Construction sites SonVis Algorithm documentation

Svoronos - Kanavas Iason

June 2, 2022

Contents

1 Algorithm architecture

The overall algorithm is structured as:

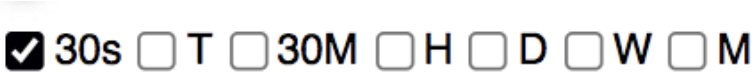
1. Interface
2. Sub-processes
3. data processing algorithm
   (a) outlier identification,
   (b) replacement,
   (c) dataframe creation and manipulation,
   (d) min-max extraction script
   (e) datetime re-sample function
4. on-run functions (includes matrix printing functions)
5. IPC inter-process communication (& connection setup functions)
6. Sonification algorithm
   (a) processing functions
   (b) data receiver – handler
   (c) synth and data parameter dictionary
   (d) event receiver - actions
   (e) IPC connection configuration
   (f) synthesisers
   (g) map to scale frequency mapping patch

2 Interface

The interface consists of certain visual elements. The datetime range slider and text input (for the time) are used for date and time accordingly. Using these the user can specify a desired datetime period in the data to sonify and visualise.

The **start** button is used so that the datetime period is extracted and then sent to be sonified and visualised, while the **killall** button can interrupt/stop every running procedure in relation to the sonification and visualisation process anytime.

Below, there is a checkbox where the user can select the period of the values in the data to re-sample.



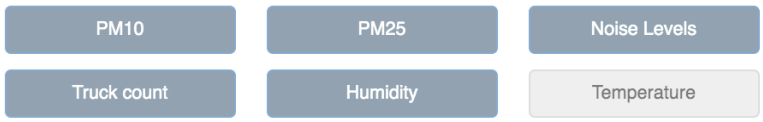
The original collection is every 30 seconds. The re-sample options are, per minute (T), per 30 minutes (30M), per hour (H), per week (W), per month (M). In the re-sample function (see below), the data values are derived in every period with selecting the max value when doing the frequency conversion to highlight peaks in the data. For example, if we want to re-sample with 1 minute re-sample frequency (T). We have initially:

timestamp,	data
2021-08-01 00:00:00,	2
2021-08-01 00:00:30,	0
2021-08-01 00:01:00,	4

Re-sample result: The extracted value for the first minute will be "4" because is the max value observed for this minute.

Then, there is a another slider that controls the data iteration frequency and it has a range from 1 to 1000 values per second.

Finally, there are six buttons where the user can use to turn on/off, the desired data parameters to sonify - visualise



Overall, the button python bokeh elements, trigger osc messages that are sent from python to supercollider in order to control different synths. The ones that utilise this functionality is the **start killall** and the synth on/off buttons (pm10, pm25, temp, humidity, noise levels, truck count). This will be elaborated in the IPC section

3 Sub-processes

On launch, sclang is initialised and runs as a sub-process within the python session. More specifically, the SuperCollider patch for sonification is evaluated using the following command in Python.

```
# run sonification patch
sclang = subprocess.Popen(
    'sclang particleSonification.scd', shell=True,
    stdout=subprocess.PIPE,
    stderr=subprocess.STDOUT)
```

Getting back now to the initialisation python script where a function obtains the IP address of the computer using a shell command and then stores it as a global variable. After that, the OSC client configuration setup uses the variable's value ($\text{udp}_{\text{client}}$ object). The function is defined the following way as well as the OSC setup. This process easily configures OSC intercommunication between python and SuperCollider therefore mistakes and hassle by hard-coding IP addresses or manual configurations are avoided.

```
# get IP address
def getip():
    global ip
    ip = subprocess.Popen(
        'ipconfig getifaddr en0', shell=True,
```

```
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT)
    ip, _ = ip.communicate()
    ip = ip.decode('utf-8')
    ip = ip.strip()
    print(ip)

# Python osc
getip() # run getip function
client = udp_client.SimpleUDPClient(ip, 57120)
```

Note: *this works* ***only for macOs****. Therefore it has to be adjusted for linux or windows.*

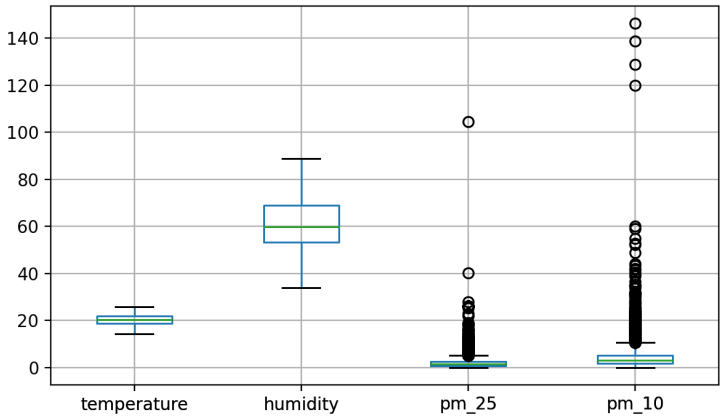
WIN hint:

```
ipconfig | grep IPv4 Address.
```

4 data processing

In this section the data processing will be described. The algorithm is developed in Python. The idea is based on combining and re-constructing the datasets after the processing results that come out from the derived stats (IQR). SC has also access to the derived dataset (it is written to disk) so that it has access to the min max values for the correct mapping (see 4.4). In this way, it is also possible to re-use the algorithm with different data since the mapping is not hard-coded.

Outlier identification and replacement was deemed necessary since it was observed by using box-plot stats the PM (both 25 and 10) showed extreme values (far from accurate measurements (140~ PM10)) that we would like to exclude.


140
120
100
80
60
40
20
0
temperature
humidity
pm_25
pm_10