clip-max-ignore-zeros-custom

August 2, 2020

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

```python
[2]: source = "6d"
     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    0.67    0.86    1.00    1.00    0.62    1.00    0.67
     1    0.67    0.86   -1.00   -1.00    0.62   -1.00    0.67
     2    0.67    0.86   -1.00   -1.00    0.62   -1.00    0.67
     3    0.62    0.79    0.98    0.92    0.57    0.53    0.62
     4    0.62    0.79    0.98    0.92    0.57    0.53    0.62
```

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

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

```
[6]: 1.data     81
     2.data      0
     3.data    342
     4.data    324
     5.data      0
     6.data    297
     7.data      0
     dtype: int64
```

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

```python
[8]: df
```

```
[8]:      1.data  2.data  3.data  4.data  5.data  6.data  7.data
     0      0.67    0.86    1.00    1.00    0.62    1.00    0.67
     3      0.62    0.79    0.98    0.92    0.57    0.53    0.62
     4      0.62    0.79    0.98    0.92    0.57    0.53    0.62
     6      0.57    0.73    0.95    0.85    0.53    0.49    0.57
     7      0.57    0.73    0.95    0.85    0.53    0.49    0.57
     ..      ...     ...     ...     ...     ...     ...
     718    0.85    0.85    0.99    1.00    0.62    1.00    0.58
     719    0.85    0.85    0.99    1.00    0.62    1.00    0.58
     726    0.85    0.85    0.99    1.00    0.63    0.85    0.58
     727    0.86    0.85    1.00    1.00    0.63    1.00    0.58
     728    0.86    0.85    1.00    1.00    0.63    1.00    0.58

     [387 rows x 7 columns]
```

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

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

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

```
[11]: 387
```

1 Ranking of WWTP

```python
[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:
```

3

```
        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 = 0.644 Maximum = 0.86 Minimum = 0.57 Amplitude (max-min)(%) = 29.0
ek = 0
WWTP 1 | ek = 0 | R1k = 0.0 | Sk = 249.15 | R2k = 0.644
WWTP 2 Mean = 0.801 Maximum = 0.87 Minimum = 0.71 Amplitude (max-min)(%) = 16.0
ek = 0
WWTP 2 | ek = 0 | R1k = 0.0 | Sk = 309.87 | R2k = 0.801
WWTP 3 Mean = 0.979 Maximum = 1.0 Minimum = 0.93 Amplitude (max-min)(%) = 7.0
ek = 107
WWTP 3 | ek = 107 | R1k = 0.276 | Sk = 378.72 | R2k = 0.97
WWTP 4 Mean = 0.94 Maximum = 1.0 Minimum = 0.84 Amplitude (max-min)(%) = 16.0
ek = 126
WWTP 4 | ek = 126 | R1k = 0.326 | Sk = 363.79 | R2k = 0.911
WWTP 5 Mean = 0.587 Maximum = 0.64 Minimum = 0.52 Amplitude (max-min)(%) = 12.0
ek = 0
WWTP 5 | ek = 0 | R1k = 0.0 | Sk = 227.08 | R2k = 0.587
WWTP 6 Mean = 0.689 Maximum = 1.0 Minimum = 0.48 Amplitude (max-min)(%) = 52.0
ek = 84
WWTP 6 | ek = 84 | R1k = 0.217 | Sk = 266.75 | R2k = 0.603
WWTP 7 Mean = 0.606 Maximum = 0.68 Minimum = 0.57 Amplitude (max-min)(%) = 11.0
ek = 0
WWTP 7 | ek = 0 | R1k = 0.0 | Sk = 234.45 | R2k = 0.606
```

```
[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  0.000  0.644
      1     2  0.000  0.801
      2     3  0.276  0.970
      3     4  0.326  0.911
      4     5  0.000  0.587
```

```
5    6  0.217  0.603
6    7  0.000  0.606
```

```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
```

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)
```

```
[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  0.644    0.860    0.570                  29.000
      1        2  0.801    0.870    0.710                  16.000
      2        3  0.979    1.000    0.930                   7.000
      3        4  0.940    1.000    0.840                  16.000
      4        5  0.587    0.640    0.520                  12.000
      5        6  0.689    1.000    0.480                  52.000
      6        7  0.606    0.680    0.570                  11.000
      7     Mean  0.749    0.864    0.660                  20.429
      8       SD  0.148    0.141    0.159                  14.391
```

```
[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/6d/"
```

```
[NbConvertApp] Converting notebook clip-max-ignore-zeros-custom.ipynb to pdf
[NbConvertApp] Writing 44947 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 46789 bytes to ../../results/6c/clip-max-ignore-zeros-
custom.pdf
```