Pyquan 1.0

Content

1. Introduction

PYQUAN is software developed to automate the analyse of GC/MS data. The purpose is to automate the most labour intensive parts of the analyse, while keeping control over the whole process. This is done by creating images in the alignment, quantification and normalization stage. My philosophy is that even the most sophisticated automatic quality control cannot beat an expert view on visual presented data.

PYQUAN is designed around the freely available AMDIS package (http://chemdata.nist.gov/dokuwiki/doku.php?id=chemdata:amdis). AMDIS is used to identify the peaks, while PYQUAN (1)checks the identification based on a sophisticated alignment algorithm and a reference library, (2) quantify the checked peaks, and (3) normalize the peaks to either the total ion current or an internal standard. If selected, peaks missed by AMDIS (false negatives) can be included in the quantification.

Please refer to Smits et al. (xxxx) when using this software for scientific purposes. Feel free to contact the builder (Mark Smits: lycopoda@gmail.com) for any questions or remarks.

2. Setting up the software

2.1 Software installation

Install the AMDIS software (http://chemdata.nist.gov/mass-spc/amdis/downloads/). If you don't have already python 2.7 on your computer, I recommend to install Anaconda (https://store.continuum.io/cshop/anaconda/). Anaconda is a Python distribution that include the right packages to run PYQUAN, except netCDF4. So, if you don't have already netCDF4, install it now. You can find the right version here: https://code.google.com/p/netcdf4-python/downloads/list.

2.2 AMDIS library

Next step is to create or move your AMDIS library in .msl format to the library folder. With AMDIS you can manage your library. From e.g. the NIST library you can import new spectra into the library. NB each compound should have a code that could be recognized by PYQUAN. In AMDIS, edit the compound information, and start the name with your code, followed by '**:**'. For example:

004a: 1-Propene, 2-methoxy-

The code itself can have any character except **?** and **:**.

2.3 Reference library

Now, you create your reference library. It is a csv file with, per row:

Code, RT, limit, mass, name, source.

Use the same code as in the AMDIS library. You can create and edit your library in e.g. Excel. Don't worry if Excel removes the first zero's from a number. PYQUAN can still link those codes to the right AMDIS code.

The reference RT values should have a linear relationship with the measured RT values in your sample. Kovat's or Lee's retention index are not linear! If you use different GC programs, you should create a reference library for each program. In the pyquan.ini file you can indicate which reference library you use for your current analysis.

Limit denotes the valid RT range for that compound. Slight variation in machine settings or the condition of the GC column can cause deviations from the expected RT. In the actual analyse, after the alignment procedure, the expected RT's are fine-tuned, and the valid RT range is reduced, as RT variation is much less within one run, than between runs.

In the mass column, you give for each compound the mass that is used for quantification. Up to now you can select only a single mass, but I am planning to allow multiple mass quantification in the next version of PYQUAN. Based on the total spectrum, single mass area will be calculated back to total ion current for each compound.

Give the name for each compound. If your name contains comma's, excel automatically place "" around the name (although they are not visible in Excel). If you create your library in a simple text editor, you have to do it manually.

Optionally, you can give the likely source of the compound.

2.4. Data structure

<u>Code:</u>

Contains the python codes.

<u>Library</u>:

Your reference library and AMDIS library should be stored here.

<u>Init</u>:

The initialization files can be found here.

<u>Data</u>:

Data files in CDF format are stored here.

<u>Projects</u>:

Here the runlist files and the AMDIS batch jobs are stored. During the pyquan run, a project folder will be created with the Projects folder:

> 'Project_name':
>
> This folder get the same name as the runlist file. It will contain the output data and a project library. It also contains the following folders:
>
> > <u>Alignment</u>:
> >
> > This folder contain images of the aligned process, for visual control. It also contains for each compound RT, RT converted to the reference scale , and the fit of the AMDIS identification.

Amdis:
Contains the AMDIS results for each sample in a separated file
Data:
Contains the fitting details of each peak.
Images:
Contains an image for each peak, for visual inspection.
Normalization:
Contains TIC images, together with the calculated baseline and all quantified peaks, for a quick visual control of missing peaks or wrong identifications.

3. Start project

1. Create runlist
2. Run a batchjob in AMDIS
3. Adjust pyquan.ini file
4. Run project in pyquan

3.1 Create a runlist file

The project runlist is a csv file (e.g. created in excel), containing one column with all sample names in the project. NB: sample names don't need to have the .CDF extension in the runlist file. The file is stored as 'project name'.csv in the projects folder.

3.2 Run a batchjob in AMDIS

Open AMDIS.
Create a batch job for all samples in your project.

- Set analyse type to 'simple'.
- Select 'Generate report', and 'Report all hits'
- Save batch job as *'project name'_AMDIS.job*

Run batchjob.

3.3 Adjust pyquan.ini file

Pyquan.ini can be opened in any text editor. Here you can change fitting parameters, reference and AMDIS library, and parameter values for alignment, normalization and peak fitting. You always can restore the default pyquan.ini file with running:
default.py pyquan
default.py check_peak

3.4 Run project in pyquan

Pyquan can be run from the (windows) command line. Ideally you create a shortcut that opens a command line in the right folder (…/pyquan/code).

The software is divided in three steps:

1. Calibrate: *calibrate.py*
2. Quantify peaks: *pyquan.py*
3. Normalize: *normalize.py*

All steps can be run at once with *analyze.py*

In the python platform you start your analyse by writing on the command line:
python analyze.py 'project name' (or *python calibrate.py 'project name'* for running only the calibration step).

4. Analyse results

Each step in the analyse stores a series of images for visual control.

4.1 Calibration:

The calibration step creates a series of images in the alignment folder (Figure 1). The RT, normalize RT and identification fit for each compound are stored in CSV spreadsheets.

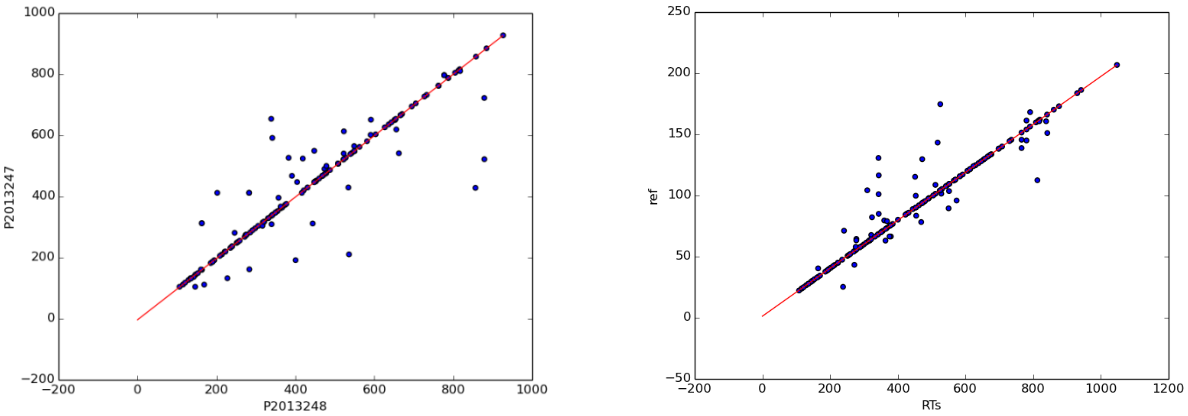

Figure 1: Alignment images. Left: he dots are compounds, identified in both samples by AMDIS. The red line is the best fit through most data points. It represents the alignment factor between both samples. Right: Here dots are compound with on the x-axis the median of the RT in the aligned samples, and on the y-axis the RT in the reference library. The red line represents the correction factor with the reference library.

4.2 Quantification:

At the quantification stage, each identified peak, that fits within the limits of the expected RT (based on the reference library) is quantified. Quantification is based on a the fit of exponential-Gaussian hybrid function. If the fitting fails, the actual surface of the peak is measured. Each quantified peak is stored as image in the image folder (Figure 2).

The naming of the images (code_sample.png) allow sorting of the images per compound code, and makes it easy scan the quantified peaks code by code.

4.3 Peak correction

If any peak is not correctly quantified, the fit or estimated RT can be adjusted with:

> *python check_peak.py*

Controlled by a init file (*check_peak.ini*) it is straightforward to vary e.g. fitting or smoothing parameters. This can be done on compounds in individual samples, or on all samples of the project at once. You can also change RT, or skip the curve fitting and just measure the actual area of the peak.

Peaks can be removed with:

python remove.py 'project name', sample, code

If you fill in 'all' instead of the sample name, it will remove the compound from all samples in the project.

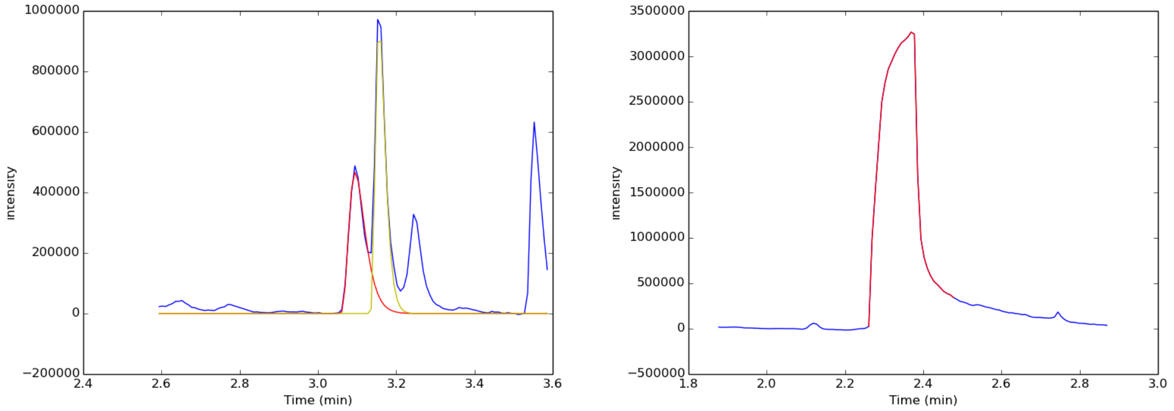

Figure 2: Left: peak quantified with an exponential-Gaussian hybrid function. In red the peak of interested, in yellow adjacent convoluting peaks. Right: a peak where fitting a curve didn't work, area is measured directly.

5. Normalization

The normalization step can be run separately via:

python normalize.py 'project name'

Peak areas can be normalized to the sum of TIC's over the whole time range minus the baseline, or normalized to any compound you like (e.g. an internal standard). The sum of all quantified peaks can be found, superimposed on the TIC, in an overview image (sample_complete.png) and in 1/10 time slides as well. These images show which peaks are not quantified (Fig. 3).

After the normalization step, output files can be found in the project folder. Projectname_area_norm.csv contains the normalized area data. If you are interested in the non-normalized area data: they can be found in the project data folder ('*project name'/data*). For the AMDIS identification fit, RT time, and RT converted to the reference scale, datasheets are created as well ('*project name'_fit.csv*; '*project name'_RT.csv*; '*project name'_RT_ref.csv*).

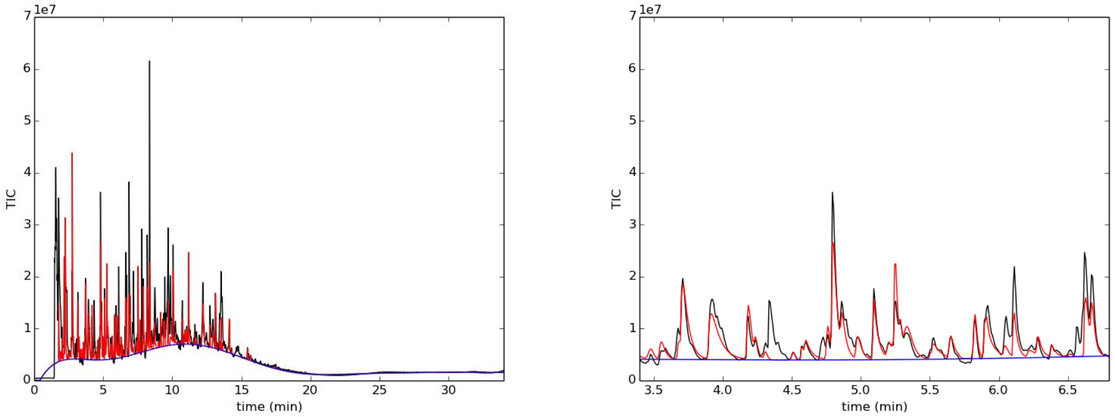

Figure 3: Normalization images. Total ion current in black, quantified peaks in red, and baseline in blue. Left: complete time range. Right: 1/10 time range.