clip-max-ignore-zeros-custom

July 29, 2020

[2]:
```
# rank the obtained results using the *.log files
import os
import pandas as pd
import numpy as np
```

[3]:
```
source = "3c"
targetdir = '../../data/' + source + "/"
filelist = sorted(os.listdir(targetdir))
```

[4]:
```
filelist
```

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

[5]:
```
# 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()
```

[5]:

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	0.88	1.00	1.00	1.00	0.73	1.0	0.80
1	0.93	1.15	0.00	1.20	0.73	0.0	0.89
2	1.00	1.22	0.00	1.33	0.73	0.0	0.93
3	0.73	0.83	0.91	0.83	0.62	0.9	0.59
4	0.73	0.83	0.91	0.83	0.62	0.0	0.59

[6]:
```
# Clip values > 1 with 1 and ignore 0s
df.mask(df > 1, 1, inplace=True)
df.mask(df == 0, np.NaN, inplace=True)
```

[7]:
```
# Count NaN values
df.isna().sum()
```

[7]:
```
1.data      0
2.data      0
3.data    324
4.data    288
5.data      0
6.data    261
7.data      0
dtype: int64
```

[8]:
```
# Ignore invalid values by dropping them from the dataframe
df = df.dropna()
```

[9]:
```
df
```

[9]:

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	0.88	1.00	1.00	1.00	0.73	1.00	0.80
3	0.73	0.83	0.91	0.83	0.62	0.90	0.59
6	0.62	0.70	0.81	0.71	0.52	0.49	0.50
7	0.62	0.70	0.82	0.71	0.52	0.49	0.50
8	0.62	0.70	0.82	0.71	0.52	0.49	0.50
..	…	…	…	…	…	…	…
718	0.68	0.73	0.97	0.98	0.53	0.49	0.51
719	0.68	0.73	0.97	0.98	0.53	0.49	0.51
726	0.69	0.74	0.99	1.00	0.53	0.50	0.52
727	0.69	0.74	0.99	1.00	0.53	0.50	0.52
728	0.69	0.74	1.00	1.00	0.53	0.50	0.52

[378 rows x 7 columns]

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

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

[11]:
```
378
```

[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()

    # Calculating R1k ek/tao
    R1k = round(ek/tao,3)

    # Calculate R2k
    if tao != ek:
        R2k = (Sk - ek)/(tao - ek)
    elif Rk1 == 1:
        R2k = 0

    R2k = round(R2k,3)

    # Printing results
    print("WWTP", wwtp,"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 = 0.713 Maximum = 0.93 Minimum = 0.61 Amplitude (max-min)(%) = 32.0
WWTP 1 R1k = 0.0 | Sk = 269.67 | R2k = 0.713
WWTP 2 Mean = 0.793 Maximum = 1.0 Minimum = 0.69 Amplitude (max-min)(%) = 31.0
WWTP 2 R1k = 0.095 | Sk = 299.85 | R2k = 0.771
WWTP 3 Mean = 0.92 Maximum = 1.0 Minimum = 0.76 Amplitude (max-min)(%) = 24.0
WWTP 3 R1k = 0.27 | Sk = 347.67 | R2k = 0.89
WWTP 4 Mean = 0.878 Maximum = 1.0 Minimum = 0.68 Amplitude (max-min)(%) = 32.0
WWTP 4 R1k = 0.286 | Sk = 331.74 | R2k = 0.829
WWTP 5 Mean = 0.583 Maximum = 0.76 Minimum = 0.5 Amplitude (max-min)(%) = 26.0
WWTP 5 R1k = 0.0 | Sk = 220.23 | R2k = 0.583
WWTP 6 Mean = 0.653 Maximum = 1.0 Minimum = 0.48 Amplitude (max-min)(%) = 52.0
WWTP 6 R1k = 0.095 | Sk = 246.96 | R2k = 0.617
WWTP 7 Mean = 0.573 Maximum = 0.86 Minimum = 0.49 Amplitude (max-min)(%) = 37.0
WWTP 7 R1k = 0.0 | Sk = 216.6 | R2k = 0.573
```

[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	0.713
1	2	0.095	0.771
2	3	0.270	0.890
3	4	0.286	0.829
4	5	0.000	0.583
5	6	0.095	0.617
6	7	0.000	0.573

[15]:
```
import os

# define the name of the directory to be created
path = "../../results/" + source + "/"

try:
    os.mkdir(path)
except OSError:
    print ("Creation of the directory %s failed" % path)
else:
    print ("Successfully created the directory %s " % path)
```

```
Creation of the directory ../../results/3c/ failed
```

[16]:
```
# Save rankings dataframe as csv file
ranked_df.to_csv(path + "ranking.csv",index=False)
```

1 Calculate Descriptive Statistics

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

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

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

[20]:
```
# 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)
```

[21]:
```
# Reorder columns
stats_df = stats_df.reindex(columns=["WWTP", "Mean", "Maximum", "Minimum",␣
↪amp_str])
```

[22]:
```
stats_df
```

[22]:

	WWTP	Mean	Maximum	Minimum	Amplitude (max-min)(%)
0	1	0.713	0.930	0.610	32.000
1	2	0.793	1.000	0.690	31.000
2	3	0.920	1.000	0.760	24.000
3	4	0.878	1.000	0.680	32.000
4	5	0.583	0.760	0.500	26.000
5	6	0.653	1.000	0.480	52.000
6	7	0.573	0.860	0.490	37.000
7	Mean	0.730	0.936	0.601	33.429
8	SD	0.128	0.087	0.105	8.550

[23]:
```
# Save statistics dataframe as csv file
stats_df.to_csv(path + "statistics.csv",index=False)
```