RDGEN 9.5

February 11, 2009

Contents

1 Introduction

RDGEN started out as a catch-all program for handling spectroscopic data, containing miscellaneous routines which shared common input/output/visualization needs. It has evolved into a front-end/back-end/sidewalls for VPFIT, and a means for looking at combined VPFIT results. Pretty well everything it does is VPFIT-compatible.
It has several built-in functions, driven by two-letter commands.

2 Startup

Some environment variables should be set before starting RDGEN. These are

- ATOMDIR: The atomic data file, as for VPFIT
- RD_PRSETUP: This points to the file containing the list of commands. This is normally called rd_prhelp.dat in the VPFIT/RDGEN source directory, and so might be

  RD_PRSETUP=/(wherever)/vpfit9.5/rd_prhelp.dat.

  If it is absent, then typing 'he' or '?' to get a command list does not give you anything.
- RDSTART: This file contains a list of startup commands which might be used to preload various files for the interactive velocity plot package, or just to list the commands.
- VPFSETUP: Some of the VPFIT setup parameters are used, so it is worth including this.
- VPFPLOTS: This just resets plot parameters at each plot, and may be omitted.

It is possibly worth setting these up when you login, and also setting aliases for VPFIT and RDGEN, so you can forget about them.

3 Spectral data

3.1 Data reading

Just `rd filename` and FITS, IRAF or ASCII data will be read in. It tries them in that order, so if you omit file extensions and use `rd something`, then if both exist something.fits will be read and something.imh (and something.pix) will not.
The data read expects to find a spectral data file, which may contain wavelength information. It will also look for a file containing the error estimates, which is assumed to be 'filename'.sig.fits (if the data is in 'filename'.fits, where 'filename' can be anything), estimates for the RMS data fluctuations (not necessarily the same thing for rebinned data) in 'filename'.rms.fits, and a continuum estimate in 'filename'.cont.fits. These can be over-ridden by using FITS header items SIGFILE, RMSFILE and CONTFILE with

(string) values giving appropriate file names in 'filename'.fits. If a wavelength table is used , then it can be a separate file with associated with WAVFILE. Each of these is held in separate, but associated, arrays. 'CONTFILE unity' has exactly the effect you might expect - it gives unit continuum without reading a file. If there is a continuum file called 'unity.fits' then it is always ignored.

You may have forgotten which data files are in the current directory. To get a list of '.fits' files just type `fq` (for file query) and you'll get a list. If there are no FITS files then it will try the old IRAF '.imh' out of sheer desperation.

ASCII files should have 3 or four columns, with each row having

```
wavelength flux error continuum
```

If the last column is absent, then unity is assumed. Any format, separated by any (> 0) number of spaces, is fine.

3.2 Data writing

`wr` will write the current data to a new file. You will be prompted for a filename if you do not give it on the command line. You can give the filename as 'file.fits' or just as 'file'. In the second case the '.fits' is added automatically. Note that it writes out ONLY the data, and not the error estimate or the continuum. If you want to write them as well you normally have to copy (cy) (see section 3.3) the data from the array in which it is held to the data array. If the file already exists, then the attempt to write it will fail and a warning message is given..

The data is written out in double precision. If for some reason you want a single precision file then

```
wr file real
```

will do it for you.

You can write out the error, continuum and fluctuation arrays as one-dimensional FITS files if you want to. For a file contining the errors use `wr file.sig errors` or `wr file.sig sigmas`, continuum `wr file.cont cont`, and fluctuations `wr file.rms fluct` or `wr file.rms rms`. In fact only the first two letters of the last parameter are checked for these, so you can abbreviate 'errors' to 'er' etc. Anything which is not a recognized option results in the writing of the data to a double precision FITS file. Note the filename extensions - these are the defaults for re-reading if pointers have not been set in the data header.

If the data is normalized to a unit continuum, then you can write out the data & errors to a FITS file in a form which is similar to the one used by UVES_popler (see http://www.ast.cam.ac.uk/∼mim/UVES_popler.html).

The file does not have all the data fields generated by UVES_popler, but has the subset which VPFIT uses. To write the data in this format use

```
wr filename UVES_popler
```

(or simply `UV` will do - only the first two letters are checked).

ASCII data
If you want an ASCII file of the data which may also be read by VPFIT then use

```
wt filename (all)
```

and filename then contains one line per pixel with

```
wavelength   data   error  continuum  expected_fluctuations
```

– where the last column appears only if the optional parameter 'all' is given.

3.3 Moving data around

`cy` gives

```
Copy data array
Enter two letter code in order (from)(to)
continuum=c, scrunched (summed) array=s, workspace=w
e.g. cw for cont. to work (help for more info.)
```

The internal arrays are workspace (w), error (e), continuum (c), scrunched (linearized summed) data (s) and its error estimate (t).
`ws` copies data and error to scrunched data and error
`sw` copies scrunched data and error to data and error
You can suppress the copying of the error arrays by appending an "n" to the above two commands– swn & wsn copy data but not errors.
`ew` is error to workspace - so only one array is copied
`we` is workspace to error
`cw` is continuum to workspace
`wc` is workspace to continuum
A few specialist options are also available. Type `help` (after `cy`) to see what they are.

4 Artificial spectra

4.1 Putting in absorption lines

Generating spectra from a list of ions, redshifts, Doppler parameters and column densities can be done using the command

```
gp
```

and the required information is then prompted for, as

```
ion,col,bval,zed?    ..  or ...
<filename> (fmt,clo,chi,zlo,zhi,blo,bhi,binc,type)
              [26,everything,everywhere]
```

As this suggests, you can enter values interactively, as

```
H I 13.5 25 1.77890
HI 13.3  30.0   1.77945
```

terminating on a blank line (it tells you this). When this is done you still need to convolve with the instrument profile, which is normally assumed to be a Gaussian, so unless you have set up a resolution file with a FITS header pointer (see VPFIT documentation, the section on Spectral resolution) you will get a line like

```
FWHM (km/s & A) were  0.0  0.000 <CR> to accept, or enter values
```

Entering

```
6.6
```

will result in the whole spectrum being convolved with a Gaussian with FWHM=6.6 km s^{-1}.

If there is a file associated with the key RESFILE in the data header (see VPFIT description), then this is used to determine the instrument profile and the convolution is preformed automatically. To turn off the automatic convolution under these circumstances, then use `gp nosmooth`. You will then be asked to enter the FWHM as above - the answer to which may be zero. Then you can include profile fits from a number of files, and preform the convolution with the instrument profile only when you reach the last one in the list.

The resultant spectrum is placed in the continuum array within the program, using the wavelength scale appropriate for whatever data has been read in. All lines from the atomic data file (environment variable ATOMDIR) which fall in the spectral range of the data are included. The spectral data which has been read in is untouched.

If you wish to use a preset list of ions and parameters then respond to

```
 ion,col,bval,zed?    ..  or ...
 <filename> (fmt,clo,chi,zlo,zhi,blo,bhi,binc,type)
               [26,everything,everywhere]
```

with a filename, followed by the parameters indicated. By default the input format is as for the VPFIT output to fort.26, so an input file can be

```
%% HE0515rc.fits     1   4174.3265   4176.5921
%% HE0515rc.fits     1   4181.2306   4183.6780
! Stats:   2     1.5273723  194  182  0.090  0  !
 C IV   1.696696  0.000009     6.12   3.31  11.946  0.405  0 !
 C IV   1.696833  0.000031    15.59   5.20  12.498  0.147  0 !
 C IV   1.697105  0.000000     8.29   0.05  13.838  0.002  0 !
```

where the precise layout does not matter, and lines beginning '%%' and '!' are ignored, as are the error estimates, so a file containing

```
 CIV   1.696696    0     6.12   0  11.946
 C IV       1.696833 0.0  15.59    0  12.498  0.147  0 !
 CIV   1.697105  0.000000   8.29   0.05  13.838   0.002
```

gives exactly the same results. You need something where the errors appear just to keep the variable positions corresponding. Gaps between single letter atomic species and ionization level don't matter.

You can also use fort.13 format files - just put a '13' as the second parameter after the filename.

The rest of the parameters on that line are for those who want to exclude lines from the list by the sizes of the parameters. If a value is blank or zero then the limit is not applied, but otherwise only HI lines with clo< $\log N$ <chi, zlo< z <zhi and blo< b <bhi are included. All metals are included normally, but if the 'type' parameter (which is a character string) includes 'z' the restricted redshift range is applied to these as well.

I can't remember what 'binc' does, and invariably leave it as zero.

You can be reminded of the 'type' codes by typing '?' where it asks for the filename, so the sequence then looks like

```
....
 ion,col,bval,zed?    ..  or ...
 <filename> (fmt,clo,chi,zlo,zhi,blo,bhi,binc,type)
               [26,everything,everywhere]
?
 Control characters:
 c - continuum adjustments only
 d - do all but continuum adjustments
 e - emission lines
 i - ignore special (i.e. <>, __, >>)
 l - Ly-a only
 m - metals only
 n - no metals
 s - single ion (prompts)
 z - strict z range for everything
 ion,col,bval,zed?    ..  or ...
 <filename> (fmt,clo,chi,zlo,zhi,blo,bhi,binc,type)
               [26,everything,everywhere]
....
```

As with the interactive case, the spectral resolution is prompted for, and used in generating the final result.

4.2 Adding noise

Once you have the model spectrum you want in the continuum array, you may want to add some noise to it. Gaussian noise my be added to the continuum by using

`noise`

or just `no`, since only the first wo letters are checked. You are then prompted for a random number seed and a (uniform) noise level, with a dialogue which looks like

```
 random number seed

 632069
 s/n relative to peak continuum? <CR> = infinity
 (OR c1,c2 gives sigma=sqrt(cont*c1**2 + c2**2))
50.0
```

You can enter a random number seed if you want, but a carriage return results in one being generated internally from the computer's internal clock time. The value it uses is reflected (632069 here) in case you want to re-run exactly the same noise model again.

The noise generated may be uniform or continuum dependent. If you enter one number the noise is uniform, and entering two gives something which depends on the continuum level plus a background. The example above, if applied to unit continuum, gives a S/N=50.

There is one possible parameter to the `noise` command - if you enter `no data` then noise is added to the data (not the continuum) using the prescription above. This is in case you want to make the real data even noisier than it really is. The error array is updated as well by adding the original noise and the artificial noise in quadrature.

However, if you try to add zero noise to the data in this way, by

```
>> noise data
 random number seed

 632069
 s/n relative to peak continuum? <CR> = infinity
 (OR c1,c2 gives sigma=sqrt(cont*c1**2 + c2**2))
0.0
```

the behaviour is different. The program adds the noise specified by the error array to the continuum and places the result in the data workspace array. This might be useful if you want an artificial spectrum with the noise characteristics of the real spectrum.

5 Command mode plotting

Command line mode data plotting offers a few simple options to control what is plotted, the plot region limits and similar things. For general browsing around a spectrum it is probably more convenient to use the cursor mode (see Section 6), but once you have decided what you want then you can set up files to repeat plot operations for printing etc. using this mode.

The default plot setup is for the data to be plotted in black (or white on a black background), the continuum in green, and the rms fluctuations in magenta. You can change this at will.

It is activated by

`pl`

or alternatively `sp`. Then you should get a prompt, which answering `he` or `?` to gives a command list

```
 plot parameter? (type he for options list)
>  ?
 format is cc.. a..a b..b ...
 sc - set scale; lo - low chan; hi - high chan
 sn - scale range min chan; sx - max; ln; lx
 ym - min y; yx - max y; yb - default baseline
 qu(it); al(l); co(ntinuum); er(ror); re(sidual p1); rm(s array)
 no co (er, rm, re) - do not plot cont (etc)
 wl - low wavelength; wh - high wavelength (wn,wx scale ranges also)
 nu(ll), nx - nx/page, ny - ny/page
 la(bel); ca(ption); te(xt); ch(aracter) size
 sp - suppress error, ov # - overplot (bias=#)
 at - set attributes (type at for help);
 tf - tick marks from file
 gk - greek symbols over ticks; ng - numbers
 ns - no symbols, tl - tick top, len (fract)
 ve - velocity limits; wc - central wavelength
 rz - reference redshift for velocity plots
 wa - plot on wavelength scale
 as - ASCII to fort.17 (toggles back to PGPLOT)
 zs - suppress zero lines; zx - x=0 line; zy y=0; zb - draw both
 cm (n) - cursor style (when used)
 <CR> - plot on last device used
 plot parameter? (type he for options list)
>
```

Responding with a carriage return gives a plot of the data on the current plot device (it asks which if none has yet been opened, and the usual default is /xwindow), with wavelength limits the last ones set (all the data if a new data file has been read in), and showing the data and error values *vs* wavelength unless something else has been set. Most limit values are remembered and used for the next plot until over-ridden; exceptions are y-limits which are reset on re-entry so that the current range covers the data.

These commands just set flags which are used when the plot is actually performed, so if there are apparent internal inconsistencies then the last entry affecting that flag is the one that is used. So e.g. `co` followed by `no co` results in the data but not the continuum being plotted.

Most of these commands are described in the following sections.

5.1 Setting ranges

The range parameters are

- `sc p1` multiply the current vertical scale by p1
- `lo p1` set the lower end of the range to be plotted to data channel p1
- `hi p1` set the upper end of the range to be plotted to data channel p1
- `sn p1` set y-scale minimum to p1, leaving the minimum channel to be plotted as it was

- `sx p1` set y-scale maximum to p1
- `ln p1` set the lower end of the range to be plotted and data scaling range to data channel p1
- `hx p1` set the upper end of the range to be plotted and data scaling range to data channel p1
- `ym p1` set the minimum y-value to be plotted to p1
- `yx p1` set the maximum y-value to be plotted to p1
- `yb p1` set the default plot baseline to minimum of data & p1
- `wl p1` set the lower end of the range to be plotted to wavelength p1
- `wh p1` set the upper end of the range to be plotted to wavelength p1
- `wn p1` set the lower end of the range to be plotted & and y-scale range to wavelength p1
- `wx p1` set the upper end of the range to be plotted & and y-scale range to wavelength p1

5.2 Controlling what is plotted

- `sp` plot the spectrum only
- `er` plot spectrum & error estimate
- `co` plot spectrum & continuum
- `al` plot spectrum, error & continuum
- `rm` plot rms array along with whatever else has been set up
- `qu` quit routine without plotting anything
- `re p1 p2` overplot residual=(data-continuum)/(rms fluctuation) as well, with horizontal lines indicating 1-σ error ranges. `p1`, if present, is a rescaling factor and p2, if present, is a shift for the residual plot.
- `ov p1` overplot the new spectrum on the previous one, with the new spectrum biassed up in the y-direction by `p1`
- `no` p1, where p1 = da(ta), er(ror), rm(s), co(ontinuum or re(sidual). Don't plot the specified array.

5.3 Labels and axes

- `la s1 s2 s3` print label `s1` under the x-axis, `s2` for the y-axis, and `s3` as the caption. If any contain spaces put them in quotes. So, for example, `'wavelength (A)' flux 'CIV region'` will give an x-label of 'wavelength (A)', y-label 'flux', and caption 'CIV region'. On the other hand, `wavelength (A) flux CIV region` will give x-label 'wavelength', y-label '(A)' and caption 'flux', which is probably not what you had in mind.
- `ca s1` put a caption `s1` on the plot. Again use quotes if there are spaces, or commas, in the caption.
- `te s1 x1 y1` put the string `s1` with its lower left corner at position (`x1`,`y1`) in the plot coordinates.
- `ch p1` set character size to be `p1`× the default size
- `tf s1` read tick mark positions from a file of wavelengths
- `gk` put Greek symbols over the ticks where the ticks are labelled
- `ng` put numbers over the ticks where the ticks are labelled
- `ns` suppress tick labels
- `tl p1 p2` sets the tick top `p1` and length `p2` as a fraction of the y-size of the plot window.
- `zs` suppress zero lines
- `zx` draw x=0 line but not y=0
- `zy` draw y=0 line but not x=0
- `zb` draw both zero lines

5.4 Velocity scale plots

- `wa` Use a wavelength scale for the x-axis
- `rz p1` set `p1` to be the reference line redshift for velocity plots, including overlaid velocity plots
- `ve p1 p2` set the velocity range for a velocity plot to be from `p1` to `p2` relative to the reference redshift. Note that if you want to include zero velocity then `p1` should be negative and `p2` positive.
- `wc s1 p2` use the line from the atomic data file with ion s1 and wavelength nearest to p2 as the one for this velocity plot. `s1` should contain no spaces, so be something like e.g. 'HI', 'NV', 'SiIII'.

5.5 Styles and colours

The plot styles and colours for various curves can be set by setting attributes which are associated with each of them.These just set PGPLOT variables, and where numbers are used these are the ones associated with a particular attribute as given in the PGPLOT documentation. If you don't have that to hand, just experiment.

If you type `at` without any prameters, as below, you get a list of options

```
> at
 at
  at (curve) (attribute) (value)
  curve = data, error, rms, continuum, axes, ticks, residual, (fit region)
  attribute = colour, style, width of lines
  value = pgplot value
  attr = type, value = curve or line for lines
  attr = type, value = hist for histograms
 plot parameter? (type he for options list)
>
```

Colours are number coded by PGPLOT, so 1=white on black for screen plotting, black on white for hardcopy (this is the default). Others depend a bit on the settings for the medium, but roughly: 2=red, 3=green, 4=dark blue, 5=turquoise, 6=lilac, 7=yellow, 8=brown, 9 & 10 are different shades of green; 11 is another blue; 12=purple; 13=magenta; 14 & 15 are shades of grey.

Line styles are again set by PGPLOT number codes, so 1=continuous (default), 2=dashed line, 3=dot-dash & 4=dots.

Line widths are also as in the PGPLOT manual. A larger number results in a thicker line.

Line types are 'curve' for continuous or 'hist' for histograms. So

```
at da co 1
at co co 3
at co st 4
at da ty hist
```

results in the next plot having the data as white histograms (if the background is black, as on the screen) and the continuum green dots (if it is plotted).

5.6 Plot output

The plotting routine asks where you want the output to go on first entry, and on first entry only you can specify the number of plots per page in the x- and y- directions through `nx p1` and `ny p1`.

If you want to change the plot device then in command mode enter

`pc n1 n2`

which clears the current device and sets things up so next time there will be `n1`×`n2` plots on a page. The plotting routine will again request a plot device when it is used.

If you want an ASCII file of the plotted quantities, then

`as`

will result in the program writing the results to fort.17 instead of the PGPLOT window or device, without closing the plot device. To restore plotting to that device simply enter `as` again when you want that to happen.

6 Cursor mode plotting

Plotting the spectral data in cursor mode allows a bewildering set of options to allow you to look at the data, plot stacked sections for different ions to see if there is common velocity structure, make line positions, set up input initial guesses for the file read mode for VPFIT, modify the continuum, flag bad pixels, visually search for redshift systems, display fits from VPFIT and see where additional components might be needed, ... All of this is via single character input from the plot window, and unfortunately as a consequence some of the associations are obscure, to say the least. You can type '?' at any stage to get a list, which helps a bit.

To get into this mode you should first read in some data (of course), then type 'dc' to normalize to unit continuum (which is assumed), and then

`pg` - plot the data with cursor control

The cursor mode (pg) operation is fairly flexible, and mimics a few of the IRAF 'splot' options. What happens is determined by a set of one-letter commands in the graphics window. The list of these is growing, some of which are:

```
Left mouse button, or "e", expands plot
  (a second position is requested for the other limit)
Center button, or "r" replots
Right button, "q" or "Q" to exit
"?" type the current version of this list
  (omitting some of the more obscure ones)
" " print wavelength, flux at cursor position
"." shift range up
"," shift range down
"=" next velocity plot to a postscript file
"-" flag as bad data
"[" table match
"{" Ly-a max redshift
"@" show/hide ref. wavelengths
"a" plot whole array
"b" mark region boundaries (B,o)
"d" demagnify by factor 2
"f" fraction of pixels above cursor
"h" print line parameters for Ly-a (H)
"i" interpolate error
"j" change y of x-point nearest cursor
"k" change error to cursor y-value at that point
"l" print line parameters (L)
"m" wavc, me/rms, mean, rms, mean error
"t" velocity ticks on/off [M overrides "on"]
"u" renormalized velocity scale
```

```
 "v" velocity scale about cursor position
 "w" wavelength scale
 "x" interpolate data
 "y" max y from cursor
 "z" change continuum at point
 "*" print a *
 "<CR>" print a blank line
 "C" overplot continuum
 "E" replace error
 "F" change data format
 "G" plot data, continuum and error
 "I" interpolate continuum
 "J" change y of continuum point nearest cursor
 "K" suppress velocity plot caption (toggles)
 "M" mark lines
 "N" no line marking
 "P" command line prompt
 "R" replace continuum by mean
 "S" suppress error, plot data only
 "T" tick mark lines
 "U" flag as bad data with errors above cursor
 "V" velocity scale, using old redshift
 "Z" mark lines at reference redshift
 "0" default/extended cursor type (toggles)
 "7" extended cross-hair cursor
 "!" unknown line
 "~" snap to .gif
 "}" indicate wavelength regions used (toggles)
             (uses same color as residual plot)
 .. any other lower case letter for command line prompt
```

.. any unrecognized character will give the command line prompt, where the options are as for the plot command sp (and which can be seen by typing a "?" in the command window). Note that "A", "D" and "X" are equivalent to left, center and right mouse buttons. Other characters may be used for test functions, so don't expect any other character to necessarily give you a command line prompt - it is safest to use "P" if you want to do that.

You can ask (on the command line) that the continuum be plotted with a particular colour (at co co 6 will give you purple continuum), and this may contain fits to the data which are added in to the continuum using gp (see below).

6.1 Velocity plots

One use is stacked plots of absorption lines on a common velocity scale, so you can easily see if HI, SiIV, CIV are all there together, or if low ionization lines are present, or the whole Lyman series, or low redshift Mg and Fe, or anything you care to look for. First you need to set up a file with the ion and wavelengths listed, one per line, like:

```
H I   1215.6701
H I   1025.7223
```

```
H I    972.5368
H I    949.7431
H I    937.8035
C III  977.020
C IV  1548.195
C IV  1550.770
N III  989.799
N V   1238.821
N V   1242.804
O VI  1031.927
O VI  1037.616
SiIII 1206.500
SiIV  1393.755
SiIV  1402.770
```

Then add it to an access list using `pf filename`, where `filename` is whatever you have called it. You can do this for up to 35 files, and select which you use later from a list which is displayed when you type "v" or "u" from a wavelength plot.

If instead of giving a filename you use `pf 0` (the 0 is important - a blank will cause a filename to be asked for) then you can enter a list of (up to 16) lines interactively. You then enter ions and wavelengths from the terminal, e.g.:

```
 ion, wavelength <CR> to end
MgII 2796
MgII 2803
MgI 2852
```

Note that you don't have to give accurate wavelengths for either the file or the interactive form. The program chooses the ones from the atomic data file which are nearest to the input wavelength.

With either or both of these set up, you can use `pg` for velocity plots.

```
>>pg
 plot parameter? (type he for options list)
> sp
 sp
 plot parameter? (type he for options list)
>

 PGPLOT device? (? for list)
> xw
```

Expand plot if needed: [program is now in cursor mode, so enter letter commands etc from the pgplot window]

```
 Cursor ("e" to mark edges, "q" when OK)
left mouse button
  Channel  19906
 right edge..
```

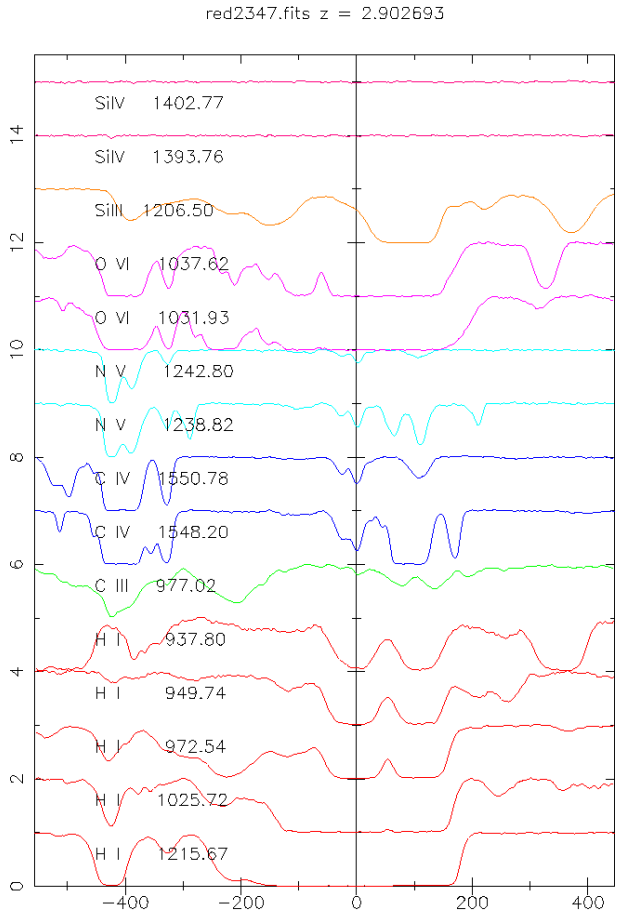

Figure 1: A common redshift plot for high ionization lines, produced using the line list given in the text.

```
right mouse button
  Channel  24638
 Cursor ("e" to mark edges, "q" when OK)
```

[mark off line of interest] `v` (in cursor window, with the cursor x-position set where you want the velocity zero to be)

```
 File ID?
  1   hiion.pg
  2   loion.pg
  3   lowz.pg
  .. in plot window
```

Then type e.g. `1` in the cursor window and a list of possible ID's from the file hiion.pg will appear on the plot. Click the left mouse button on the appropriate one and a velocity scaled plot will appear for all the lines in the list, with each marked. An example is shown in Fig. 1.

If you want to access the interactive list then just type `0` instead of any of the numbers associated with a file. Then, for the example above, you would get the list MgII 2796, MgII 2803, MgI 2852 from bottom to top.

You can shift the centre by moving the cursor to a new position and entering 'v' again, as often as you want. To look at a particular line in detail, just use the left cursor button to mark the left and right positions on the screen, making sure the y-position is about level with the label, and you will get it on a wavelength scale. Using 'v' and the appropriate line ID you can go back to the velocity plot if you want to. When you are fed up with this, 'a' gives you the full spectrum back – and if you get lost (or the program does), use 'a' to restart. 'q' quits this mode, so you can read in more data or whatever.

If you are visually seaching for a system, then an easy way to do it is to set up the lines you want in a file, start with 'v' or 'u' as above, and then put the cursor near the right (for searching up in wavelength) or left (for searching down), and hit 'v' (or 'u') repeatedly until the lines you want in the pattern appear together. In this mode the ',' and '.' keys have the same effect as 'v', so, depending on the position of the cursor, you may find yourself going in the opposite direction in wavelength to the one you expect! The redshift of the zero velocity line in the plot is given at the top of the pgplot window.

You can send the stacked velocity plots to a postscript file, for which the best approach is to create one on the screen using 'v', when you are happy type '=', then place the cursor on the centre of the line you want and type 'v' again. The interactive screen will vanish while a hardcopy is done, and then what you sent to a file will appear on the screen. Since you might want to do this more than once, the files are named in order pgp101.ps, pgp102.ps etc so you don't overwrite the hardcopy each time you use '='.You should then print the plotfiles, or examine and print at leisure. If you exceed 99 in a session I have no idea what happens. If you restart the program, it starts at pgp101.ps again, so any old one by that name will be overwritten.

You can also send other plots from the interactive plot mode to a postscript file. A way do this is to set up the display you want, then '=', and then type 'w' to redraw the plot. An alternative approach is to use the non-interactive plotting routines ('pl') and set up a list of commands which will do similar for you, and then either cut and paste into the command window or set up as a redirected input file. This is the only method of getting velocity stacked plots with data from multiple files.

If you type 'M' then wavelength (but not velocity) plots have the line positions from all ions in the input Voigt profile generating file marked, with identifications. This can be useful when unravelling messy blended systems. In velocity plots only the tick marks are displayed.

'Z' takes the last reference redshift used (from 'v'), and plots the positions of all the lines in the atom.dat file at that redshift when you replot on a wavelength scale. You can set the redshift by using

```
pp z 1.412354
```

before entering the cursor mode plot. You'll need to turn it off if you want ID's from a file though, using

```
pp f fort.26
```

or whatever.

If you have an overfull atomic data file so wind up with confusing and irrelevant tick marks all over the spectrum, then you can make a shorter one and get the program to use it by typing `at`. Then you are prompted for a filename. You can also use a shortened version of the distributed one by:

```
>>at
  Atomic data filename
/(path)/atom.dat short
 Using data from : /(path)/atom.dat
>>
```

With the second parameter, the contents of the file up to the first occurrence of the comment line

```
! -- Suppl
```

are read in, and the later entries are ignored.

6.2 Start files for VPFIT

With the velocity plots described above you can use the cursor to set up start files for fitting systems using VPFIT. To delineate fit region boundaries, put the cursor on the left edge in the velocity sgement you are interested in, type `b`, move to the other end and type `b` again, and a line like

```
 %% red2347.fits  1  4825.44922  4839.00928
```

will appear in the text window. You can do this for as many regions as you wish. Then, for line ID's put the cursor where you think a feature might be and type `l`, and you should get e.g.

```
N V     2.896937  0.000000     5.41    0.00 12.704  0.000
```

Again you can do this for several possible positions. Then all you need to do is cut and paste into an editor window to produce a file which might contain:

```
 %% red2347.fits  1  4825.44922  4839.00928
 %% red2347.fits  1  4841.30859  4854.33545
N V     2.896937  0.000000     5.41    0.00 12.704  0.000
N V     2.897233  0.000000    17.26    0.00 14.407  0.000
N V     2.897659  0.000000    13.45    0.00 14.068  0.000
N V     2.898401  0.000000     8.09    0.00 13.186  0.000
N V     2.901424  0.000000    23.79    0.00 12.895  0.000
N V     2.902389  0.000000    15.46    0.00 13.162  0.000
N V     2.902779  0.000000     5.79    0.00 12.951  0.000
N V     2.903039  0.000000     7.92    0.00 12.318  0.000
```

This is one of the two formats accepted by the `f` option (start from file) in VPFIT.

You can also type `b`,`l` and `h` in the wavelength rather than the velocity plots. The `b` action is as before - it just prints out a wavelength region against a filename. The ion associated with `l` is the one taken from the last velocity plot when you set the cursor

on a region and typed `w`, or used the left cursor to define a wavelength region from a velcity plot. It is the line in the list appropriate to the region you chose, and it is left as the default ID for wavelength plots until you go through the region selection from a velocity plot a further time.

The other format is also catered for. Just use upper case letters `B` and `L` instead of the lower case ones - the only trouble is then that you need the '*' list separators, which you can either edit in or obtain by typing `*` in the plot window.

In principle this information could be written direct to a file, but if you are anything like me you'll backtrack, make errors, and change your mind enough that cutting and pasting is a quicker option in the long run. However, if you want to write the results of the `b` and `l` commands to a file, then before you start `pg` set

```
pp s fgout.dat
```

and they will be written to a file `fgout.dat` (which could be anything) in the order in which you executed them. You can then edit this file, if you can remember which sets are associated with each other.

7 Modifying the error arrays

A feature (?) of the UVES pipeline is that the error estimates in the data at the bottoms of saturated absorption lines are too low by a factor of roughly 2. Consequently, even if the zero level is correct, or corrected (it is normally to high by something up to about 2% of the continuum, but it is wavelength dependent), a satisfactory χ^2 will never be reached when using VPFIT on these features because the fluctuations in the bases of the lines are significantly larger than the error estimates.

RDGEN has an option for modifying the error arrays to avoid this problem. The approach is not based on any analysis of the expected errors - all it does is take a function which is roughly 2 where the signal is near zero, and roughly 1 where the data is close to the continuum, and multiply the error arrays by this function.

For any pixel i with wavelength w_i, data value d_i and continuum estimate c_i the error esimate e_i is modified so that the new value

$$\begin{aligned} \epsilon_i &= e_i \times \left[a + \left(b + c \log w_i + d(\log w_i)^2 \right) \right. \\ &\quad \left. \times \left(1 - \max \left(0, \min \left(1, \frac{d_i}{c_i} \right) \right) \right)^s \right]^{1/t} \end{aligned}$$

where the user specifies s, t, a, b, c and d. For things to change little when the data is near the continuum, and by a factor 2 when the data is zero, then it is worth choosing $s = t = 4$, $a = 1$ and $a + b$ such that $(a + b)^{\frac{1}{t}} = 2$, so $b = 15$ if there is no wavelength dependence ($c = d = 0$). Those are the default values, but you can feed in any values for these six coefficients which you feel happy with.

To activate this option on data which has been read in, just type `me` (for modify error) and if you have the current version the response will be something like:

```
 Modify error & expected fluctuations:
 multiply by f=[a+B*(1.0-data/con)**s]**(1.0/t)
 where B=b+c*log10(lambda)+d*log10(lambda)**2
 for wavelength lambda
 If (1.0-data/con)<0, set to zero; if >1 set to 1
 Enter s,t,a,b,c,d (need not be integers)
Defaults:  4.0 4.0  1.00 15.00  0.00  0.00
 to do nothing, enter q or Q
```

The correction is applied to BOTH the error array and the array giving the expected rms fluctuations in the data, if it is present.

Note that you have a last-minute bail-out feature in case you change your mind - just type 'q'.

There is not a great deal of point in just modifying the error arrays internally. You need to write them out for VPFIT to use them. For fits files use the `wr` command (see section 3.2). This can involve shuffling things around a bit in RDGEN, but you don't need to do it often. The sequence might be:

```
>>rd datafile          [ Read in the data
>>me                   [ modify the error arrays
Modify error & expected fluctuations:
 multiply by f=[a+B*(1.0-data/con)**s]**(1.0/t)
 where B=b+c*log10(lambda)+d*log10(lambda)**2
 for wavelength lambda
 If (1.0-data/con)<0, set to zero; if >1 set to 1
 Enter s,t,a,b,c,d (need not be integers)
Defaults:  4.0 4.0  1.00 15.00  0.00  0.00
 to do nothing, enter q or Q
                       [ Carriage Return gives the defaults
>>wr newdata.fits      [ write out a new data file
>>cy ew                [ copy modified error array to workspace
  Copy data array ew
>>wr newdata.sig.fits  [ write out modified error
>>cy re                [ copy rms fluctuation arry to error
  Copy data array re
>>cy ew                [ modified fluctuation array to workspace
  Copy data array ew
>>wr newdata.rms.fits  [ write out fluctuations
>>cy cw                [ continuum array
  Copy data array cw
  Continuum to workspace
  error set to zero
>>wr newdata.cont.fits [ write out continuum
```

You may not need to rewrite the data or the continuum, but the above sequence will give a complete set of files for use with VPFIT.

An alternative for NORMALIZED DATA is to write the file out in a format similar to that written by the UVES_popler package (but without all the data fields to do with original continuum and clipping information). This is much simpler - after the `me` just use `wr` with a second parameter `UV` , so

```
>>wr newdata UVES_popler
```

and the reponse will be

```
Writing to file: newdata.fits
Wavelength coefficients to header
Wrote UVES_popler file newdata.fits
```

The file 'newdata.fits' then contains the data, errors, and a flag which tells VPFIT that the data is normalized to unit continuum.

8 Column density upper limits

If the redshift and Doppler parameter are determined for some reference ion, for example CIV, then for other ions within the same region the redshift will be the same and one can estimate a range of acceptable Doppler parameters b under the assumption that there is a turbulent component $b_{\rm turb}$ which is the same for all ions and a thermal component $b_{\rm T}$ which is proportional to the square root of the mass m of the ion. Then the two components add in quadrature to give the actual Doppler parameter, so

$$b^2 = b_{\rm turb}^2 + b_{\rm T}^2.$$

where $b_{\rm T} = \sqrt{\frac{m}{m_{\rm ref}}}\ b_{\rm T,ref}$ where the subscript 'ref' is the value for the reference ion. Possible extreme values for b are then determined assuming that the Doppler parameter for the reference ion is fully turbulent and fully thermal. In the implementation used here this range was extended by using the 1-σ error estimates for the reference ion, so reference Doppler parameters $b_{\rm ref} \pm \sigma$ were used and the most extreme values of b adopted to give the Doppler parameter ranges for each ion.

Using the redshift for the reference ion, and a sequence of Doppler parameters from the minimum to maximum obtained in this way, a grid of Voigt profiles convolved with the instrument profile is constructed for the transitions of the test ion available in the observed range. The line profiles arre compared with the data, and a χ^2 determined for pixels where the Voigt profile was below the data value plus 1-σ, and within $2b$ of the line center for each transition. If no pixels satisfy this criterion then the column density iss increased until some do. The column density limit for each b-value is taken as the highest value for which the χ^2 value over this range had a probability of occurring by chance of less than 0.16 (corresponding to a 1-σ one-sided deviation, but this can be changed). The final overall limit which is adopted is the maximum of these over the range of Doppler parameters. This yields a maximum possible column density for the ion even in the presence of blends, since it is effectively only the pixels where the trial fitted profile is too low which contribute to the significance level.

The chance probability criterion for accepting or rejecting possible line profiles is arbitrary, so there seems little point in iterating or interpolating to achieve high accuracy. We adopt a column density step of $\Delta \log N = 0.05$ for $\log N < 13.5$, and double this for higher values.

To implement this in RDGEN first load the spectrum with the error estimates and the continuum, **NORMALIZE TO UNIT CONTINUUM**, and then type `uc` (for upper column density limits, pixellation as for the data) or `uc subpix` (gives upper limits with subpixels used if the Doppler parameters are small - so it is slower).

This results in an input request:

```
xn,xb,nminf,pchslim,ctype,vres(km/s),nmdef,bmin
                                    [1 2 5 0.160 cmin 6.70 3 0.50]
```

for which the easiest response is a carriage return, which yields the default values given in the parentheses. This means that

- `xn=1` For comparison with the data, only data below the fit + `xn` $* \sigma$ is included. The default is 1σ.
- `xb=2` The comparison region is $\pm$`xb` $* b$ from the line centre, where b is the Doppler parameter for the lines.
- `nminf=5` The minimum number of pixels in the comparison range for all lines used.
- `pchslim=0.16` Is the (one-sided) probability limit for the χ^2 value for a trial column density and Doppler parameter to occur by chance. The value 0.16 corresponds roughly to a 1σ upper limit.
- `ctype=cmin` Use noise estimate if no line is detected, so continuum bias is less important. To turn this off, type `nocmin`.
- `vres=6.7` Instrument resolution in km s^{-1}.
- `nmdef=3` Min number of pixels if a single line is used.
- `bmin=0.50` Is the minimum b-value allowed for the search.

The ones you are most likely to change are the probability and the resolution - you'll need to enter values for the others under those circumstances.

Responding `?` gives some help.

After this you get

```
 Ion,bval,(err),redshift,newion,(lambda,lambda...)
```

Here you must enter something - the reference ion, normally the one you have used to determine the redshift, its Doppler parameter (with an error if you wish), the redshift, and the ion which you wish to determine an upper limit for, followed by a list of wavelengths to be used to set that upper limit. If the wavelengths are omitted, it just takes the first two from the atomic data file and uses those. If you want something different, give the wavelengths. These need not be accurate wavelengths - the program searches the atomic data list for the nearest, and uses those.

So, given a CIV with Doppler parameter $b = 4.0 \pm 0.2$ at $z = 2.385355$ for which an upper limit for OVI is sought an appropriate entry would be

```
CIV 4.0 0.2 2.385355 OVI
```

Then you get

```
  .. third parameter taken as bval error
 Using   1031.9261  1037.6167
 1  4.2  12.05  1.17690025 23
 2  3.29246625  11.95  1.19452383 18
  O VI    2.385355SZ 0.000000    4.20SB   0.00          <12.050
```

The intermediate lines give trial Doppler parameters and limit column densities (with χ^2 and number of channels) for those Doppler parameters, and the final line gives the parameters for the highest column density case found. The format looks a bit strange, but is chosen to be compatible with the summary output files from VPFIT, so there are some letters (SZ & SB) and zeros which mean nothing here.

If you are using the potentially slower sub-pixel mode, then the input ion request becomes

```
 Ion,bval,(err),redshift,newion,(lambda,lambda...)
                                 (or filename, column)
```

so instead of wavelengths on the command line you can have a file of the ones you want to use, one per line. So if you wanted to look for molecular hydrogen, J=0, you might enter

```
H2J0 5.0 4.0 2.059331 H2J0 temp.in 2
```

where `temp.in` might contain e.g.

```
H2J0  1108.1273270 0.001664570 1.868E9  2.01588 -0.00800319  ! L  0 R(0)   1a
H2J0  1092.1952340 0.005783580 1.741E9  2.01588 -0.00092454  ! L  1 R(0)   1a
H2J0  1077.1387420 0.011667900 1.632E9  2.01588  0.00558220  ! L  2 R(0)   1a
H2J0  1062.8821370 0.017895200 1.536E9  2.01588  0.01156759  ! L  3 R(0)   1a
H2J0  1049.3674390 0.023192900 1.450E9  2.01588  0.01706801  ! L  4 R(0)   1a
H2J0  1036.5458060 0.026833700 1.370E9  2.01588  0.02211196  ! L  5 R(0)   1a
H2J0  1024.3739540 0.028708500 1.300E9  2.01588  0.02672449  ! L  6 R(0)   1a
H2J0  1012.8130270 0.029702000 1.200E9  2.01588  0.03092982  ! L  7 R(0)   1a
H2J0  1008.5519150 0.015349300 1.180E9  2.01588 -0.00476718  ! W  0 R(0)   1a
```

which was simply cut and pasted from the atomic data file. The output then can look like

```
  .. third parameter taken as bval error
Using   17 wavelengths from file
 Substeps:           7
          9  lines in            9  spectral regions
   #      b     log(N)  chi2/n   n
   1     1.000  17.30   0.864   38
   2     1.442  16.70   1.062   28
```

```
   3     2.080  15.90   1.019   32
   4     3.000  15.50   0.932   35
   5     4.327  15.30   1.030   56
   6     6.240  15.10   1.125   71
   7     9.000  14.90   0.827   61
  H2J0    2.059331SZ 0.000000    1.00SB   0.00          <17.300
```

Some results using this are given by Schaye et al. (MNRAS, 379, 1169, 2007 = astro-ph/0701761), and the description of the method here is essentially the same as is given in their Appendix.

9 Other programs

9.1 Auto-startup files for VPFIT

If you have a lot of Lyα simulations it is a pain to have to set up fitting regions by hand. If the simulationed spectra are small enough you can cover the whole lot in one fitting region, of course, but if there are more than about 80 - 100 lines the fitting process can be too slow. Ther is an ancillary program AUTOVPIN which can set up region limits for you, based on where there might be continuum regions. It takes output from the `abs` (find absorption lines) option in RDGEN (the results of which are usually written to fort.9), and uses this to generate region limits. It is not at all sophisticated - all it does is try to extend the regions containing the absorption lines listed by `abs` by a bit, subject to a constraint that they should not overlap. If they do, it merges them.
To run it you first need to 'make autovpin' sometime, and then

```
$./autovpin
 Spectral data filename?
spectrum.fits
 linelist filename? [fort.9]

 output filename? [vpin.dat]
vpin.dat
 List order as input, or reversed? (i/r) [r]

 sigmathres,pbtythres,maxext,minext
 [4.5,0.0,20,5]
5 0 20 9
 Maximum redshift, # Lyman series lines?
 (leave second one blank if just want Ly-alpha)
3.05
 Using data from : /home/rfc/cflib/vpfit9.3/atom.dat
 Atomic data table contains 679 lines
 1 Lyman series lines used
 54 regions given from a total of  93
$
```

The spectral data filename is needed only to put it into the VPFIT startup file, and it does not need the `.fits` part which is put in automatically by VPFIT if it is omitted.

The real work is done using the *fort.9* file, where regions containing complexes with significance >`sigmathres` and chance probability >`pbtythres` are included. The actual limits to the regions are extended by the minimum of `maxext` and half the distance to the next complex on either side to include some continuum, unless this extension is less than `minext`. In the latter case the two regions are merged into one. In the example above a possible 93 regions in *fort.9* became 54 after such merging.

The `vpin.dat` file then contains something like

```
%% spectrum.fits  1   4862.49023  4880.18213

%% spectrum.fits  1   4855.69092  4860.35986

%% spectrum.fits  1   4826.90967  4855.6499

%% .....
```

which is fine as a startup file for VPFIT. In the absence of any initial values it guesses Lyα based on an a rough internally generated line list, and goes on from there. As suggested by the above, you can put in the Lyβ, γ etc regions as well, but this has not been used much, so you should treat it as largely untested.

A Command list

The following is a list of some of the commands within RDGEN, and brief one-line descriptions of what they might do. Some ask for additional information, so if you are not sure what something does, try it and see if the prompts which follow are obvious. It is hard to do any permanent damage - unless you use one of the two data write (`wr` or `wt`) routines and over-write something of course.

```
ab generate absorption list (p1:file)
ad linearize and sum spectra
at change atomic data file
bb generate a black body spectrum
cy copy data arrays
dc divide data by continuum (p1: low threshold)
ex exit program
fq list fits files
gm generate metal list and display
gp generate Voigt profiles in continuum
hd dust extinction from Hb/Ha
he (or ?) give a list of commands
le set data array length
lo exit program (for IRAF devotees)
mc multiply data by continuum
me modify error arrays
mf median filter data, result to continuum (p1: no. of pixels)
mu multiprocess files
no add noise to continuum -> new spectrum
pc close plot device
pf linelist file name input
pl plot the data (same as sp)
pg plot with cursor control
pw print wavelength coefficients
qu quit program
rc recall internal data
rd read FITS, IRAF or ASCII data file
sm (p1) boxcar smooth the data, width p1 pixels
sp plot the data
st store data internally
sw swap spectrum with stored one
sz data stacker
tp tickmarked velocity plot in a standard format
uc estimate upper limit to the column density for an ion
un set continuum to unity for current data
wa wavelength coefficient reset
wl set linear wavelength coefficients and length
wr write an IRAF data file
wt write data as a text (ASCII) file (p1: filename)
zr redshift correct to rest wavelength (p1: redshift, p2:n,b,[i])
zt ztable function
<  redirect input (p1: filename)
```

B Environment variable associated files

B.1 RDSTART

The following is an example of a file containing startup commands which would be executed when RDGEN starts if the file were associated with the environment variable RDSTART:

```
pf /home/vpfit/vpfit9.5/pgfiles/hiion.pg
pf /home/vpfit/vpfit9.5/pgfiles/loion.pg
pf /home/vpfit/vpfit9.5/pgfiles/lohi.pg
pp s fgout.dat
?
fq
```

The actions are to

- read three files of parameters for velocity stacked plots
- set up guess parameter file as `fgout.dat`, so guesses from appropriate cursor mode commands will be written there
- print the shortened help list
- print out the available FITS files in the working directory

The `hiion.pg` file might contain, for example,

```
H I   1215.6701
H I   1025.7223
H I    972.5368
H I    949.7431
H I    937.8035
C III  977.020
C IV  1548.195
C IV  1550.770
N V   1238.821
N V   1242.804
O VI  1031.927
O VI  1037.616
SiIII 1206.500
SiIV  1393.755
SiIV  1402.770
```

When it is used in curser plot 'v' mode (see Section 6 this gives a plot such as that shown in Fig.1.

B.2 VPFPLOTS

The following file, when associated with the environment variable VPFPLOTS, sets the continuum color to green, error to red, tick marks to turquoise, and region markers to yellow. The data display is set to histogram mode. This file is re-read for every plot, so if you want to change any of these you need to do it every time. See subsection 5.5 for more details.

```
at co co 3
at er co 2
at ti co 5
at fi co 7
at da ty hist
```