Hi Clara,

I believe I’m at last done with that RASCALL issue-finding code. Here’s a short report/manual on how it works, what are its potential issues, and what tentative conclusions I could draw from my use of it.

First things, first, how it works:

* The prerequisite modules, on top of those already required for RASCALL itself, are lmfit and scipy, both can just be installed with “pip install…”.
* Place all the .py files inside the “rascall” folder with all the other RASCALL modules, except for executable\_issues\_finder.py, which you’ll place in the master directory, where rascall\_plot and rascall\_list are. I also recommend commenting out the print statement of line 7 of functional\_parser.py, or else it will be everywhere on your terminal as this code runs – I’m including a version with the line already commented out.
* Run from the terminal “python exectuable\_issues\_finder.py”. When running it, you have the following options, all of which are governed by terminal prompts:
  + Setting a minimum number of molecules for which we have NIST spectra in which a functional group needs to be present for it to be considered by the above algorithm – this is because the code above only has a point when running on large numbers of molecules, where the imprecisions in fitting etc can average out.
  + Setting a maximum number of molecules. This is useful to limit the running time of the code: as it iterates functional group by functional group, molecule by molecule, and is already quite slow in itself, if at some point it encounters a functional group which is present in say 50 molecules, that can hugely slow it down.
  + Whether to overwrite previous results (I would advise against this because you’d be uselessly repeating lengthy calculations)
* The core points of the workings of the code are the following:
  + Once a list of functional groups to iterate through is established, it iterates through each functional in that list. Let us say it begins with a functional group called F, which is present in N molecules for which we have NIST spectra.
  + It looks at the RASCALL-predicted features of F, and splits them up in groups of nearby features, which it will iterate through. Let us say it starts with the α feature group of F, which spans a certain wavenumber range.
  + For molecule 1 of the N to be considered, it takes its NIST spectrum, and attempts to fit it as a superposition of a variable number of Voigt profiles on a constant background. To pick the correct (“best-fit”) number of Voigt profiles, it uses the fit convergence criterion which we already discussed – say we attempt fitting for peaks with , with a corresponding sum of square residuals of . Then:
    - We define a “relative improvement” measure for the -peaks fit given by:

Notice this means that a fit that is worse in absolute terms than the preceding one will have

* + - If occurs, then the fit with peaks is selected; otherwise, more Voigt profiles are added until the condition is satisfied, up to a given maximum (which is anyway hardly ever reached).
  + For the chosen fit, the position and amplitude (meant as area) of each Voigt profile is stored. Once the above process is completed for all N molecules for the α feature group of F:
    - The spectral region is subdivided into a number of bins determined by how wide it is and by where it is (a region at higher wavenumbers will have wider bins)
    - For each bin, the number V of Voigt profiles (found through the fitting of the N spectra) which are centred within that bin is found. Each of these Voigt profiles will have amplitude , and then the total “cumulative amplitude” for each bin is found as

The multiplication by V is there so that one very big Voigt profile counts less (or at least not disproportionately more) than many smaller Voigt profiles.

* + - Two further measures of “cumulative amplitude” are taken, to account for possible contamination by other functional groups. A first one, , is calculated just identically to A above; however, it excludes in all calculations the molecules for which RASCALL predicts functional groups other than F will have features within the range spanned by the bin (hereafter “degenerate molecules”). I call this method “elimination”. A second one is given by , where N’ is the number of degenerate molecules. I call this method “multiplication”. As you can imagine, often these two measures yield nearly the same result.
    - The following data are then saved as follows:
      * If not pre-existing, a func\_group\_stats directory is created within the master RASCALL directory
      * Within it, a directory named as the functional group F in question (up to some special characters which needed to be replaced, as they weren’t accepted as part of directory names) is created
      * Within it, a txt file (with filename given by the minimum and maximum wavenumbers spanned by α – e.g., if it spans the region between 300 and 500 wn, the file would be called 300-500.txt) is created, containing, for each bin (a row in the file) the following columns:
        + Wavenumber at which the bin is centred
        + Values of A, A’’, A’
        + Value of V and value of V as found when only considering non-degenerate molecules
      * Within the RASCALL master directory, a directory named all\_plots is created, with three subdirectories: “original”, “elimination”, “multiplication”. In each is saved the plot of the histograms plotted using respectively A, A’ and A’’. In our example, for functional F in the range 300-500wn, these files would be named respectively “F\_orig\_300-500.jpg”, “F\_elim\_300-500.jpg” and “F\_mult\_300-500.jpg”
  + The above is repeated for all feature groups of each functional group, and for each functional group in the list.

Now, onto a couple of more points on what caveats I see with using this:

* First, this is ***slow***. It can easily take several hours for it to work through a relatively short list of functional groups (for reference, it took about 3h 30m to iterate through 18 functional groups present in at least 10 and no more than 25 molecules, if my memory serves me well). This can probably be solved by identifying and removing some redundancies which I’m sure will be there, as well as by finding/creating a better fitting package, because that’s where the bottleneck is.
* Second, while the fitting algorithm and best-fit selection are overall decent enough for me to be mostly confident in the overall result of the fitting of several molecules, where I would expect any mistakes in the fitting to average out, it is by no means good enough to be reliable on any individual spectrum. Initially, as you know, my fit convergence criterion was biased against picking the first fitting attempt (the one with the least superimposed Voigt profiles); indeed, it would not pick it unless the second attempt was worse *in absolute terms* than the first attempt. This was because the corresponding value of the relative improvement was set to 0, so that . By instead setting, as I did, as I did, I somewhat counterbalanced against this bias, by only skipping over this first fit if the second one is at least 10% better than the first one. This is an entirely arbitrary number can be adjusted as you see fit (pun intended).

Lastly, my conclusion from the calculations I ran with this is that, unfortunately, **assuming my code is reliable enough,** individual functional groups do not seem to have a well-defined enough set of own spectral features which repeat in all molecules they’re present in for the RASCALL approach to achieve the degree of accuracy needed in, for example, cross-sections for atmospheric retrievals on exoplanets. Maybe someone else might be able to find a fix to this problem.