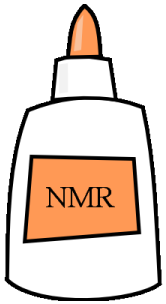
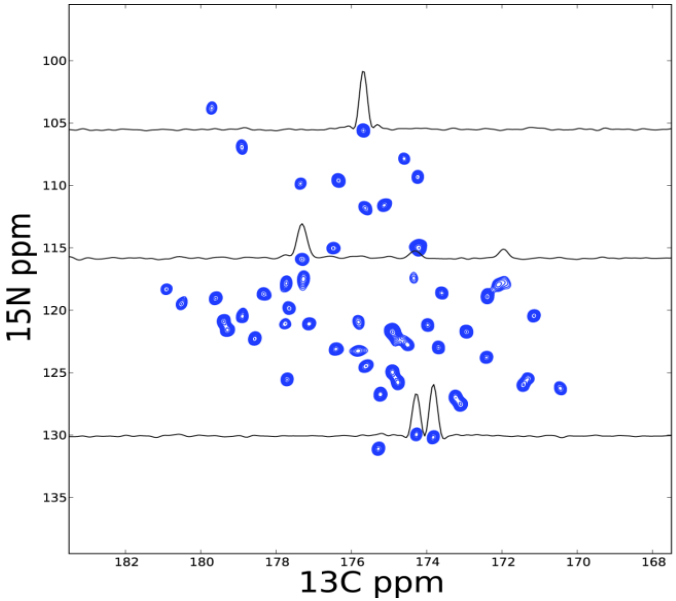
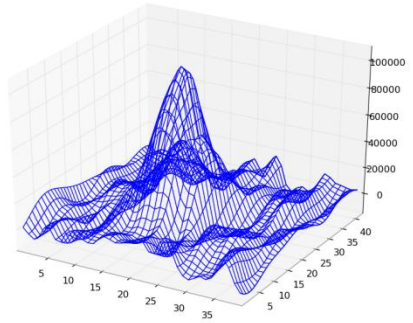
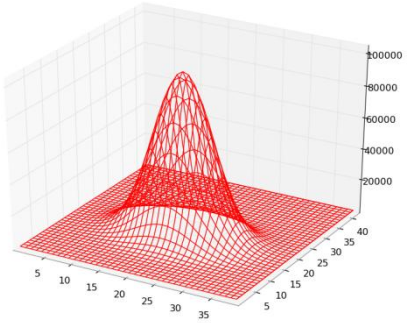
nmrglue: a Python Module for Working with NMR Data









Jonathan J. Helmus, Ph.D.
Molecular, Microbial and Structural Biology Department
UConn Health Center, Farmington, CT

2012 SciPy Conference, July 19th, 2012 Austin, TX

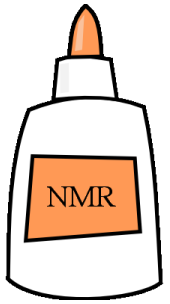
Overview

nmrglue is an open source Python module for working with NMR data which acts as the "glue" to tie together existing NMR programs, and can be used to rapidly develop new NMR processing, analysis or visualization methods.

NMR = Nuclear Magnetic Resonance

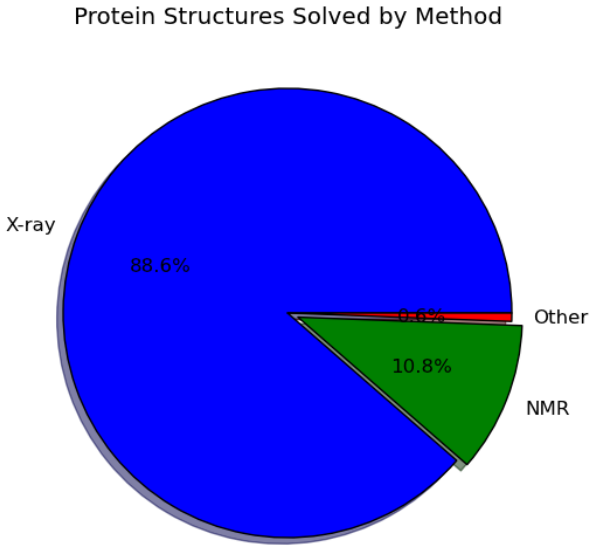
This talk will discuss:

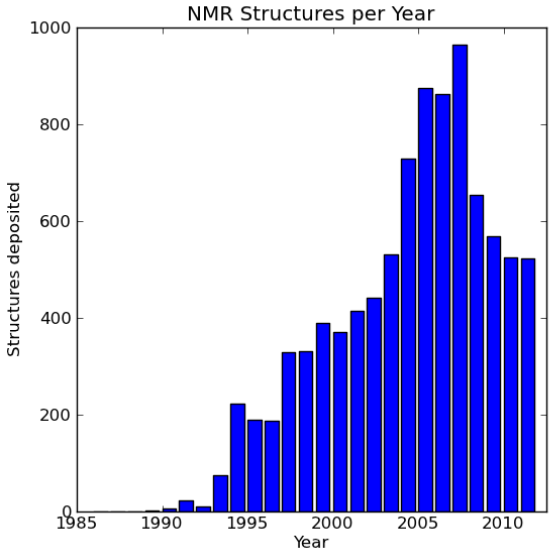
- A short background on NMR and protein NMR
- Why I wrote nmrglue.
- The design and features of nmrglue.
- A few examples.
- Opportunities for code reuse.



NMR - Introduction

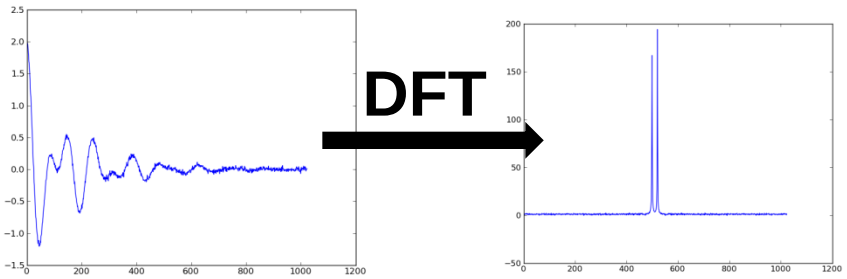
- Nuclear magnetic resonance (NMR) spectroscopy is a key analytical technique used to characterize the **structure and dynamic** properties of chemicals.
- The most common use of NMR spectroscopy is for the non-destructive identification of small organic molecules based on ^{1}H and/or ^{13}C spectra.
- NMR has also found uses in the biomedical field; being used in drug discovery, metabolomics, and imaging as well as being the primary method for the determination of the structures of biological macromolecules in solution.
- NMR is the second most used experimental method for solving protein structures, accounting for over 8,000 of the over 76,000 protein structure in the Protein Data Bank.





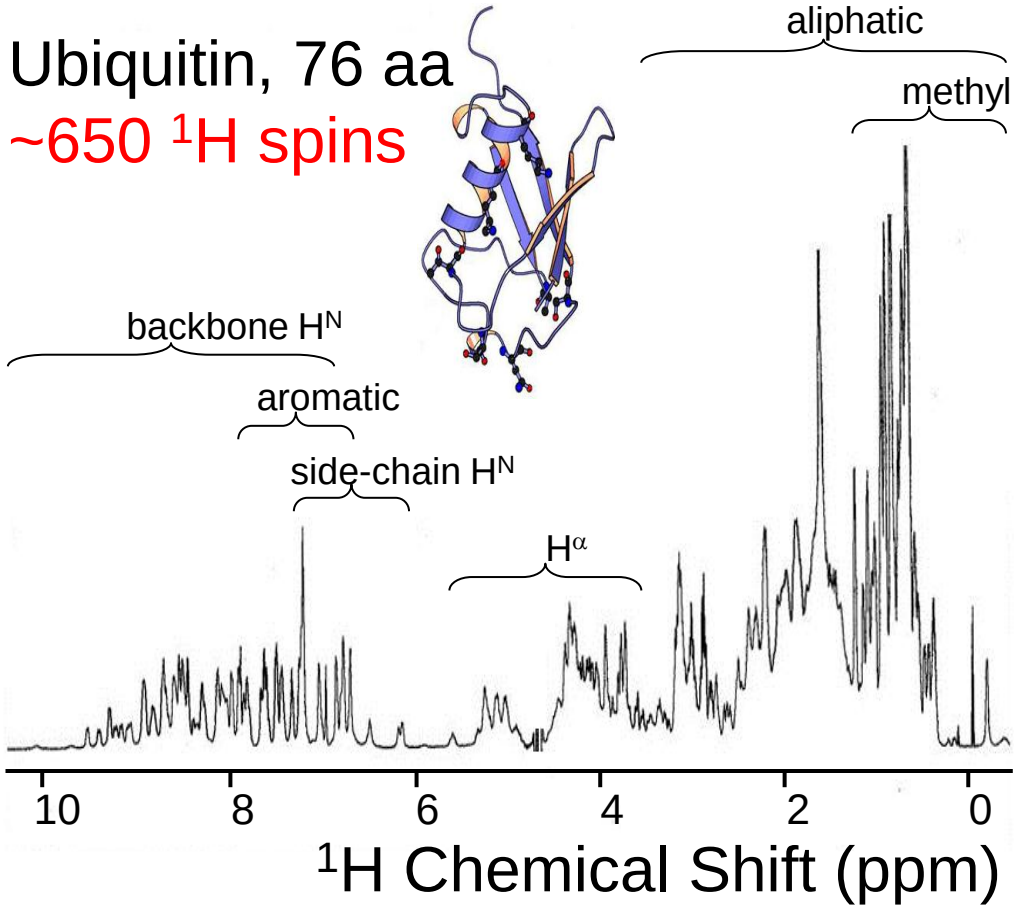
NMR – How it works

1. A sample is prepared with **labeled isotopes**.
2. The sample is placed in a **large magnetic field**. The nuclei align along the magnetic field.
3. Precisely timed **radio frequency pulses** are applied to manipulate the nuclear spins.
4. The **time domain response** to these pulses is collected, the free induction decay (FID).
5. This time domain signal is converted to the frequency domain, a spectrum, using a **discrete Fourier transform**, or another signal processing methods.
6. The spectrum is **visualized** and peaks are **assigned .**
7. Peaks are **analyzed** for their linewidths, heights, volumes, etc from which structural or dynamics parameters can be determined.



Protein NMR

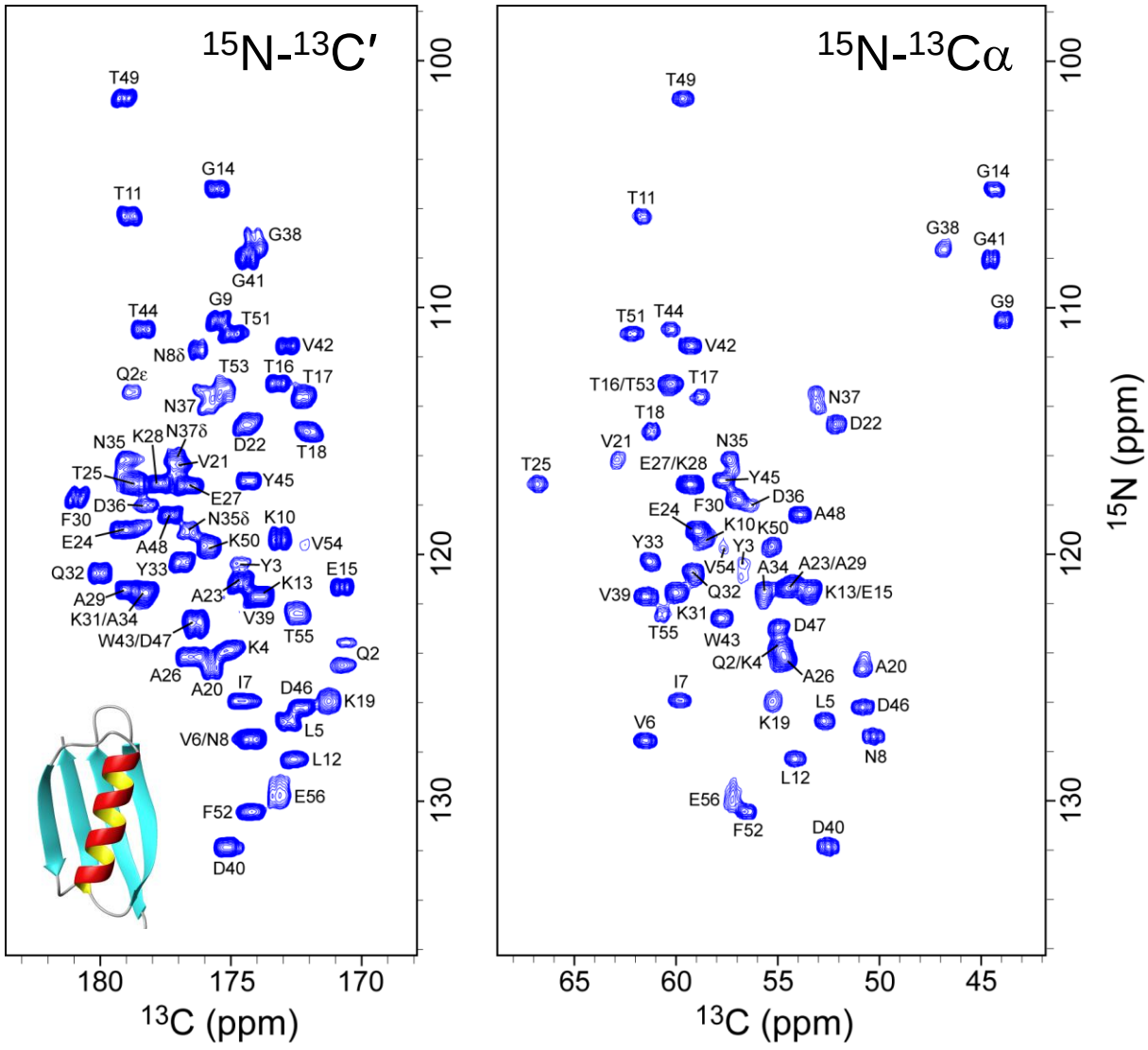
- Proteins are the primary actors in cells and participate in nearly all processes with a cell.
- Proteins are made up of a linear chain of amino acids.
- 20 amino acids which compose nearly all proteins, which themselves are composed mostly of hydrogen, carbon, and nitrogen and atoms.
- Proteins with NMR active isotopes of hydrogen (^{1}H), carbon (^{13}C) and nitrogen (^{15}N) can be readily expressed in both solution and as solids.
- A key challenge in protein NMR is to obtain spectra with sufficient resolution to resolve individual peaks.



Multidimensional Protein NMR

- Additional resolution can be obtained by separating peaks along a 2nd dimension.
- 2D, 3D and even 4D protein NMR experiments are now common.
- These multidimensional data sets can be large in size. ~1-2 GB 3D data sets are common and 4D data sets can be even larger.

2D ^{15}N-^{13}Cα/^{13}C′ Correlation Spectra



Nadaud, Helmus & Jaroniec, *Biomol. NMR Assign.* 2007

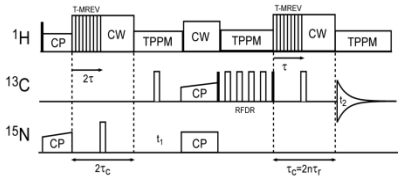
NMR: an abundance of programming opportunities

The NMR field is rich with interesting programming problems. To address these problems a plethora of software packages exist.

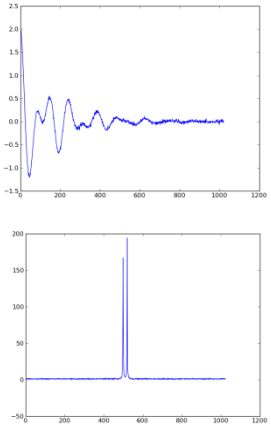
Collection → Processing → Visualization Assignment → Analysis

Microcontrollers

RF pulse programming



Signal processing



2D and 3D plotting

Automated assignment

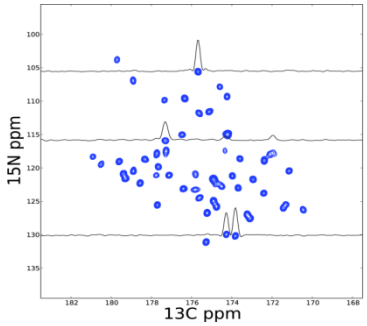
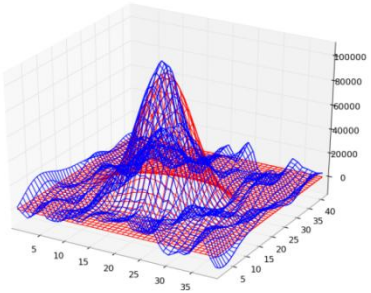


Image processing

Optimization and curve fitting.



<u>Software</u>
VNMR
VnmrJ
XWINNMR
Topspin

<u>Software</u>
NMRPipe
The Rowland NMR Toolkit
ACD/NMR

<u>Software</u>
NMRDraw
Sparky
NMRView
CcpNmr Analysis

<u>Software</u>
NMRPipe
relax
REDCAT
TALOS

Why another NMR software package?

Software is abundant in the NMR field. Commercial businesses, university research labs and government institutes already provide a number of software packages to collect, process, visualize and analyze NMR data. So why write more NMR software?

- The lack of a standard format for storing NMR data has lead to a **number of incompatible NMR file formats**. We need a way to access and interconvert between these formats.

- NMR is an active research area; new experiments and methods are constantly being developed. Scientists need software which can be **adapted** for these new methods. Only a handful of NMR software packages provide direct access to the NMR data in a scripting or macro language and only one in a standard language (MatNMR, Matlab).

- Protein NMR by it's nature is **repetitive**, the same analysis is performed on each of the 100 to 1000's of peaks in a given spectra. Software needs to be **automated**.

- Very few of the software packages are **open source**. Despite using well established algorithms there are few if any available implementations. This results in many programmers having to "**recreate the wheel**".

Design of nmrglue

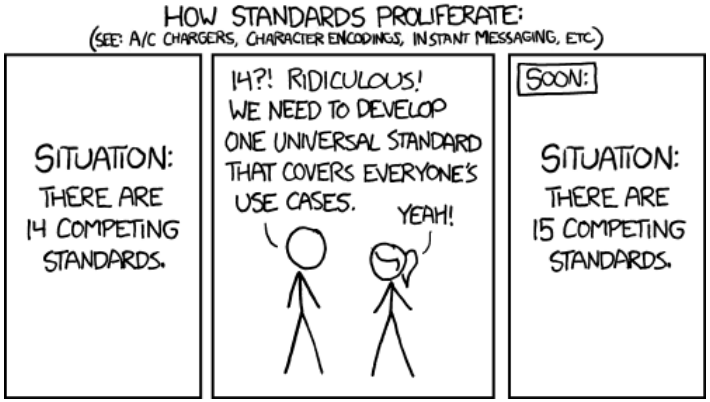
nmrglue aims to be:

…powerful yet easy to install and use.

- The high-level interpreted Python language offers clear and expressive syntax.
- Numpy and Scipy offer powerful mathematical and scientific functions in Python.
- Installation of Python and libraries are easy on Windows, OS-X and Linux.
- Ample tutorials and documentation already exists for Python and these libraries.

… a system for interconnecting existing NMR programs.

- Care is taken NOT to create a new standard for storing NMR data.
- can read and write directly from a number of NMR file formats
- NMR data is stored as numpy ndarray allowing easy and robust manipulation
- Spectra parameters are stored as dictionaries.



http://xkcd.com/927/

… an environment for rapidly developing new processing, analysis and visualization methods.

- Common NMR processing and analysis routines are provided in nmrglue
- New routines can be built from numpy, scipy, matplotlib or other Python modules.
- Ipython offers a wonderful environment for interactive examination and visualization of data.

nmrglue module overview

Module	Description
fileio	
bruker	Reading/writing of Bruker files.
pipe	Reading/writing of NMRPipe files.
rnmrtk	Reading/writing of Rowland NMR Toolkit files.
sparky	Reading/writing of Sparky files.
varian	Reading/writing of Agilent/Varian files.
convert	Convert between any of the above formats.

- All of these routines also offer a method for “on-demand” or “low-memory” reading of the various file formats.

processing	
proc_base	Common NMR processing functions (apodization, shifts, transforms, …)
proc_bl	Baseline filtering, smoothing and correcting functions.
proc_lp	Linear prediction modeling and extrapolation
pipe_proc	NMRPipe like processing functions.

analysis	
peakpick	Numerous peak picking algorithms which work in arbitrary dimensions.
linesh	Fitting and simulation of arbitrary dimensional lineshapes.

File operations example

nmrglue can read and write a number of common NMR file Spectral data are stored in memory as a numpy ndarray object spectral parameters in a dictionary.

Reading in a 2D time domain NMRPipe file and examining the data in the python shell:

```
$python
>>> import nmrglue as ng
>>> dic, data = ng.pipe.read("test.fid")
>>> data.ndim
2
>>> data.shape
(332, 1500)
>>> data.dtype
dtype('complex64')
>>> dic["FDF2SW"]
50000.0
>>> dic["FDF1LABEL"]
'15N'
>>> data[2,100]
(-9.9586811+69.388367j)
```

This data can be modified in memory and then written to a new file. For example setting first trace of the above 2D to zero and writing to "new_file.fid":

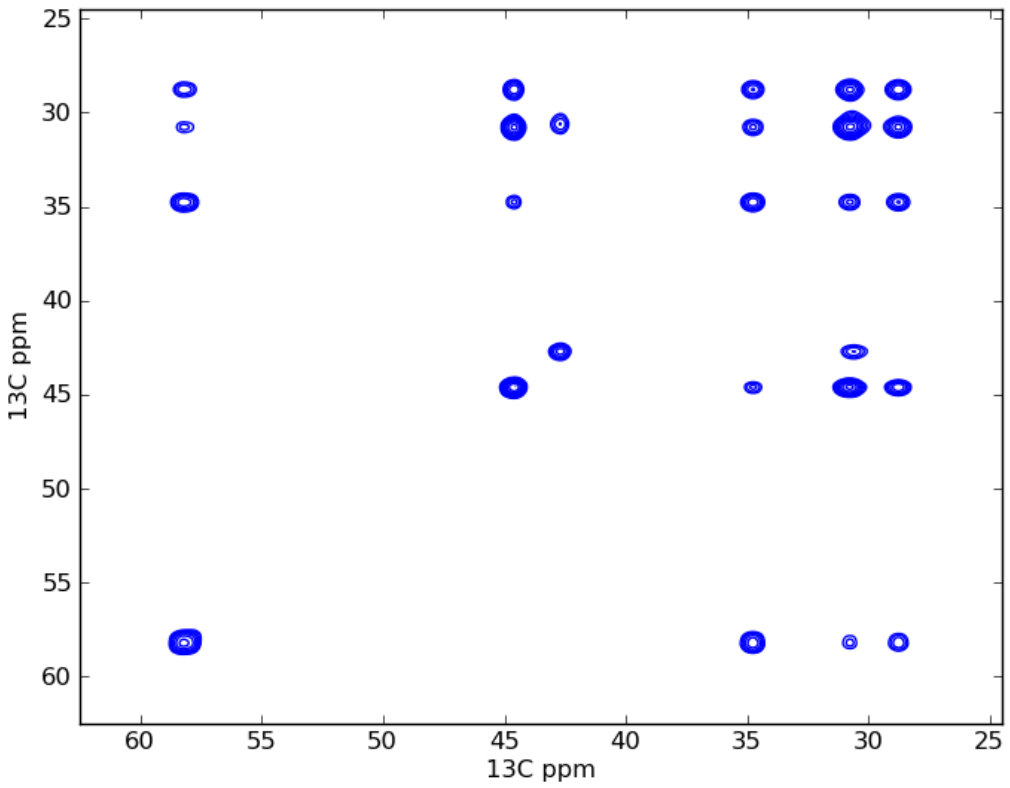
```
>>> data[0] = 0.0
>>> data[0]
array([ 0.+0.j,  0.+0.j,  0.+0.j, ...,  0.+0.j,
0.+0.j,  0.+0.j], dtype=complex64)
>>> ng.pipe.write("new_file.fid", dic, data)
```

nmrglue can also be used to convert between file formats using the convert module. For example to convert a 2D frequency domain NMRPipe file to a Sparky file:

```
>>> dic,data = ng.pipe.read("test.ft2")
>>> C = ng.convert.converter()
>>> C.from_pipe(dic, data)
>>> sparky_dic, sparky_data = C.to_sparky()
>>> ng.sparky.write("sparky_file.ucsf",
       sparky_dic, sparky_data)
```

Processing example

nmrglue can be used in conjunction with Numpy to perform covariance processing[1] on 2D ^{13}C-^{13}C NMR data of bolaamphipile **A** nanotubes[2]. This non-standard processing method is not supported by other processing packages, but can easily be performed in Python.



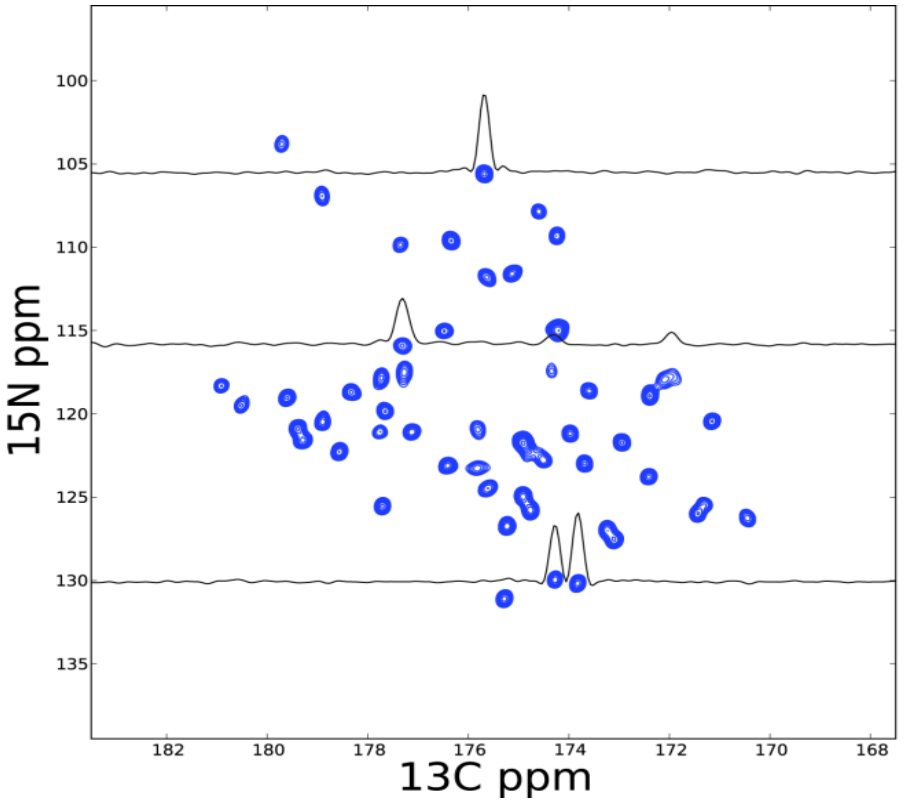

```python
import nmrglue as ng
import matplotlib.pyplot as plt
import numpy as np

# open the data
dic, data = ng.pipe.read("test.ft")

# compute the covariance
C = np.cov(data.T)

# plot the spectrum
uc = ng.pipe.make_uc(dic, data, 1)
x0, x1 = uc.ppm_limits()
fig = plt.figure()
ax = fig.add_subplot(111)
cl = [1e8 * 1.30 ** x for x in range(20)]
ax.contour(C,cl,colors='blue',
    extent=(x0, x1, x0, x1), origin=None)
ax.set_xlabel("13C ppm")
ax.set_xlim(62.5, 24.5)
ax.set_ylabel("13C ppm")
ax.set_ylim(62.5, 24.5)
fig.savefig("spectrum.png")
```


1) R. Bruschweiler and F. Zhang, *J. Chem. Phys.*, **2004**, 120, 5253.
2) H. Shao, *et al.* Angew. *Chem. Int. Ed.* **2010**, 49, 7688.

Visualization example.

```
import nmrglue as ng
import matplotlib.pyplot as plt

# read in the data from a NMRPipe file
dic, data = ng.pipe.read("test.ft2")

# create a unit conversion object for each axis
uc_15n = ng.pipe.make_uc(dic, data, 0)
uc_13c = ng.pipe.make_uc(dic, data, 1)

# find the ppm limits
x0, x1 = uc_13c.ppm_limits()
y0, y1 = uc_15n.ppm_limits()

# plot the spectrum
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111)
cl = [8.5e4 * 1.30 ** x for x in range(20)]
ax.contour(data, cl, colors='blue',
   extent=(x0, x1 ,y0 ,y1), origin=None)

# add 1D slices
x = uc_13c.ppm_scale()
s1 = data[uc_15n("105.52ppm"), :]
s2 = data[uc_15n("115.85ppm"), :]
s3 = data[uc_15n("130.07ppm"), :]
ax.plot(x, -s1 / 8e4+105.52, 'k-')
ax.plot(x, -s2 / 8e4+115.85, 'k-')
ax.plot(x, -s3 / 8e4+130.07, 'k-')

# label the axis and save the figure
ax.set_xlabel("13C ppm", size=20)
ax.set_xlim(183.5, 167.5)
ax.set_ylabel("15N ppm", size=20)
ax.set_ylim(139.5, 95.5)
fig.savefig("spectrum.png")
```

Matplotlib can be used to generate publication quality plots of NMR spectra using nmrglue.



Analysis example

Extracting relaxation trajectories for a series of 2D NMR spectra. Summation over rectangular boxes are used for the peak volume due to the poor lineshapes typically found in solid-state spectra. Trajectories can be fit to a relaxation model or graphed in an additional scripts

```
# read in the integration limits and list of spectra
peak_list = np.recfromtxt("boxes.in")
spectra_list = np.recfromtxt("spectra.in")

# prepare the trajs records array
num_spec = spectra_list.size
num_peaks = peak_list.size
elist = [np.empty(num_spec, dtype="float") for i in xrange(num_peaks)]
trajs = np.rec.array(elist, names=list(peak_list.f0))

# loop over the spectra
for sn,spectra in enumerate(spectra_list):
    # read in the data from a NMRPipe file
    print "Extracting from:", spectra
    dic, data = ng.pipe.read(spectra)

    # loop over the integration limits
    for name, x0, y0, x1, y1 in peak_list:
        # integrate the region and save in record array
        if x0 > x1:
             x0, x1 = x1, x0
        if y0 > y1:
             y0, y1 = y1, y0
        trajs[name][sn] = data[y0:y1 + 1, x0:x1 + 1].sum()

# normalize each trajectory
for peak in trajs.dtype.names:
    trajs[peak] = trajs[peak] / trajs[peak].max()

# save the trajectories records array to disk
np.save("traj.npy",trajs)
```

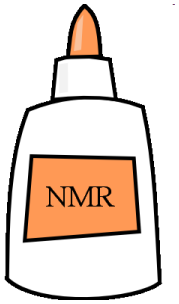
```
#Peak     X0      Y0      X0      Y1
A20      4068    938     4079    913
A24      3992    1013    4000    997
A26      4065    962     4075    940
…
```

```
data/Ytau_100.fid/test.ft2
data/Ytau_100000.fid/test.ft2
data/Ytau_250000.fid/test.ft2
…
```

Opportunities for code reuse in nmrglue

Routines in nmrglue which may be of interest to the wider scientific community:

- On demand / “low memory” file access.
- Constrained multivariate least-squares optimization.
- N-dimensional peak picking
- N-dimensional lineshape fitting and simulations



Code available at github.com/jjhelmus (BSD licensed)

On demand (“low memory”) file access

Often in NMR (and other fields) large data sets are created, but only a small region of the data is needed for processing, visualization or analysis. Reading the whole data set into memory may be time consuming or resource intensive and is often unnecessary.

nmrglue offers data_nd objects which are similar to ndarrays but data is only read from disk when a slice is requested.

- Slicing returns ndarray objects. Many slicing methods are supported.
- ndim, shape, dtype attributes.
- Iteration is supported.
- swapaxes and transpose methods return copies with the new axes ordering.

- Easy to adapt, three methods must be defined, __init__, __fcopy__ and__fgetitem__
- __fgetitem__ is passed a well formed tuple of slice objects and must read in the data from file and return an array, all transposing, reversing, etc. is taken care of!

```
>>> dic,data = ng.pipe.read_lowmem(“test%03d.ft3")
>>> type(data)
<class 'nmrglue.fileio.pipe.pipe_3d'>
>>> data.shape
(64, 64, 4096)

>>> data[0,0,1]       # data is read in from file here
array(-2827.07861328125, dtype=float32)
>>> data[2,2,:]
array([ 16753.04296875,    192.06381226,
-5558.41552734, ...,
```

```
>>> tdata = data.tranpose()
>>> tdata.shape
(4096, 64, 64)
>>> data[1,0,0]
array(-2827.07861328125, dtype=float32)
```

See nmrglue/fileio/fileiobase.py for details

Bounded least squares optimization

Many of the optimization functions in scipy support constraints. The Levenberg–Marquardt / Least Squares optimization function (scipy.optimize.leastsq) does not. A desire for a bounded version of this function has been discussed on the mailing list and a number of options outside of scipy exist (lmfit-py, mpfit, an open pull request, …). For nmrglue I wanted a version which was a drop-in replacement for Scipy's leastsq.

nmrglue's leastsq_bound function:

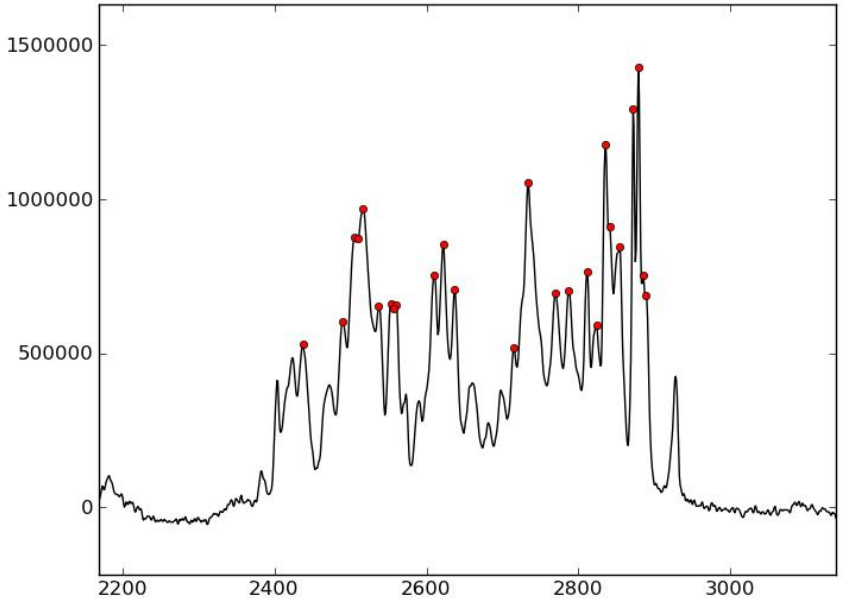
- Identical call and return structure as scipy's leastsq with the addition of a bound parameter.
- Bound parameter can be None of (min, max) for each variable being optimized. Min or max can be None.
- Restraints imposed by variable substitution using functions (sin and sqrt) which have self-imposed limits. This method is used by MINUIT to introduce constraints[1].
- Available as a separate package: github.com/jjhelmus/leastsqbound-scipy
- Still needs some work to be a full replacement for leastsq: factor and scale parameters should effect on original variables as should infodic['qtf'].

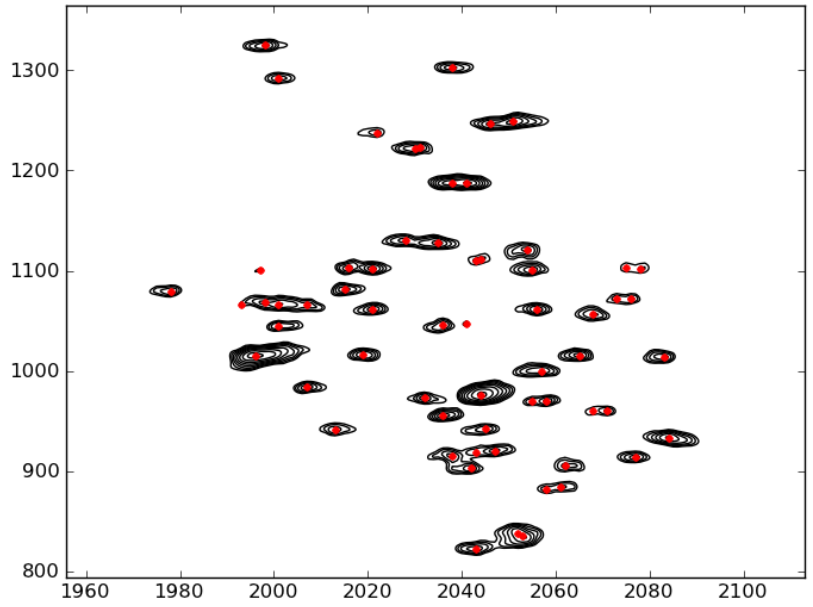
[1]F. James and M. Winkler. MINUIT User's Guide, July 16, 2004.

N-dimensional peak picking

nmrglue supports peak picking with its **peakpick.pick** function.

- Peaks can be picked in a 1-D, 2-D, … N-dimensional ndarray.
- Both positive and negative peaks can be detected.
- Three picking algorithms can be used; threshold/minimum distance, connected, or downward connected.
- Peak amplitudes and linewidth can be estimated.
- Function can also segment the spectrum into connected regions.
- Details in analysis/peakpick.py and analysis/segmentation.py



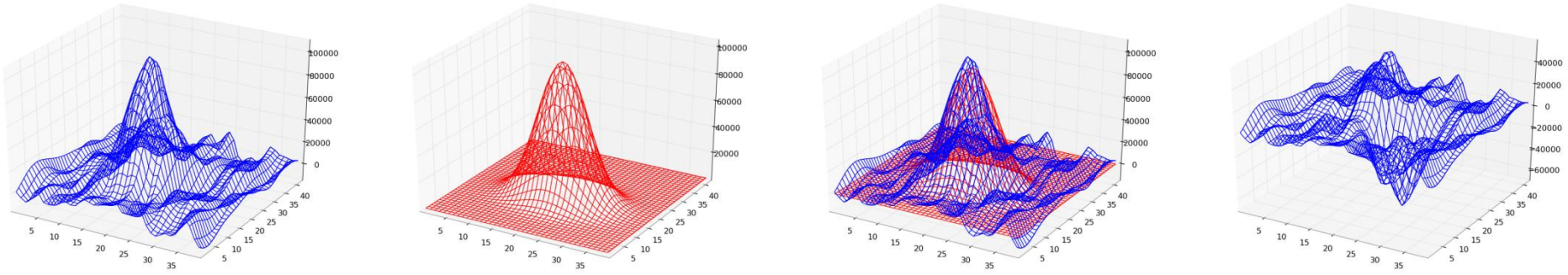


N-dimensional lineshape fitting

Peaks can be fit to common peak lineshapes using the **linesh.fit_spectrum** or **linesh.fit_NDregion** functions.

- 1-D, 2-D, … N-dimensional regions can be fit and simulated.
- Multiple peaks can be fit simultaneously.
- Constraints can be placed on the peak parameters (uses leastsqbound)
- Custom lineshapes can be defined.
- Details in the analysis/linesh.py

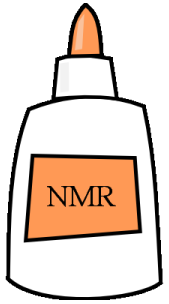
Regions containing one or more peaks can also be simulated with the above criteria



Additional information

- **nmrglue tutorial**
  - Introduces many of the most useful features of nmrglue.
  - Assumes some familiarity with Python and NMR.

- **Reference Guide**
  - Full listing of all functions and classes within nmrglue.
  - Closely follows the Numpy/Scipy doc standards.
  - Automatically build using Sphinx.



- **Numerous examples.**

- **Mailing List.**

- **Install files for Window, OS-X and Linux.**

- **Source code (BSD Licensed)**

http://code.google.com/p/nmrglue

Future Directions and Acknowledgments

Future directions for nmrglue

Support for additional file formats

- SIMPSON
- NMRView
- XEASY
- ???

Allow on demand / “low memory” writing (writing to a subsection) of a file.

Functions for generating common NMR plots.

A more complete tutorial which includes discussion of analysis functions.

More examples.

Special thanks go to my Ph.D. advisor, Dr. Christopher Jaroniec for allowing me to spend part of my graduate career developing nmrglue, and to Dr. Jeff Hoch, my present advisor, for allowing me to continue the development during a Post Doc at UConn.

This research was supported by the National Science Foundation (CAREER Award MCB-0745754 to C.P.J.) and by The Ohio State University Presidential Fellowship.