QAQC PIPELINE NOTES

Written by Julien Bodart on 2024-08-25, contact julien.bodart@unibe.ch

General rationale

The aim of the QAQC pipeline is to remove as many offsets, outliers, and weird jumps as possible, without which the data often looks difficult to interpret, in the most automated way possible. The process is not perfect, and might require light labor occasionally, but because it is automated, it should relieve a fair amount of manual work previously done by one or two people. It's likely an evolving process, with the potential to improve more as new issues are encountered and specific problems need to be tackled. Note that some issues are hard to automate (e.g. see (4) below or the last section in the README.md file on GitHub). For these, it is recommended to make changes manually if possible, or dedicate time to develop something automatic that will be robust enough to handle different scenarios.

The codes are **written in Python and available on GitHub**. For each value that is QAQCed, there is a **flag number** associated with it which indicates why a particular data point was (or was not) QAQCed. Flag "0" indicates no QAQC took place on a specific value, whereas flags >0 refer to a particular QAQC process or step (**see README.md file** for more information on each of these steps, or in the codes directly – see (5) below).

The codes are **run automatically every Sunday at 1am (UTC-7 Vancouver time) to** QAQC **last week's of data**. It takes **on average 1 hour** for the codes to QAQC one week of data and push these to the database. One could also only run them twice a month (so as to QAQC only the last two weeks of data instead of every week), but processing and pushing more than 2 weeks of data might lead to memory issues on server and will kill the process. Likewise, each step will take longer to run, so this must be accounted for by changing the crontab file to avoid having multiple codes run at the same time and overload the server memory. (see description of steps above for that).

Workflow to make changes to the codes

If you need to make changes to the QAQC codes, this is the usual workflow:

- fork the CHRL QAQC GitHub repo https://github.com/CHRL-VIU/QAQC_VIU_wx on your own GitHub account
- clone the forked GitHub repo onto your local machine
- request the "config.py" file from someone at VIU CHRL so local machine can access Island Hosting database (this file can also be found on the CHRL droplet under the folder /python-scripts/QAQC_VIU_wx/
- whitelist local machine IP address on Island Hosting https://galiano.islandhosting.com
- make changes to the Python codes, Python files or CSV files on your local machine
- push those changes onto your own forked copy of the GitHub repo
- pull the changes made on your own GitHub account to the CHRL-VIU GitHub account
- SSH on Digital Ocean remote server (chrl-3 droplet): ~# ssh root@137.184.164.151 and enter the pswd for your own private-public key, which should be stored in advance on the Digital Ocean

account (see: https://docs.digitalocean.com/products/droplets/how-to/add-ssh-keys/to-existing-droplet/)

- once you're on the server, cd to the directory where the remote GitHub project lives by running: ~# cd /python-scripts/QAQC_VIU_wx/
- pull the changes made to the QAQC repo from the CHRL-VIU GitHub account onto the remote server by running in the terminal: ~# sudo git pull origin
- to add, remove, or change the timing of the automated QAQC process on the server, simply run: ~# crontab -e. Careful: once crontab is amended the changes are lost. To simply view the time at which the QAQC process is, run: ~# crontab -l.

Explanation of the QAQC pipeline

The QAQC pipeline can be divided into **6 main steps**:

(1) **'qaqc_stations_list.py'**: This file contains the name of the weather stations and variables that are to be QAQCed. Any weather station or variable not in this list will not be QAQCed. The name of the stations and variables must match the name of the "clean" databases on Island Hosting. Simply add a variable or weather station if it is missing from the list or comment out a weather station or variable which you wish **not** to be QAQCed for a specific reason (e.g. the station is offline and slowing down or killing the QAQC workflow). This file is akin to the tbl_defs.php code on the CHRL-VIU GitHub "GOES" repo.

(2) **'qaqc_functions.py'**: This file contains all the functions designed to QAQC all weather stations and variables. Any changes to the actual process through which the functions operate and QAQCes the data should be made there. For example, if you find that the moving window code is not working as well as you like, or you notice a mistake in a specific function which affects how it QAQCes a particular variable, you can make modifications there. You can also add additional functions which you create yourself to address specific problems (see for example limitations highlighted in (4) below).

(3) **'push_sql_function.py'**: This file contains support functions that are called in the codes to push the QAQCed data to the "qaqc_*" databases. It works well with the new and older Python versions (3.2, 3.6. and 3.12) and SQLAlchemy version 2.0 and below. If you notice errors in the log files (see "Useful information" section below) which describe issues with regards to pushing or writing data to the databases, this is likely the culprit (or also check 'fill_db_nans_weekly.py' code below).

(4) **CSV files**: There are three CSV files (**'sdepth_zeroing_dates.csv'**, **'SWE_zeroing_dates.csv'**, **'PrecipPipeRaw_drain.csv'**). These files are called inside the 'qaqc_functions.py' and in the respective codes for each of these three variables to aid the QAQC process.

Right now, these files need to be updated manually a few times a year by entering the date and time of a particular event. The first two CSV files only need to be updated at the start of the summer of every year (usually June) by "eyeballing" the approximate date and time when snow depth and SWE for each weather station has melted out (implying there is no more snow on the ground) – this enables the code to know when to zero out all values in the summer. At times

(particularly for SWE), the algorithm can detect pretty accurately this transition and there is no need to enter this date and time manually. The third CSV file needs to be updated throughout the year by adding the date and time at which the precipitation pipe was drained and recovered again.

Ideally, a more technical solution could be found (see attempts commented in the 'qaqc_function.py' file), however these are hard to automate due to noise in the underlying data or weird offsets/jumps. The manual task of entering those values should take less than a few minutes, so it is not a terrible burden in the meantime.

(5) '***_QAQC.py**': There are 11 Python codes, each of which belong to a specific variable (e.g. AirTemp_QAQC.py, etc.). These contain the specific steps, thresholds, window lengths, and any other things associated with the QAQC of a particular variable. If you find that the algorithm appears too strict for specific variables and that it deletes good data, you can open one of those files and modify the thresholds, window lengths, etc. to adapt the process. Note that for the Tipping Bucket Cummulative ('PC_tipper') variable, the data is not QAQCed but rather the code 'PC_Tipper_recalculated.py' takes the previously QAQCed data obtained from running 'PP_Tipper_QAQC.py' and re-calculates the yearly cumulative values from the QAQCed data directly. If you change the thresholds, make sure you write the date when this change was made.

Also note that in some of the codes, there are individual offsets which are manually removed (or nan-ed out) because sometimes the QAQC *does not pick them up. There is also an offset to the Snow Depth and SWE values which is applied in the codes for specific weather stations, but that may change if changes are made in the field with regards to sensor or sensor height. It is important to check at the start of each water year whether the offset in the* QAQC *codes for the SnowDepth and SWE are correct, otherwise the* QAQC *values will be offset by a wrong value for the whole year.*

(6) 'fill_db_nans_weekly.py': This file reads the "clean" databases on Island Hosting from the 'qaqc_stations_list.py', calculates today's dates minus one week, and appends additional empty rows to each weather station's "qaqc_*" databases. This is so that these new rows can be filled by the codes in (5). **For this reason, this code must always be run <u>first</u> before any of the other QAQC codes in the pipeline.**

Useful information

- There is a **log file** which contains **the output (or print) of the Python codes**. This file is available at /python-scripts/QAQC_logfile.txt. Note that this file is wiped every time the codes are run again in crontab. If any errors are returned or the QAQC data for the previous week is not available on the plots on the viuhydromet website, then you can find the error source there and fix it.

- The codes are ran using a **virtual Python environment on the server** (called '**qaqc_venv**'). This is to avoid messing with the local Python environment on the server. The file 'run_venv_py.sh' is called before each cron job is triggered to tell the server to run the codes through this virtual environment. If you don't use this environment, the codes won't work as they need specific Python libraries only found in this virtual environment. To run a specific QAQC Python file (e.g.

'AirTemp_QAQC.py') directly in the terminal without the use of crontab, one can run: ~# /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/AirTemp_QAQC.py.

- Careful: Don't modify codes directly on the remote server using vim. Since they are synced with the GitHub repo of CHRL-VIU, it will lead to conflicts with branches. You can do so temporarily if you want, and either fix the merging of the branches, or remove the folder entirely and re-pull it from the GitHub repo onto the server once you're done with your tests.

- To add new libraries to the virtual Python environment, cd into the virtual environment first, then run: ~# source bin/activate, then run: ~# pip install *package-name-of-your-choice*.

- Avoid running the QAQC process on chrl-1 droplet (aka alex@138.197.171.225) or chrl-2 droplet (aka root@143.198.38.84) as it gets killed instantly due to memory issues and other processes happening at the same time on those two droplets. chrl-3 droplet (aka root@137.184.164.151) has more memory (2GB instead of 1GB for the other two droplets) and has no large processes taking place on it, meaning the QAQC process does not get killed instantly. **For this reason, avoid using chrl-3 droplet for large pulling/pushing of data on the date and time the cronjobs are running!**

- The crontab jobs (see at the end of this document) where spaced out enough to avoid that a code that has not yet finished running conflicts with a new job starting. During test mode, each job could complete within approximately 7 minutes, but for safety, each job has been spaced out by 15 minutes. If issues arise and show up in the log file as "killed" or "could not push to sql db", then it is likely that the jobs were conflicting with each other as it took longer for a specific code to run, which results in overloaded memory and the server to kill the jobs. This can happen if other codes or apps are installed on the server and eating a lot of RAM, or if pushing more than 1 week of data. **If this is the case, try to space out the cron jobs by more than 15 minutes.**

- A useful way to update the database beyond just last week and without using the automated cron job is to run the codes locally on your own machine. This is useful if you want to loosen a threshold for a particular QAQC step and want that change to happen for all of the data in the past (i.e for all previous water years, for example). To do that, you can simply un-comment the lines of codes in the '*_QAQC.py' files under the header "*#%% write data to sql database using brute approach (re-write whole db - quicker on laptop but gets instantly killed on remote desktop)*" and comment the lines under the header "*#%% write data to sql database using soft approach (re-write only idx and vars needed - very slow on* laptop *but fast on remote desktop)*". **Careful though, this will replace all the existing QAQC data for the new one you're running, so you will lose all the data already on the dbs.** Best to save the database locally first in case you make a mistake.

Record of crontab jobs on remote server (as of 2024-08-25)

0 8 * * 7 /python-scripts/QAQC_logfile.txt && echo "/// Starting new qaqc for last week at $(date)" >> /python-scripts/QAQC_logfile.txt 2>&1 && /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/fill_db_nans_weekly.py >> /python-scripts/QAQC_logfile.txt 2>&1

15 8 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/AirTemp_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

30 8 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/BP_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

45 8 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/WindSpeed_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

0 9 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/PkWindSpeed_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

15 9 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/WindDir_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

30 9 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/PkWindDir_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

45 9 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/PP_Tipper_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

0 10 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/Precip_pipe_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

15 10 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/SWE_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

30 10 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/SnowDepth_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1

45 10 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/RH_QAQC.py >> /python-scripts/QAQC_logfile.txt 2>&1 # depends on AirTemp_QAQC so allow time to push dependent data

0 11 * * 7 /python-scripts/QAQC_VIU_wx/run_venv_py.sh /python-scripts/QAQC_VIU_wx/PC_Tipper_recalculated.py >> /python-scripts/QAQC_logfile.txt 2>&1 # depends on PP_Tipper_QAQC.py so allow time to push dependent data