default-statistics-and-rankings

July 13, 2020

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

[2]:
```
targetdir = '../data/processed/'
filelist = sorted(os.listdir(targetdir))
```

[3]:
```
filelist
```

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

[4]:
```
df = pd.DataFrame()
```

[5]:
```
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)
```

[6]:
```
df
```

[6]:

	1.data	2.data	3.data	4.data	5.data	6.data	7.data
0	1.00	1.00	1.00	1.00	0.73	1.00	1.00
1	1.06	1.16	0.00	0.00	0.73	0.00	0.00
2	1.12	1.22	0.00	0.00	0.73	0.00	0.00
3	0.99	0.83	0.91	0.83	0.63	0.91	1.00
4	0.99	0.83	0.91	0.83	0.63	0.00	0.00
..	…	…	…	…	…	…	…
724	1.06	0.87	0.00	0.00	0.65	0.88	1.01
725	1.11	0.87	0.00	0.00	0.65	0.00	0.00
726	1.00	0.74	1.00	1.00	0.54	0.50	1.00
727	1.00	0.74	1.00	1.00	0.54	0.50	1.00
728	1.00	0.74	1.00	1.00	0.54	0.50	1.00

[729 rows x 7 columns]

[7]:
```
# Creating ranked dataframe
ranked_df = pd.DataFrame()
stats_df = pd.DataFrame()
```

[8]:
```
# Creating scenario quantity variable
tao = len(df)
tao
```

[8]:
```
729
```

[9]:
```
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 = 1.028 Maximum = 1.4 Minimum = 0.0 Amplitude (max-min)(%) = 140.0
WWTP 1 R1k = 0.631 | Sk = 749.71 | R2k = 1.077
WWTP 2 Mean = 0.885 Maximum = 1.36 Minimum = 0.7 Amplitude (max-min)(%) = 66.0
WWTP 2 R1k = 0.306 | Sk = 645.09 | R2k = 0.834
WWTP 3 Mean = 0.514 Maximum = 1.06 Minimum = 0.0 Amplitude (max-min)(%) = 106.0
WWTP 3 R1k = 0.148 | Sk = 374.57 | R2k = 0.429
WWTP 4 Mean = 0.488 Maximum = 1.03 Minimum = 0.0 Amplitude (max-min)(%) = 103.0
WWTP 4 R1k = 0.148 | Sk = 355.5 | R2k = 0.399
WWTP 5 Mean = 0.64 Maximum = 0.79 Minimum = 0.51 Amplitude (max-min)(%) = 28.0
WWTP 5 R1k = 0.0 | Sk = 466.74 | R2k = 0.64
WWTP 6 Mean = 0.438 Maximum = 1.06 Minimum = 0.0 Amplitude (max-min)(%) = 106.0
WWTP 6 R1k = 0.084 | Sk = 318.99 | R2k = 0.386
WWTP 7 Mean = 0.558 Maximum = 1.06 Minimum = 0.0 Amplitude (max-min)(%) = 106.0
WWTP 7 R1k = 0.399 | Sk = 406.53 | R2k = 0.264
```

[10]:
```
# Reorder columns to be usable as a results table
ranked_df = ranked_df.reindex(columns=['WWTP',R1k_col, R2k_col])
```

[11]:
```
ranked_df
```

[11]:

	WWTP	R^1	R^2
0	1	0.631	1.077
1	2	0.306	0.834
2	3	0.148	0.429
3	4	0.148	0.399
4	5	0.000	0.640
5	6	0.084	0.386

```
6    7  0.399  0.264
```

[12]:
```
# Save rankings dataframe as csv file
ranked_df.to_csv("../results/default/ranking.csv",index=False)
```

[13]:
```
# 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)
```

[14]:
```
# 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)
```

[15]:
```
# 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)
```

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

[17]:
```
# Reorder columns
stats_df = stats_df.reindex(columns=["WWTP", "Mean", "Maximum", "Minimum",␣
↪amp_str])
```

[18]:
```
stats_df
```

[18]:

	WWTP	Mean	Maximum	Minimum	Amplitude (max-min)(%)
0	1	1.028	1.400	0.000	140.000
1	2	0.885	1.360	0.700	66.000
2	3	0.514	1.060	0.000	106.000
3	4	0.488	1.030	0.000	103.000
4	5	0.640	0.790	0.510	28.000
5	6	0.438	1.060	0.000	106.000
6	7	0.558	1.060	0.000	106.000
7	Mean	0.650	1.109	0.173	93.571
8	SD	0.206	0.194	0.278	33.312

[19]:
```
# Save statistics dataframe as csv file
stats_df.to_csv("../results/default/statistics.csv",index=False)
```