -1.0 Amplitude (max-min)(%) = 200.0
ek = 144
WWTP 3 | ek = 144 | R1k = 0.198 | Sk = 41.18 | R2k = -0.176
WWTP 4 Mean = 0.094 Maximum = 1.0 Minimum = -1.0 Amplitude (max-min)(%) = 200.0
ek = 162
WWTP 4 | ek = 162 | R1k = 0.222 | Sk = 68.4 | R2k = -0.165
WWTP 5 Mean = 0.647 Maximum = 0.7 Minimum = 0.59 Amplitude (max-min)(%) = 11.0
ek = 0
WWTP 5 | ek = 0 | R1k = 0.0 | Sk = 471.69 | R2k = 0.647
WWTP 6 Mean = 0.048 Maximum = 1.0 Minimum = -1.0 Amplitude (max-min)(%) = 200.0
ek = 105
WWTP 6 | ek = 105 | R1k = 0.144 | Sk = 34.93 | R2k = -0.112
WWTP 7 Mean = 1.0 Maximum = 1.0 Minimum = 1.0 Amplitude (max-min)(%) = 0.0
ek = 729
WWTP 7 | ek = 729 | R1k = 1.0 | Sk = 729.0 | R2k = 0
```

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

```python
[14]: ranked_df
```

```
[14]:    WWTP    R¹      R²
      0     1  1.000   0.000
      1     2  0.000   0.860
      2     3  0.198  -0.176
      3     4  0.222  -0.165
      4     5  0.000   0.647
      5     6  0.144  -0.112
```

```
6    7  1.000  0.000
```

```python
[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)
```

```
Creating folder and saving
```

0.2 Calculate Descriptive Statistics

```python
[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)
```

```python
[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)
```

```python
[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)
```

```python
[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)
```

```python
[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  1.000    1.000    1.000                   0.000
       1      2  0.860    0.950    0.790                  16.000
       2      3  0.056    1.000   -1.000                 200.000
       3      4  0.094    1.000   -1.000                 200.000
       4      5  0.647    0.700    0.590                  11.000
       5      6  0.048    1.000   -1.000                 200.000
       6      7  1.000    1.000    1.000                   0.000
       7   Mean  0.529    0.950    0.054                  89.571
       8     SD  0.416    0.104    0.922                  95.779
```

```
[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/9c/"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-custom.ipynb to pdf
[NbConvertApp] Writing 45138 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 47064 bytes to ../../results/9b/clip-max-ignore-zeros-
custom.pdf
```

```
[ ]:
```