MultiNest for NEMESIS:

An intro to ellipsoidal nested sampling with NEMESIS, including examples and installation

Ryan Garland

&

Prof. Patrick G.J. Irwin

Part One:

An introduction to Nested Sampling & MultiNest

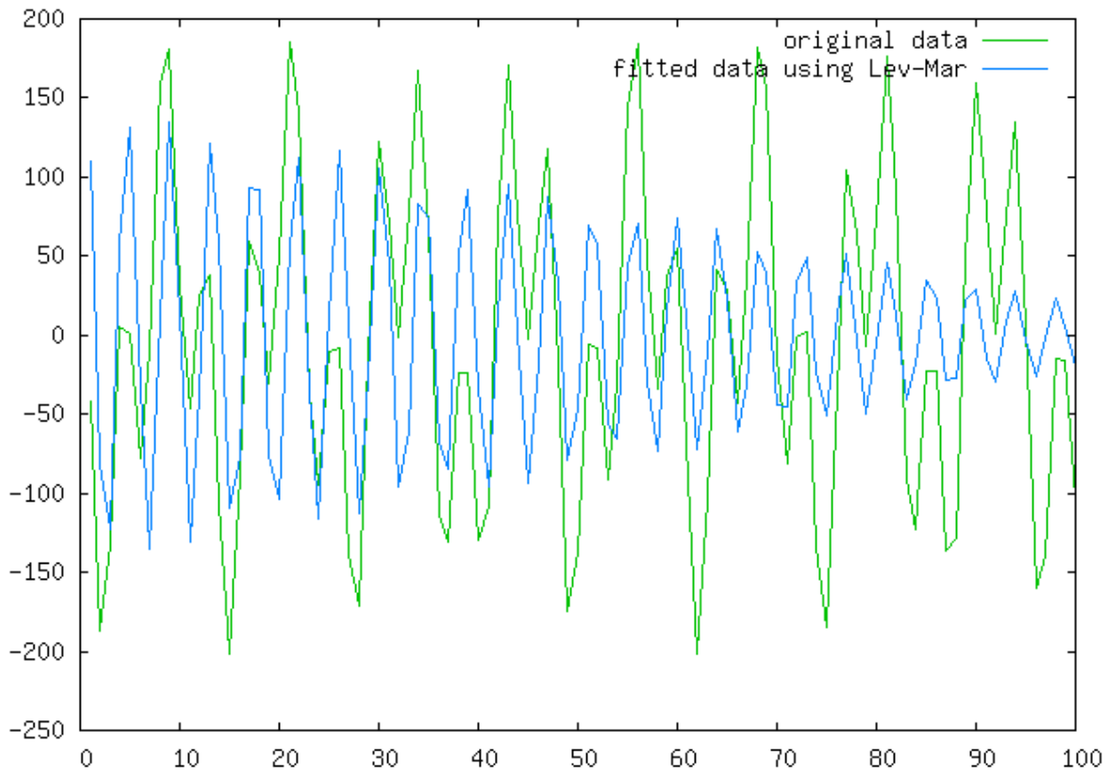
WHY NESTED SAMPLING? BACKGROUND ON OPTIMAL ESTIMATION

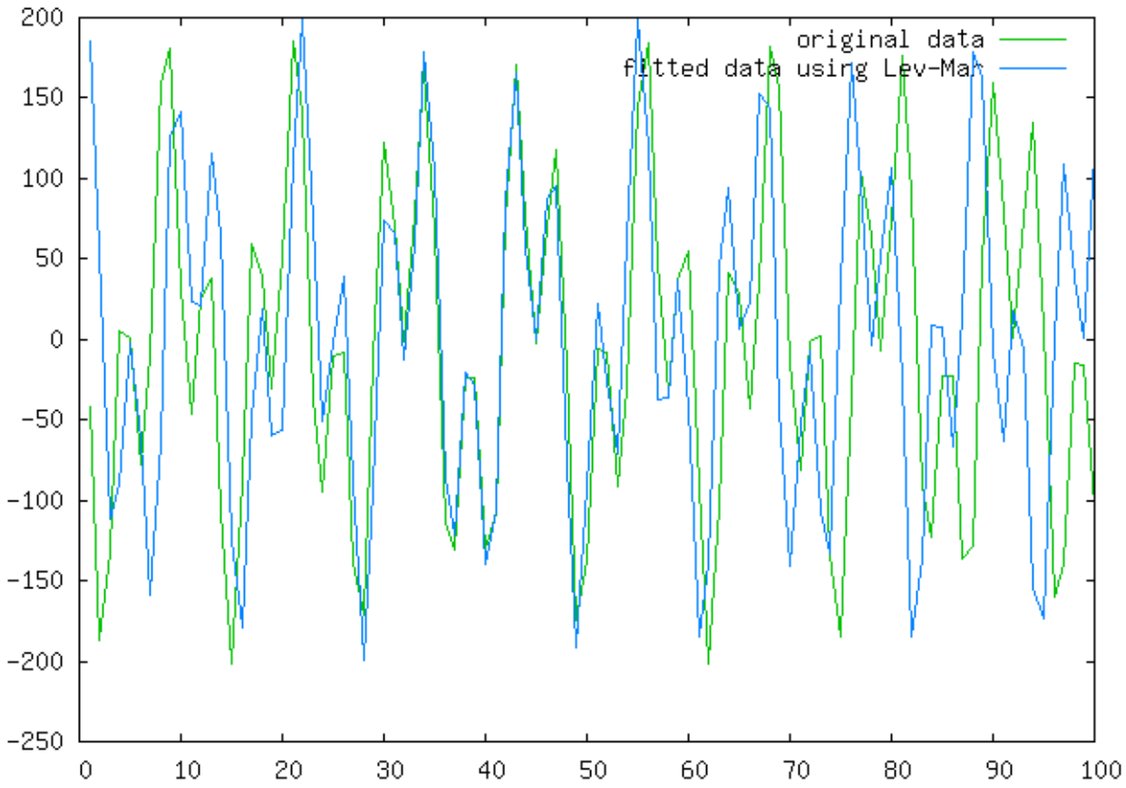
*NEMESIS by default uses optimal estimation as its inverse method. This is an extremely efficient inverse method, **if** your prior is close to your true answer.*

However, a large range of priors must be investigated to ensure your solution is not degenerate.This includes creating a grid of priors with means and (co)variances of each parameter which are largely unknown. There is always the chance that parameter space is not fully explored, and the converged solution is not the best fit to the data.

As well as this prior grinding issue, OE assumes that all errors are Gaussian, which is not necessarily the case.

WHY NESTED SAMPLING? BACKGROUND ON OPTIMAL ESTIMATION





Prior far from true solution

Prior close to true solution

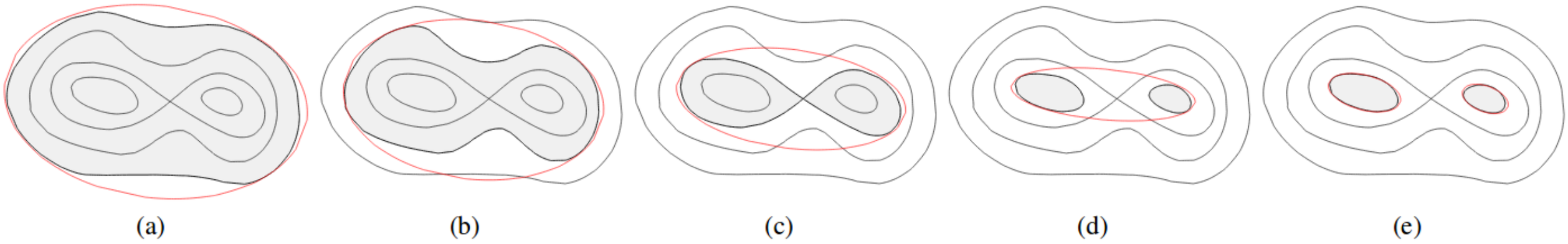
WHY NESTED SAMPLING? AN INTRO

Nested Sampling is an inverse method which calculates the Bayesian evidence by sampling the posterior distribution, and produces probability distribution functions for each parameter as a by-product.

- *does not assume anything about the shape of the probability distributions (OE assumes Gaussian)*
- *it fully explores parameter space (no more prior gridding)*
- *fully explores all degenerate solutions (multi-modal solutions now possible)*
- *automatically calculates the Bayesian evidence (can directly compare two different models, and which model is more appropriate when taking into account the number of free parameters in the model)*

WHAT IS MULTINEST? THE ALGORITHM

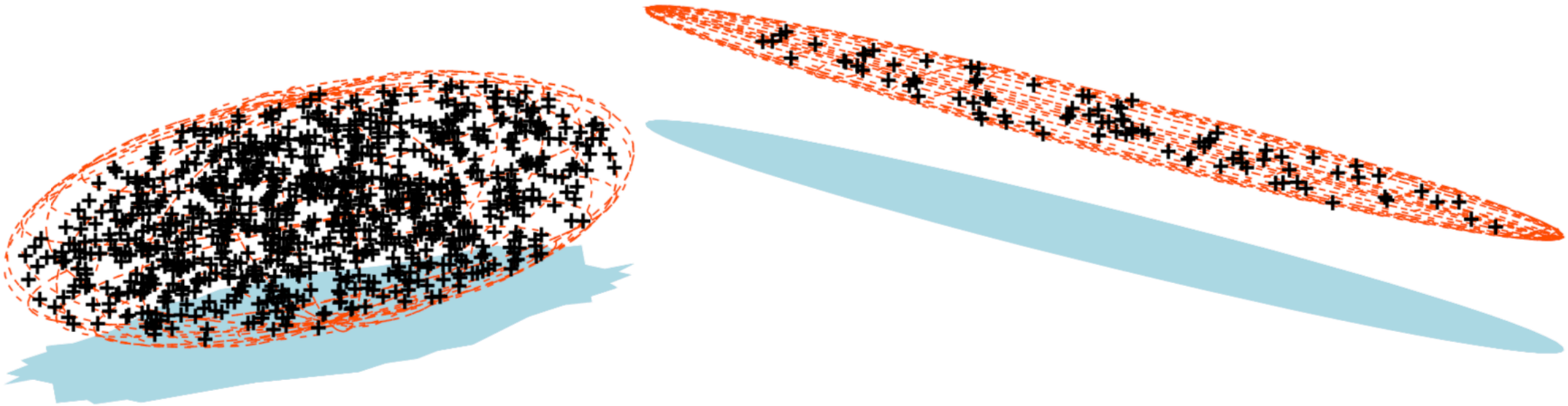
MultiNest is a simultaneous ellipsoidal nested sampling algorithm. In plain English, the procedure of the algorithm goes like this:



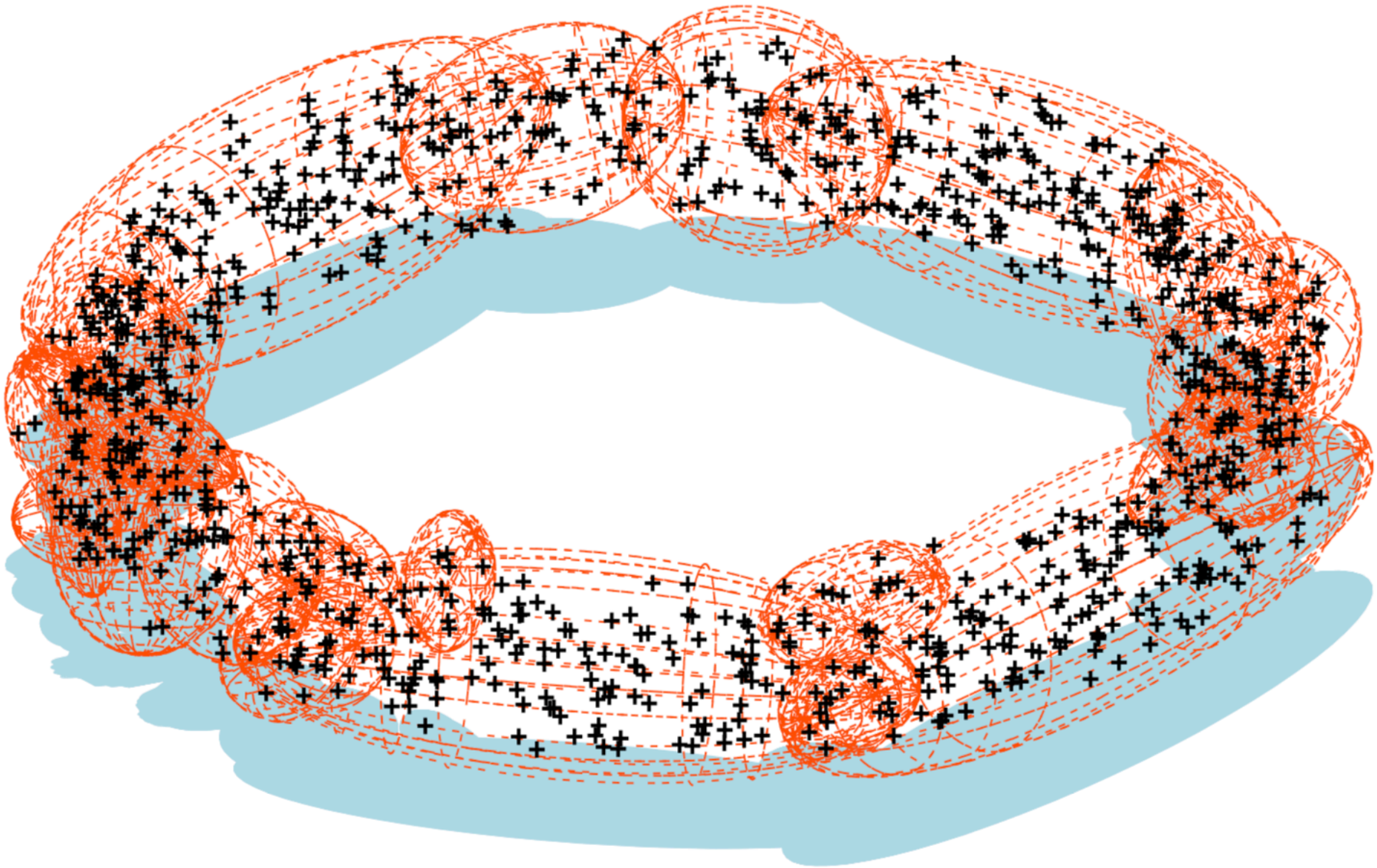
(a) (b) (c) (d) (e)

1) *Set up a range for your parameters (a uniform prior). For example, logVMR of H2O is between [-10,-1].*
2) *Normalise all of these ranges to be between 0 and 1. This creates a unit hypercube in parameter space. (N-dimensional cube with x,y,z etc between 0 and 1).*
3) *Run MultiNest. It uniformly samples the hypercube with 'live points' (state vectors), and fits a bounding ellipsoid around these points.*
4) *Checks are made to see if the posterior is multi-modal, and if this is true a k-means algorithm in applied recursively to split the ellipsoid into pairs of ellipsoids. (See Figure)*
5) *An associated spectrum is calculated with each live point, and compared to the data to calculate a probability (like a chi-squared).*
6) *The lowest probability live point is removed. A new live point is created, and a new bounding ellipsoid is calculated. Repeat from step 4.*
7) *Ends once a convergence criterion is met (change in Bayesian log-evidence 0.1).*

WHAT IS MULTINEST? MULTIMODAL POSTERIOR



WHAT IS MULTINEST? TOROIDAL POSTERIOR

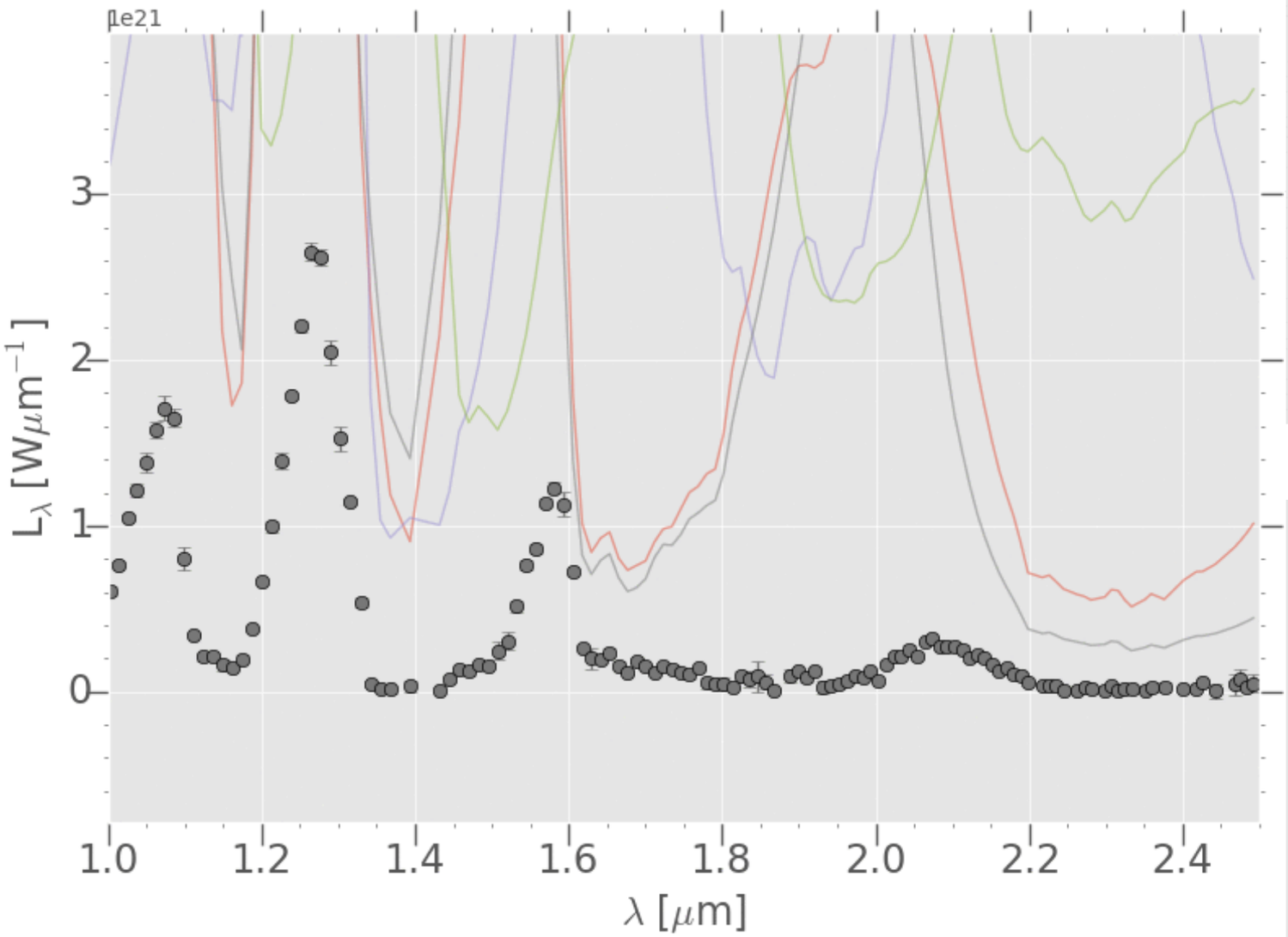


Part Two:

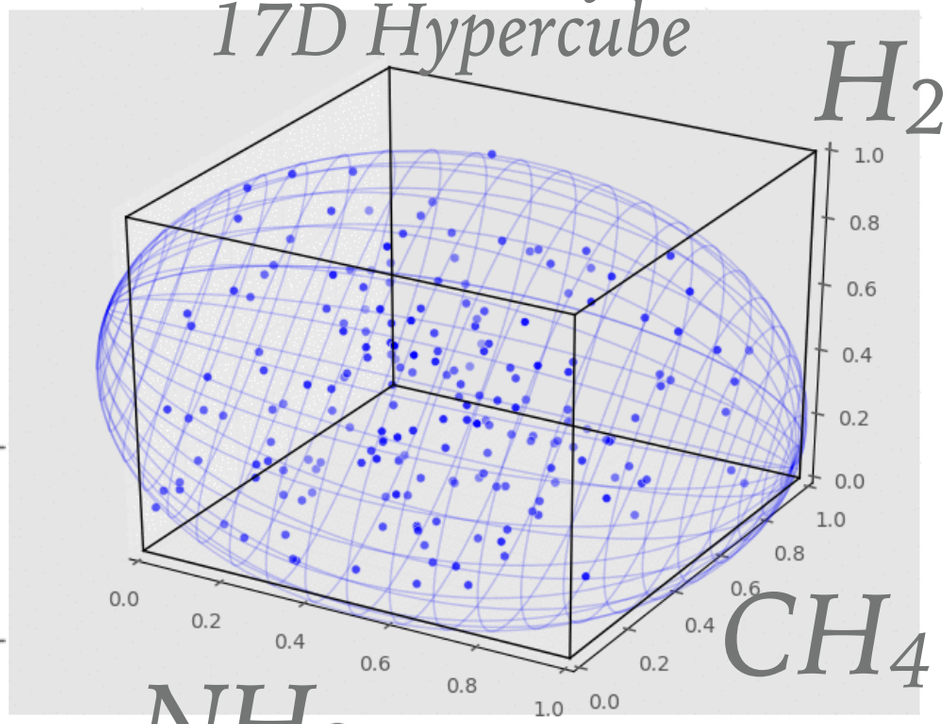
MultiNest & NEMESIS:
What MultiNest can do for you

WHAT IS MULTINEST? AN INTRO BY EXAMPLE: BROWN DWARF

Data (Near-Infrared Spectrum)



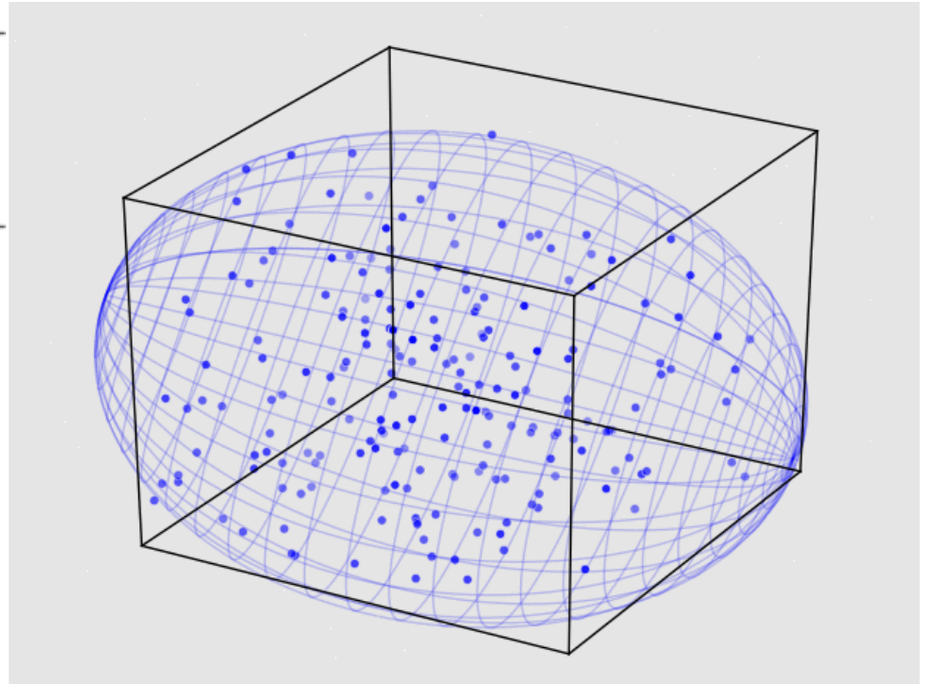
3D Slice of 17D Hypercube



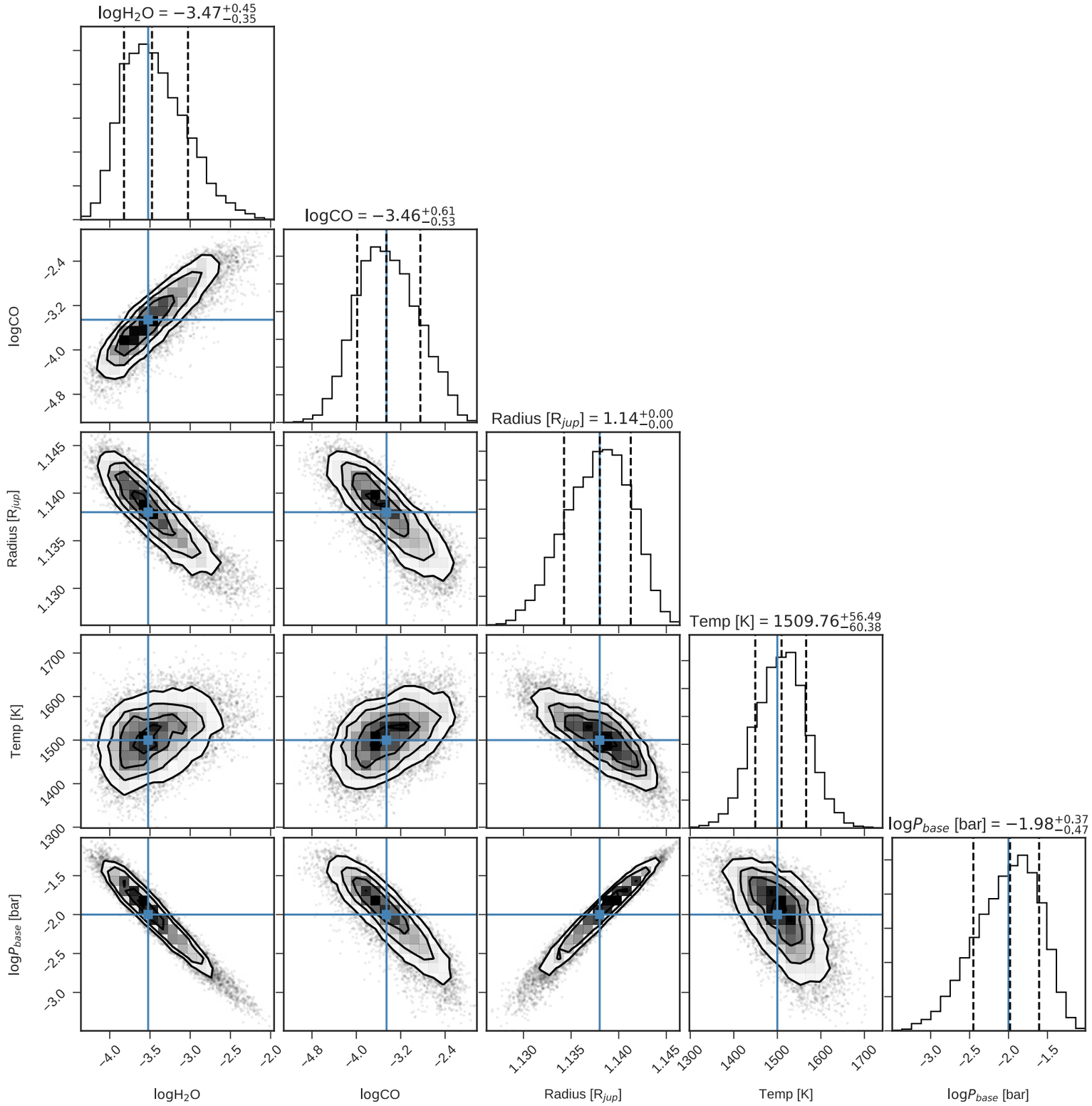
H_2O

CH_4

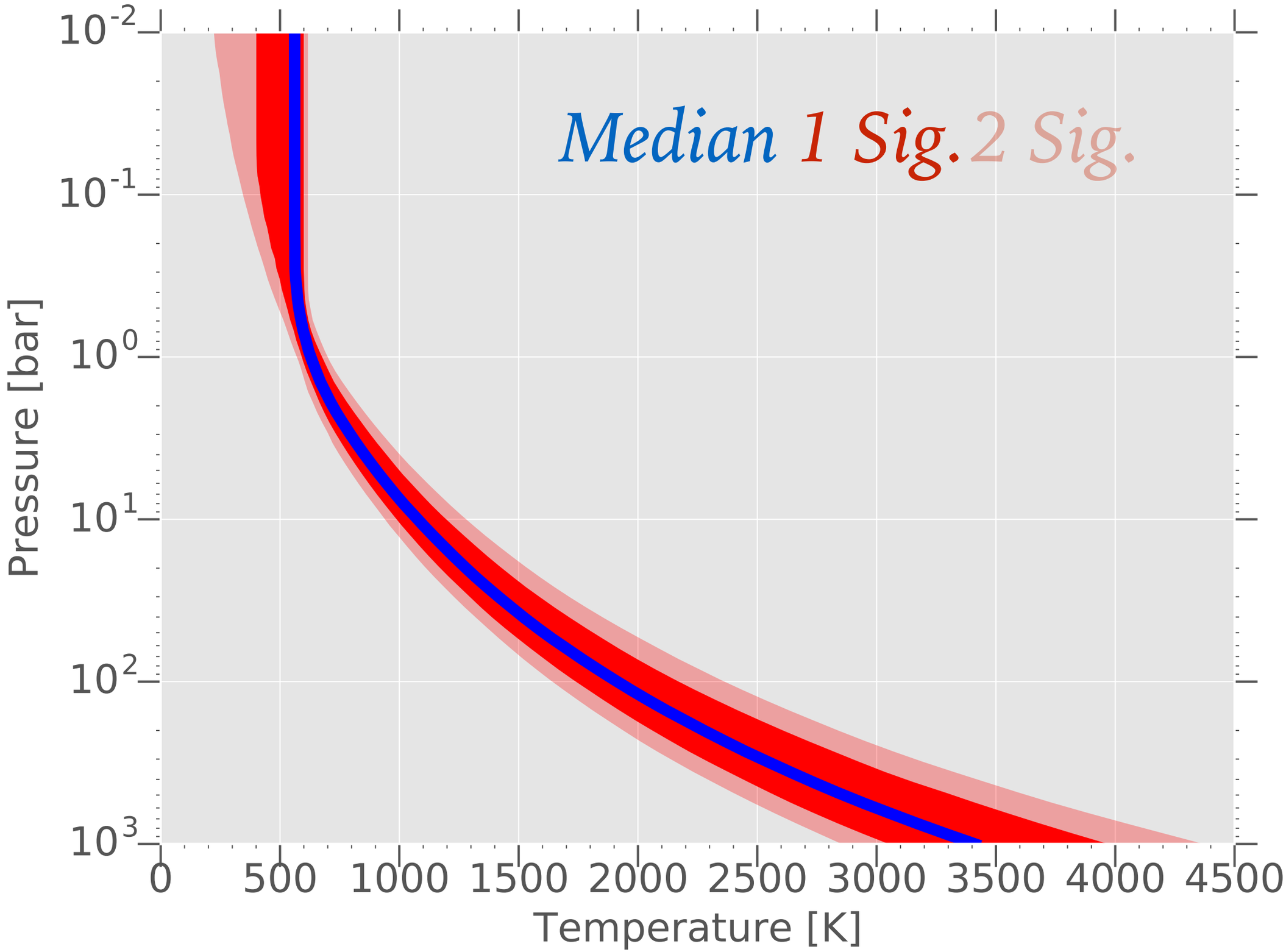
NH_3



MULTINEST RESULTS: AN INTRO BY EXAMPLE: PRIMARY TRANSIT FAKE DATA



MULTINEST RESULTS: TEMPERATURE PROFILE

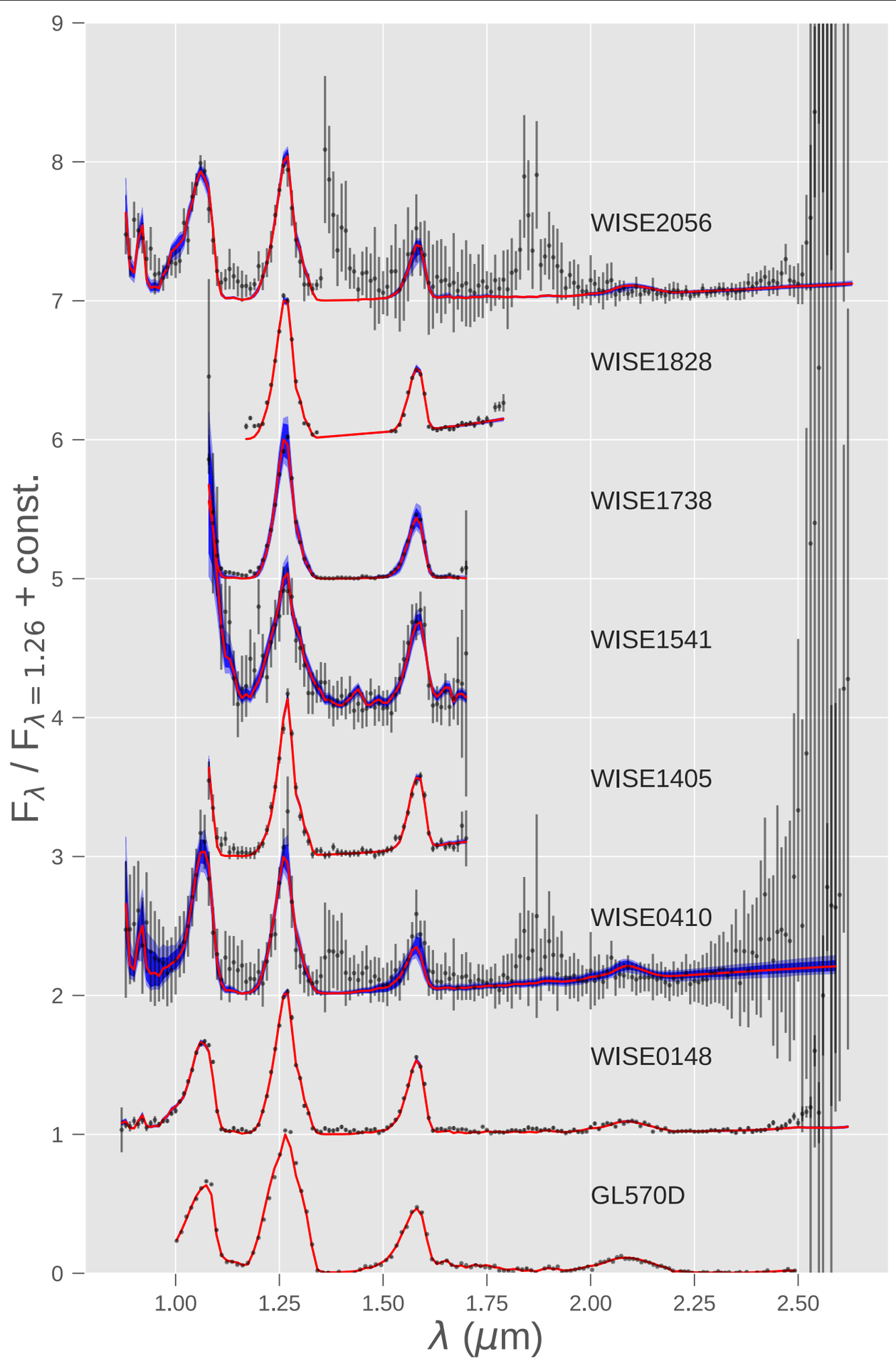


MULTINEST RESULTS:

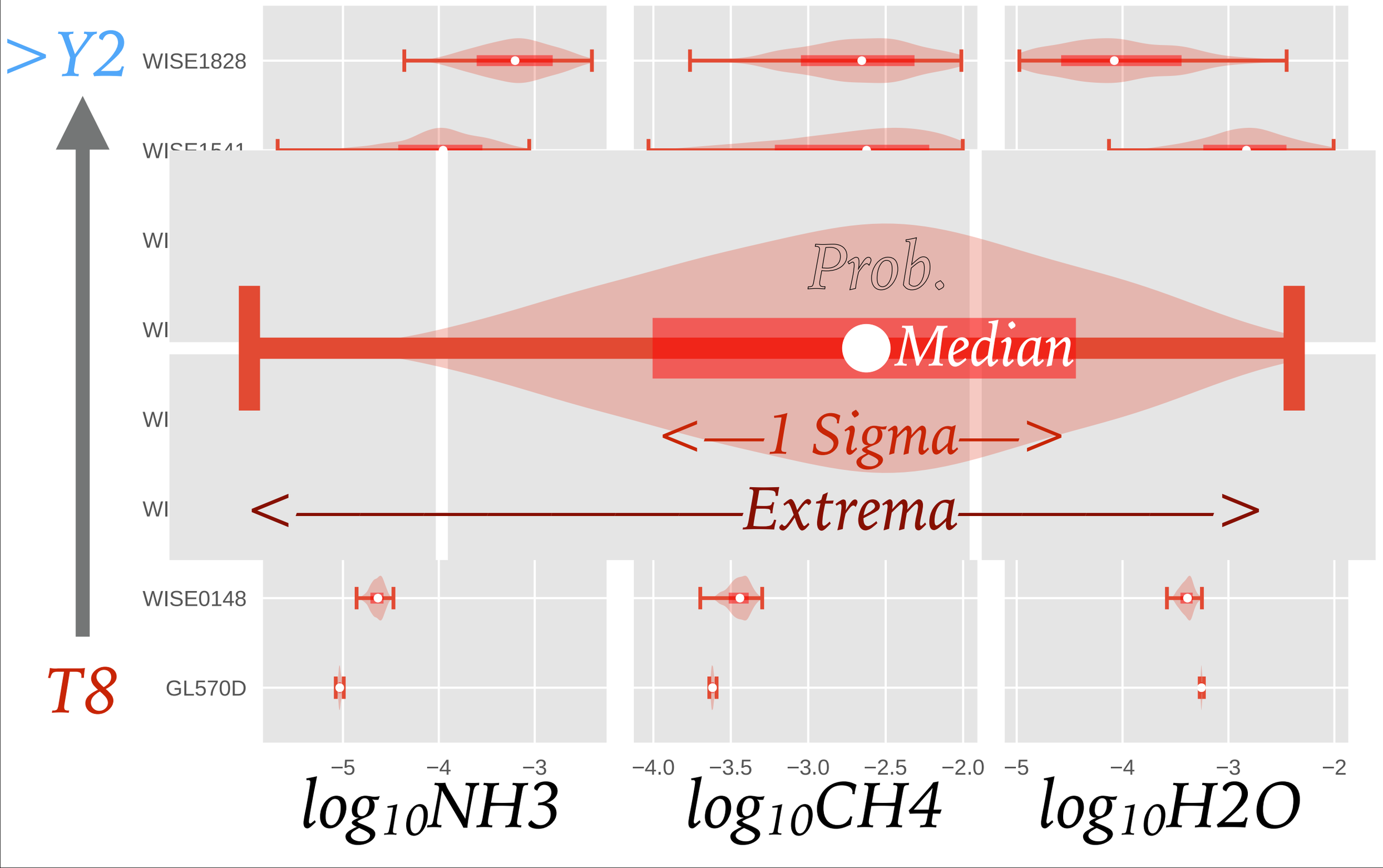
SPECTRAL FITS OF BROWN DWARFS

Data *Median*

1 Sig. *2 Sig.*



MULTINEST RESULTS: COMPARING POSTERIORS



MULTINEST USERGUIDE: INSTALLATION

1) *Download latest git version of radtrancode.*
2) *External: follow https://johannesbuchner.github.io/PyMultiNest/install.html*
3) *Internal: add my LD_LIBRARY_PATH to your .tcshrc file. Also add: module load pymods_trusty to .tcshrc.*
4) *Go to Nemesis subfolder in radtrancode, and type e.g.: source nemesisPyMult.comp to create the nemesisPyMult.so shared object file (basically a python library you can import). Do the same for nemesisPT_NS.comp, and any other versions you'd like.*
5) *Next is to use the compiled file!*

MULTINEST USERGUIDE: SETUP

1) *Create a normal Nemesis folder with all the relevant inputs, as for OE*
2) *Switch the number of iterations to -1, and set the .apr file to have zero parameters (*clouds are included in the .apr file, if you wish to have them)*
3) *Take these files, and duplicate them so that there are 64 folders (or however many cores you will use), numbered from 0-63. Also create a 999 folder, which is used for plotting.*
4) *In the directory containing all of these duplicate folders, copy the .so file from the Nemesis subfolder.*
5) *Setup your retrieval in python (see NS_ret.py for example usage) i.e. your forward model and associated hypercube*
6) *Run the retrieval (64 is core number, unnecessary text being redirected to trash) with: mpiexec -n 64 python NS_ret.py > /dev/null &*
7) *Either wait until the whole retrieval is finished and use NS_plot.py, or modify it slightly to watch convergence occur while the retrieval runs. The modification is done like so (note the triple quotes):*

```
#Get retrieval info

prefix = './chains/GL570D_'
a = pymultinest.Analyzer(n_params = n_params, outputfiles_basename = prefix)
s=a.get_stats()
values = a.get_equal_weighted_posterior()

#values=np.loadtxt('chains/GL570D_phys_live.points')
```

```
'''
prefix = './chains/GL570D_'
a = pymultinest.Analyzer(n_params = n_params, outputfiles_basename = prefix)
s=a.get_stats()
values = a.get_equal_weighted_posterior()
'''
values=np.loadtxt('chains/GL570D_phys_live.points')
```

MULTINEST USERGUIDE: **NOTES**

- *Temperature and VMRs are passed through with the same number of layers as in the .ref file.*
- *The VMRs must be in the same order as the .ref file. Must sum to 1.0*
- *Mass and radius are in terms of Jupiter, so a 1.0 radius is 69111km*
- *The aforementioned parameters are always passed, as well as pressure, and height (though NEMESIS recalculates height anyway, only the base height is important (0.0 km)).*

MULTINEST USERGUIDE: PARAMETERS

```
# run model
 model = nemesisPyMult.nemesispymult(runname, len(spec), len(temp), vmr.shape[1],
        ith, temp, P, vmr, mass, rad, H, Hb, opac, FSH, prad, pvar,
        len(pimag), pimag, preal, Hb2, opac2, FSH2,
        nav, flat, flon, solzen, emzen, azi, wt)
```

1. *runname: name of nemesis files, e.g. 'jupiter' for jupiter.ref.*
2. *len(spec) = length of spectrum, number of points in .spx file.*
3. *len(temp) = number of pressure points in .ref.*
4. *vmr.shape[1] = number of gases in .ref file.*
5. *ith = i^{th} directory to perform calculation in. Linked to which core is being used.*
6. *Temp = temperature profile*
7. *P = pressure profile*
8. *vmr = vmr profile for all gases*
9. *Mass = mass in jupiter masses*
10. *Rad = radius in jupiter radii*
11. *H = height profile*
12. *Hb = base height of cloud in km (used for cloud parameterisation #9)*
13. *Opac = integrated optical depth at specified wavelength (#9)*
14. *FSH = fractional scale height (#9)*
15. *prad = particle radius in microns (cloud param #444)*
16. *pvar = particle size variance in microns (cloud param #444)*
17. *len(pimag) = number of imaginary indices from refractive index*
18. *Pimag = imaginary refractive index spectrum*
19. *PReal = real refractive index at specified wavelength*
20. *Hb2 = second cloud #9*
21. *opac2*
22. *FSH2*
23. *Nav = number of averaging weights in .spx file*
24. *Flat = latitudes of weights*
25. *Flon = longitude of weights*
26. *Solzen = solar zenith angle*
27. *Emzen = emission zenith*
28. *Azi = azimuthal angle*
29. *Wt = weights*

MULTINEST USERGUIDE: PARAMETERS

```
# run model
 model = nemesisPyMult.nemesispymult(runname, len(spec), len(temp), vmr.shape[1],
        ith, temp, P, vmr, mass, rad, H, Hb, opac, FSH, prad, pvar,
        len(pimag), pimag, preal, Hb2, opac2, FSH2,
        nav, flat, flon, solzen, emzen, azi, wt)
```

1. *runname: name of nemesis files, e.g. 'jupiter' for jupiter.ref.*
2. *len(spec) = length of spectrum, number of points in .spx file.*
3. *len(temp) = number of pressure points in .ref.*
4. *vmr.shape[1] = number of gases in .ref file.*
5. *ith = i^{th} directory to perform calculation in. Linked to which core is being used.*
6. *Temp = temperature profile*
7. *P = pressure profile*
8. *vmr = vmr profile for all gases*
9. *Mass = mass in jupiter masses*
10. *Rad = radius in jupiter radii*
11. *H = height profile*

Compulsory parameters

MULTINEST USERGUIDE: **PARAMETERS**

```
# run model
 model = nemesisPyMult.nemesispymult(runname, len(spec), len(temp), vmr.shape[1],
        ith, temp, P, vmr, mass, rad, H, Hb, opac, FSH, prad, pvar,
        len(pimag), pimag, preal, Hb2, opac2, FSH2,
        nav, flat, flon, solzen, emzen, azi, wt)
```

Cloud parameters are read in IF the appropriate .apr parameters have been inserted. If .apr is empty, then any placeholder values can be used as the clouds are not included in the retrieval.

12. *Hb = base height of cloud in km (used for cloud parameterisation #9)*
13. *Opac = integrated optical depth at specified wavelength (#9)*
14. *FSH = fractional scale height (#9)*
15. *prad = particle radius in microns (cloud param #444)*
16. *pvar = particle size variance in microns (cloud param #444)*
17. *len(pimag) = number of imaginary indices from refractive index*
18. *Pimag = imaginary refractive index spectrum*
19. *PReal = real refractive index at specified wavelength*
20. *Hb2 = second cloud #9*
21. *opac2*
22. *FSH2*

```
******** any header info you like. One line only ********
1                       ! number of variable profiles (vmr,T, or c
-1 0 9                  ! cloud variable 1
12.95 12.95
5.24355 5.24355
0.5 0.5
```

```
******** any header info you like. One line only ********
3                       ! number of variable profiles (vmr,T, or cont)
-1 0 9                  ! cloud variable 1
12.95 12.95
5.24355 5.24355
0.5 0.5
444 1 444
cloud.dat
-2 0 9                  ! cloud variable 1
12.95 12.95
5.24355 5.24355
0.5 0.5
```

MULTINEST USERGUIDE: **PARAMETERS**

```
# run model
 model = nemesisPyMult.nemesispymult(runname, len(spec), len(temp), vmr.shape[1],
        ith, temp, P, vmr, mass, rad, H, Hb, opac, FSH, prad, pvar,
        len(pimag), pimag, preal, Hb2, opac2, FSH2,
        nav, flat, flon, solzen, emzen, azi, wt)
```

If nav = 0, then .spx file is used normally. Otherwise, you can put in your own averaging weights.

23. *Nav = number of averaging weights in .spx file*
24. *Flat = latitudes of weights*
25. *Flon = longitude of weights*
26. *Solzen = solar zenith angle*
27. *Emzen = emission zenith*
28. *Azi = azimuthal angle*
29. *Wt = weights*

MULTINEST USERGUIDE: **EXAMPLES**

Examples are given in ~garland/NS_EXAMPLES/

I suggest you follow the installation instructions, copy the BASIC_BD_RET example somewhere, copy in your new .so file, and then run as per instructions. Check out the plots from NS_plot, and see if they look sensible!