clip-max-ignore-zeros-asymmetric

August 13, 2020

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

[2]:
```
source = "6a"
targetdir = '../../data/' + source + "/"
filelist = sorted(os.listdir(targetdir))
```

```
␣
↪---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call␣
↪last)

<ipython-input-2-dbdedf4be382> in <module>
  1 source = "3a"
  2 targetdir = '../../data/' + source + "/"
----> 3 filelist = sorted(os.listdir(targetdir))


FileNotFoundError: [Errno 2] No such file or directory: '../../data/3a/'
```

[ ]:
```
filelist
```

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

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

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

[ ]:
```
df
```

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

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

[ ]:
```
# going through every column
for column in df:
    wwtp = column[0]

    # In every column, drop na values
    asym_column = df[column].dropna()

    # and calculate individual tao
    tao = len(asym_column)

    # calculate mean
    avg_eff = round(asym_column.mean(),3)

    # calculate max
    max_eff = round(asym_column.max(),3)

    # calculate min
    min_eff = round(asym_column.min(),3)

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

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

    # print stats results
    print("WWTP",wwtp,
          "\nMean =",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)

    # Calculating Sk sum of factors
    Sk = round(asym_column.sum(),3)

    # Calculating ek sum of factors of 1 (or above if errors in calculation)
    ek = asym_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 R1k == 1:
        R2k = 0

    R2k = round(R2k,3)

    # Printing results
    print("tao =",tao,"| ek =",ek,"| R1k =",R1k, "| Sk =",Sk, "| R2k␣
↪=",R2k,"\n")

    # 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)
```

0.1 Ranking of WWTP

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

[ ]:
```
ranked_df
```

[ ]:
```
import os

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

# 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)
```

0.2 Calculate Descriptive Statistics

[ ]:
```
# 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)
```

[ ]:
```
# 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)
```

[ ]:
```
# 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)
```

[ ]:
```
# 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)
```

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

[ ]:
```
stats_df
```

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

[ ]:
```
# Convert Jupyter Notebook to PDF LaTeX file
```

```
!jupyter-nbconvert --to pdf "clip-max-ignore-zeros-asymmetric" --output-dir "../
↪../results/6a/asymmetric"
```

[ ]: