

Highly Configurable Analysis:

Lessons From A Decade In Astrophysics

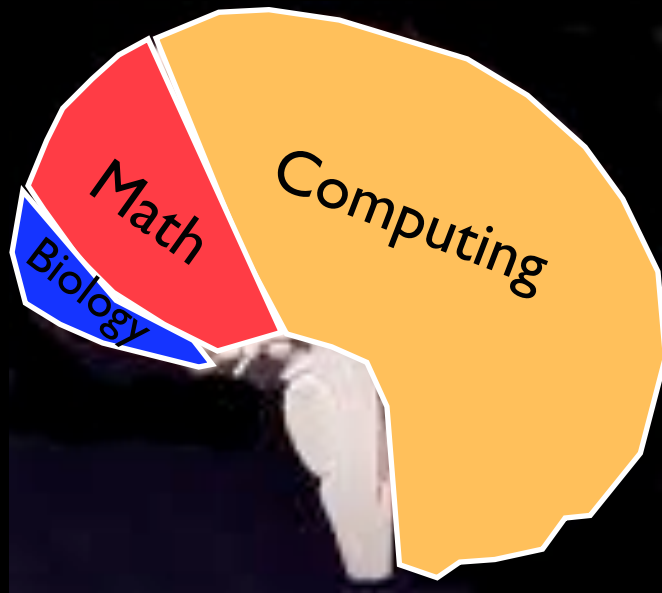
Michael S. Noble
Kavli Institute For Astrophysics
M.I.T.
June 8, 2010

Observation: All Science Is Interdisciplinary

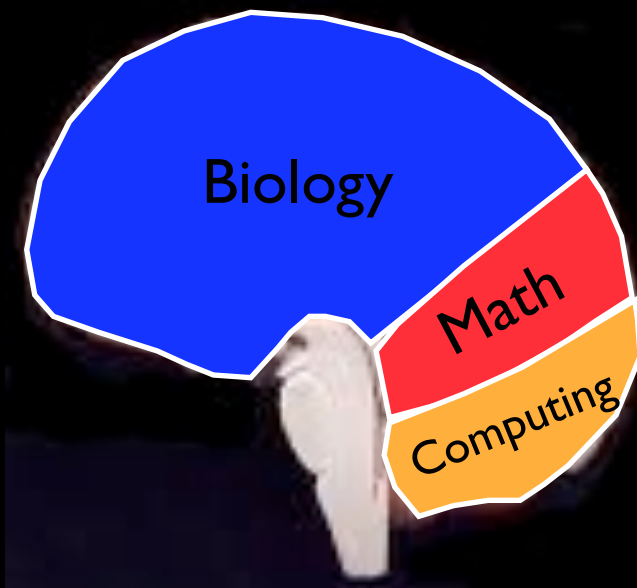
Newton inextricably linked science with math

Turing inextricably linked science with computers

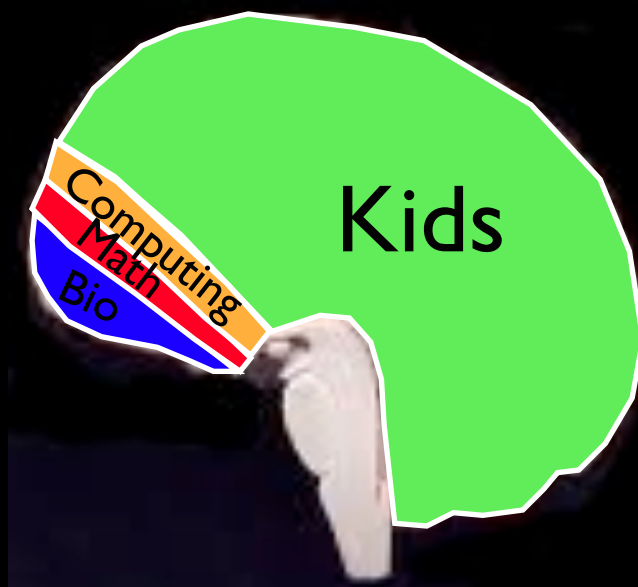
∴ Biologists will write math & code



When Coding
Or Data
Exploration
Is Hard



When
Easier



But Most Parents
Will Say This
Is More Accurate

Seem Familiar?


We all work hard,
and have good intentions.

So, why does scientific data
exploration & software
sometimes feel harder
than it need be?


A Tale of Two Coders

Software Engineer

Researcher



Careful, deliberate design
Towards production deployment
Must be fastidious



Exploratory, open-ended analysis
Towards publication
Can be messy

Overlapping, But Not Identical, Aims

A Tale of Two Coders

Software Engineer

Researcher

Careful design

Speed of obtaining result

Compiled/Binaries: C/C++/Java/etc

Interpreted Scripts: compiling nuisance

Static typing catches bugs earlier

Variable declaration a nuisance

Robust / predictable / clarity

Intuitive / flexible / contextual

OO is best

Not if mandatory for common/simple case

Creating new S/W
is most fun/rewarding

Prefers trusted methods, vetted by
years of research / publication use

Novice in science domain?

Novice in CS domain?

Version control, Makefiles, Regression tests

What?

Classic Case : new / malloc Errors

ENOMEM is Fatal

```
isis> load_data
```

```
isis> fit_model
```

...WAIT 2 DAYS ...

```
isis> volview(3D_gas_cloud)
```

Out Of Memory!
Aborted

```
linux%
```

assert() or exit() on error?

Not Necessarily

```
isis> load_data
```

```
isis> fit_model
```

...WAIT 2 DAYS ...

```
isis> volview(3D_gas_cloud)
```

Out Of Memory:
Close applications and retry.

```
isis> save_results
```

```
isis> volview(3D_gas_cloud)
```

Not if I lose days of analysis!

Human Context Trumps Dogma

What to do?

Claim

Given that science pushes boundaries of knowledge

And of technology to acquire / distill / analyze it

Scientific analysis S/W is never truly complete

Claim II

In our best interest, then, that it be EXTENSIBLE

To Enable

Researchers to incorporate custom/experimental codes

And drive system in directions not yet supported

Or perhaps even envisioned by its creators

Without programmer-army rewrite every few years

Claim III

Simply, but WLOG / power / flexibility

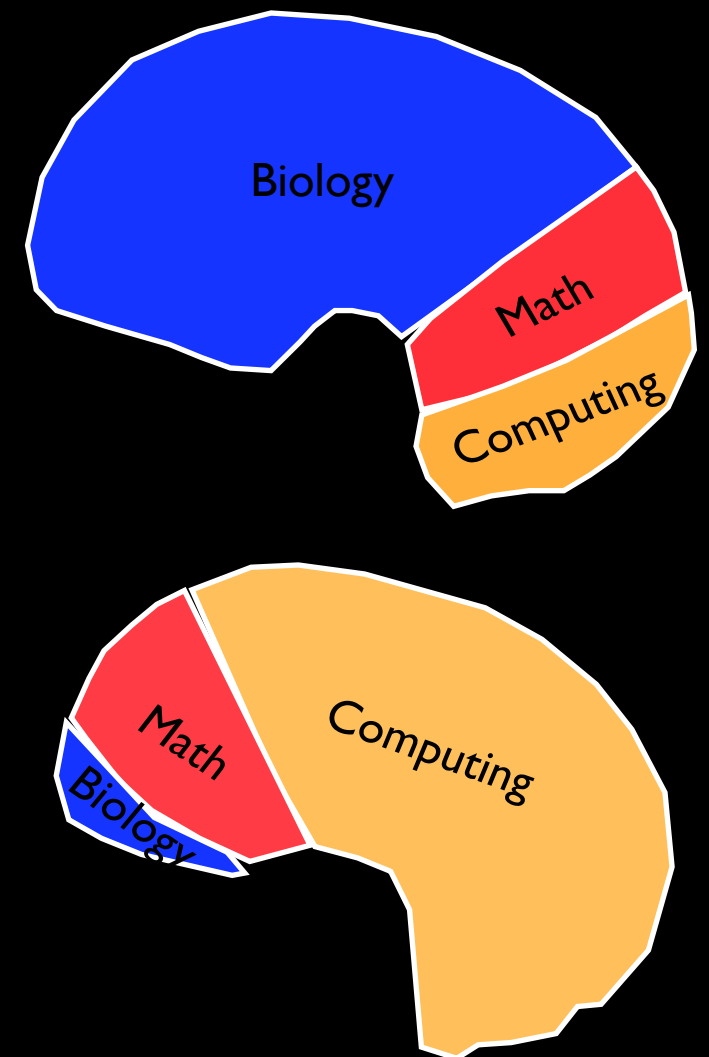
Freeing practitioners to concentrate on

Scientific and algorithmic concerns

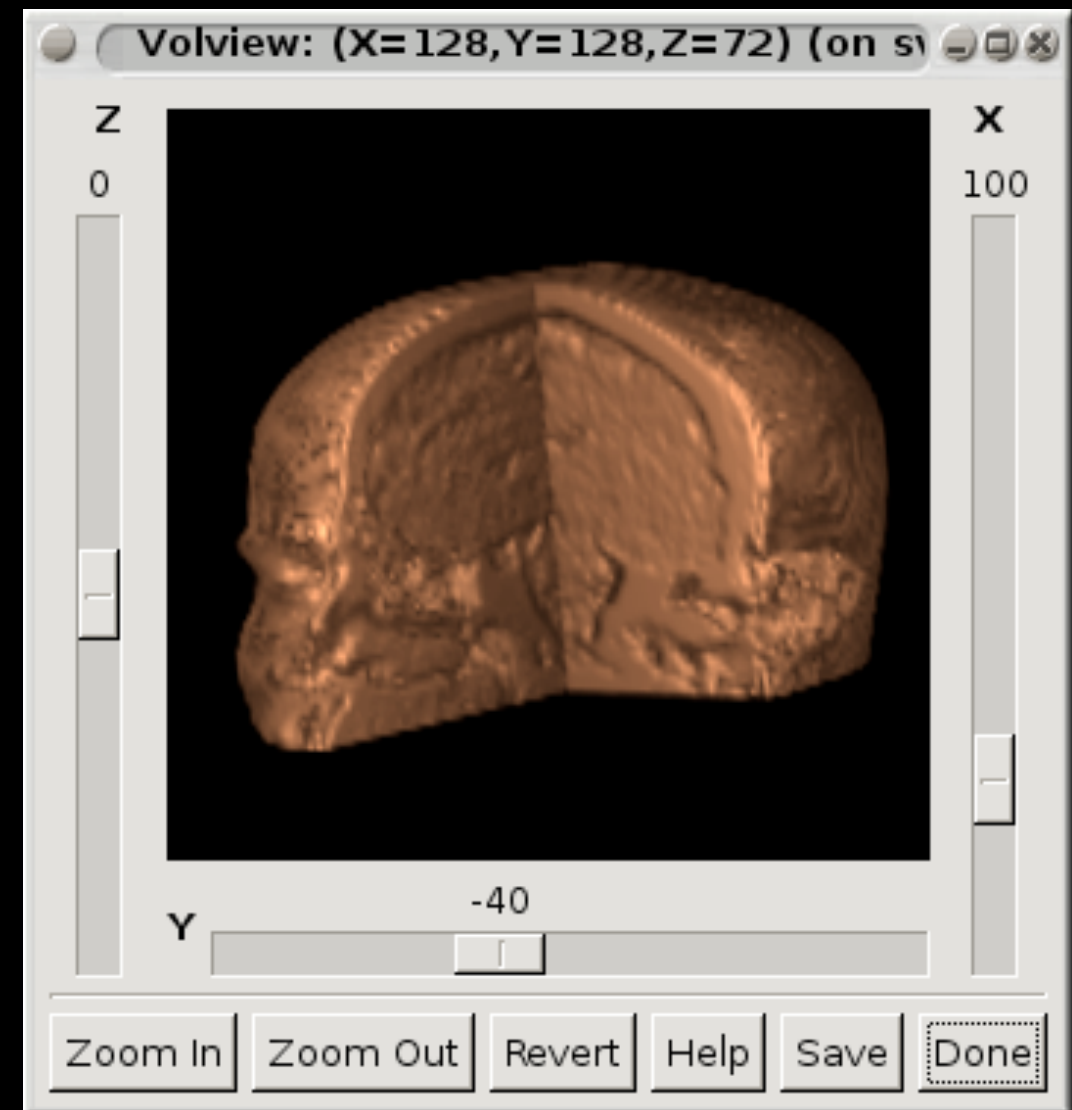
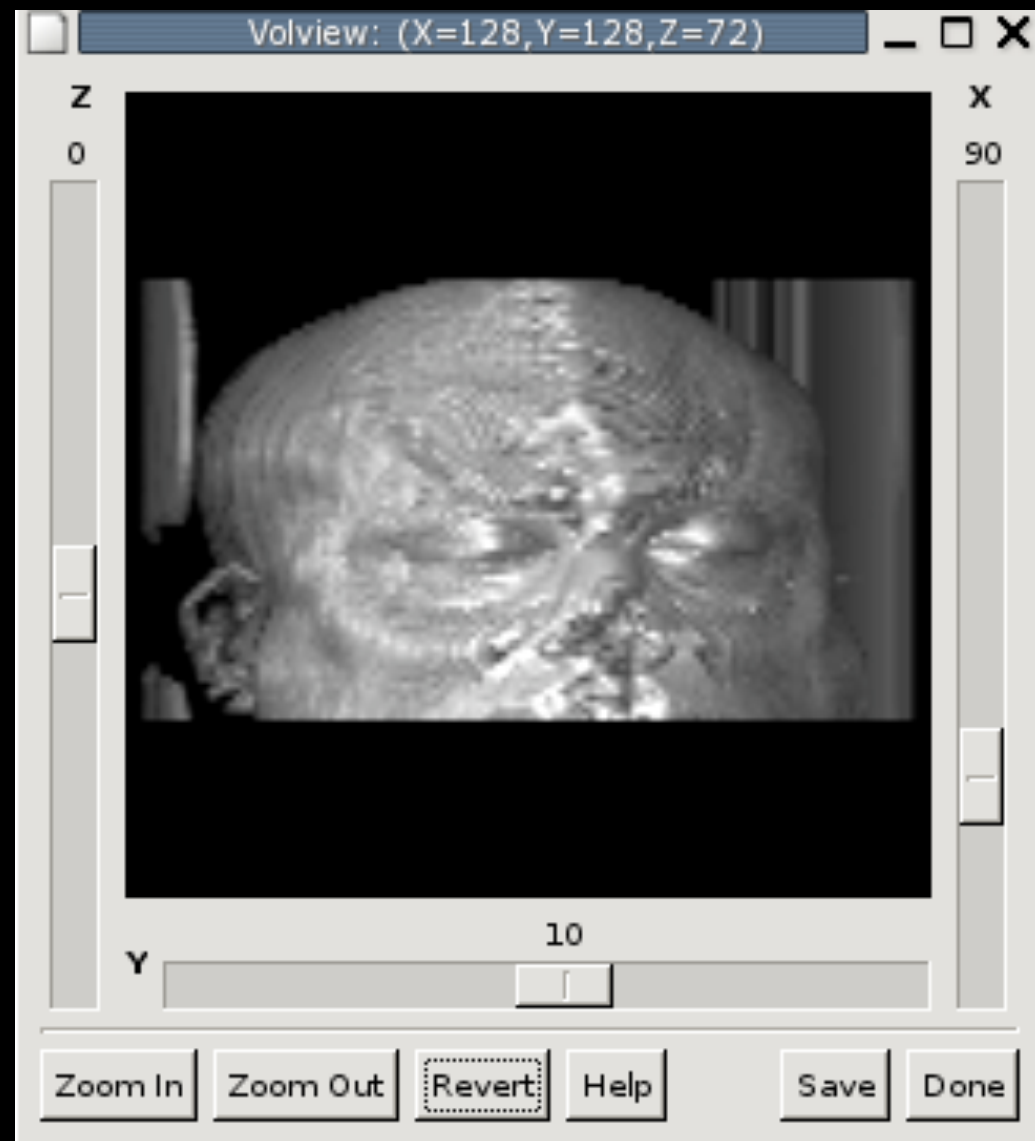
Rather than

Plumbing details of computational platform

(benefits developer, too: see modularity)



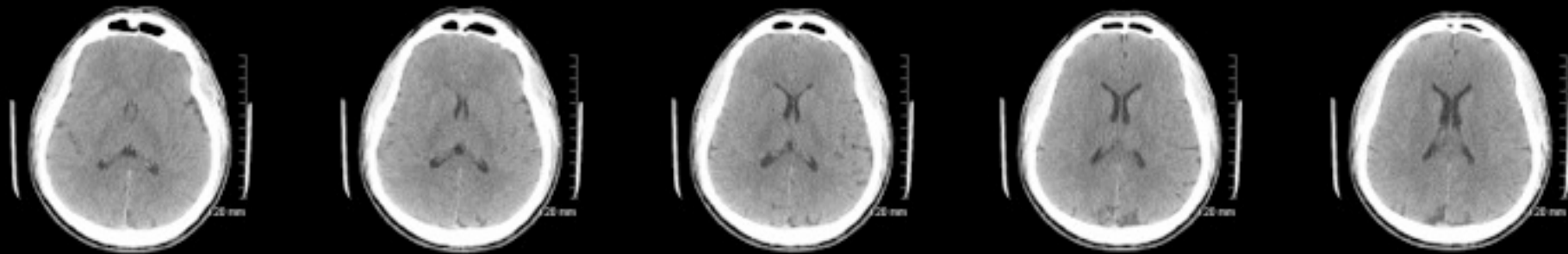
Case Study in Serendipity



MRIs in Astro Spectral Modeling S/W

10 Minutes of Goofing Around

- Series of 2D images in DICOM format



- On CD with Windows XP reader software
- Read as PNG, stacked as 3D & rendered on Linux
- With 120 Kb module wrapper of VOLPACK (1994)

Flexible S/W adapts for unintended uses

OLD ~~≠~~ BAD : volpack VERY FAST (Go NetLib!)

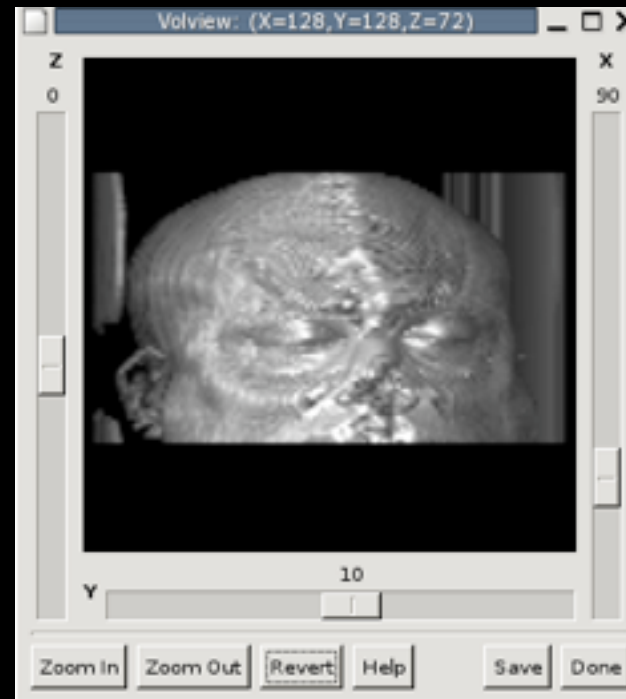
The Entire Script

```
n = 73
vol = UInt_Type[n, 128, 128]

foreach i ([1:n])
    vol [i-1, *, *] = png_read( ...)

volview(vol)

h5_write("mri.h5", vol)
```



Create 3D
From 2D Slices

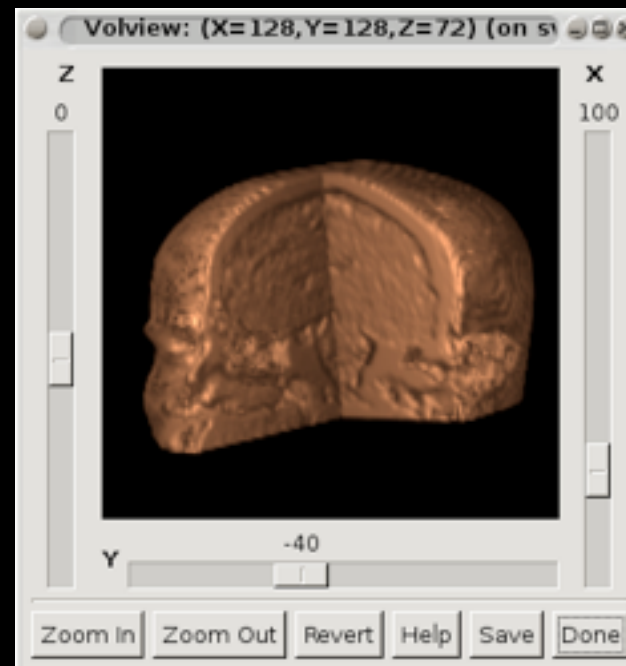
Render
volume

Save as HDF5
for later ease

```
vol [*, [49:99], [6:11]] = 0
vol [*, [28:99], [115:123]] = 0
vol [*, [103:106], [101:118]] = 0

vol [*, [:64],[64:]] = 0

volview(vol)
```

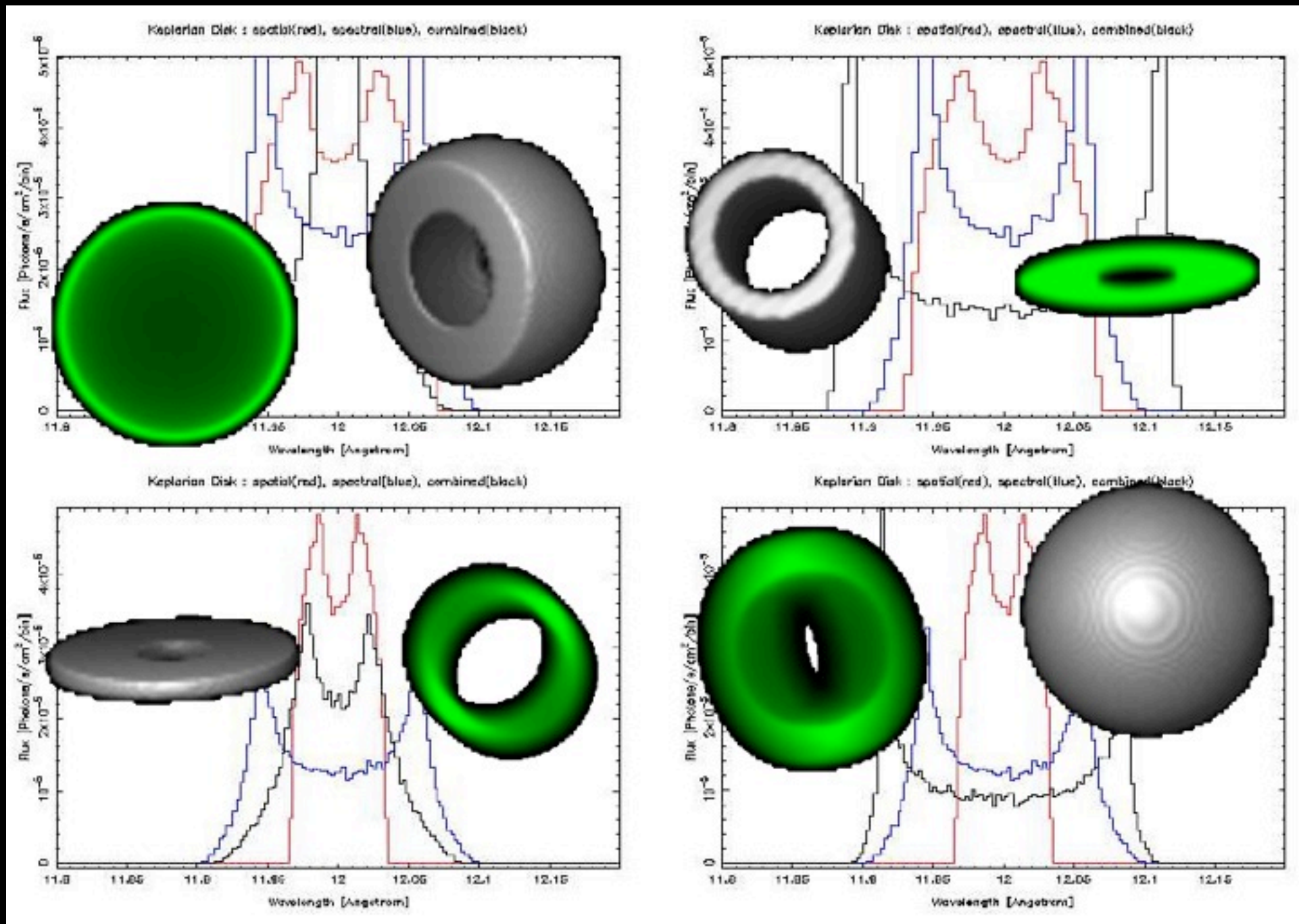


Clean noise
from sides

Cut out
quadrant
to see
interior

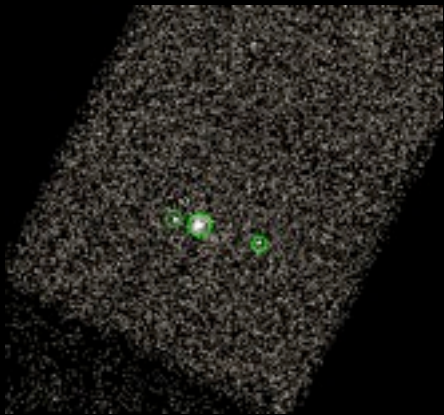
No Need To Convene Design-By-Committee

Real Science : Spatial+Spectral Fitting



Qualitative Visualizations Steer
Quantitative Analyses

ND Arrays are Scientific Clay



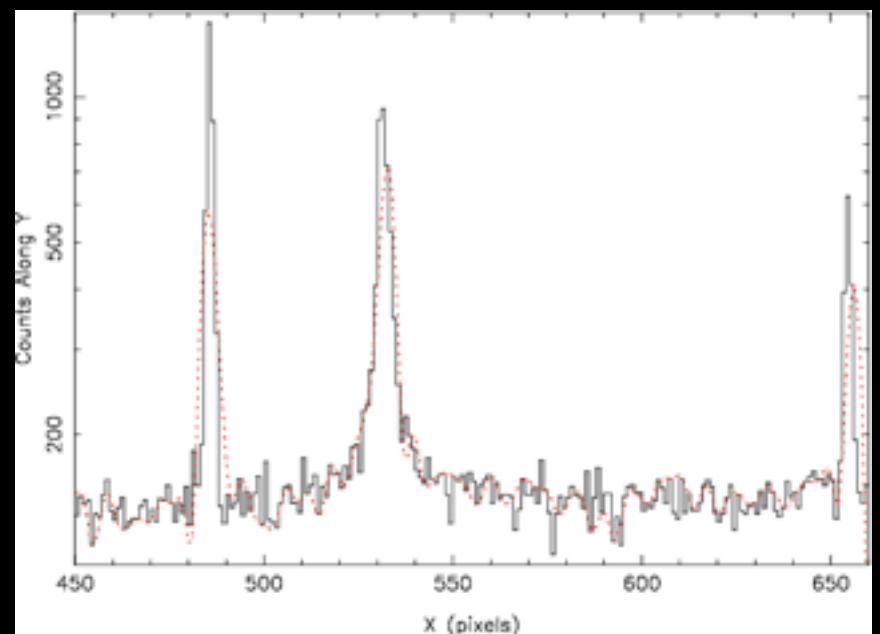
DS9 visualizer showing
ACIS CCD image &
3 regions of interest

```
isis> image = ds9_get_array()  
isis> image  
Float_Type[1024, 1024]
```

To be shaped ...

```
isis> hist = sum(im, 0)  
isis> range = [440:680]  
isis> hplot( range-1, range, hist[range])  
isis> xa = [440:680:4], ya = hist[xa]
```

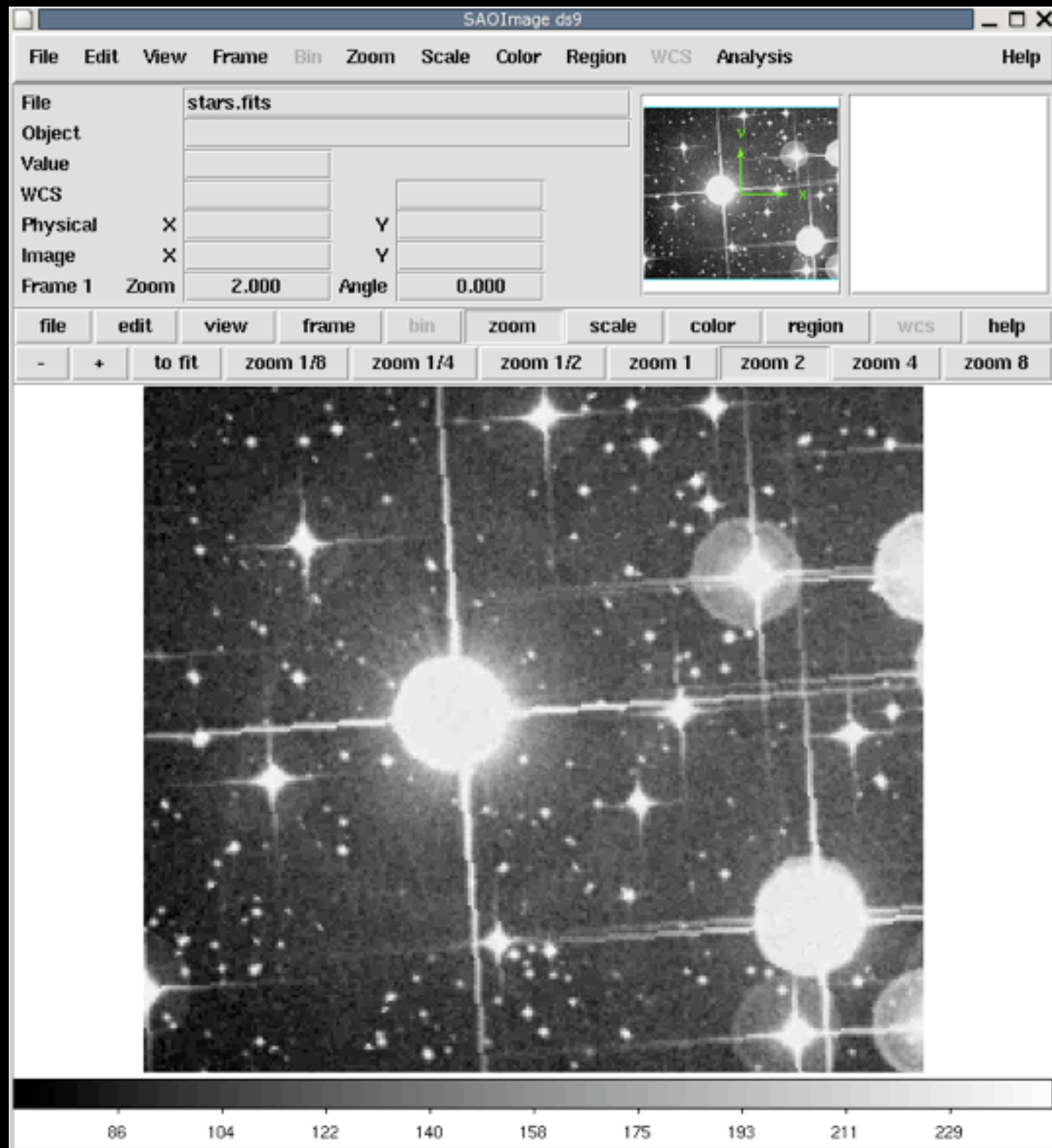
```
isis> require("gsl")  
isis> smooth = interp_cspline(range, xa, ya)  
isis> oplot(range, smooth)
```



... and seen in arbitrary ways

Using a numerical interpreter as potter's wheel

Previous Example : So What?



DS9 single most widely used tool in astronomy

Mostly qualitative, though

Importing Controller Module

```
isis> import("ds9")
```

Merges DS9 imaging with
ISIS numerics & extensibility

Without changing either app!

Modernity: Melange Of

File formats ...

... programming languages,

... monolithic GUIs

Given this complexity ...

More Concerned With How To

Load your data as ND array ...

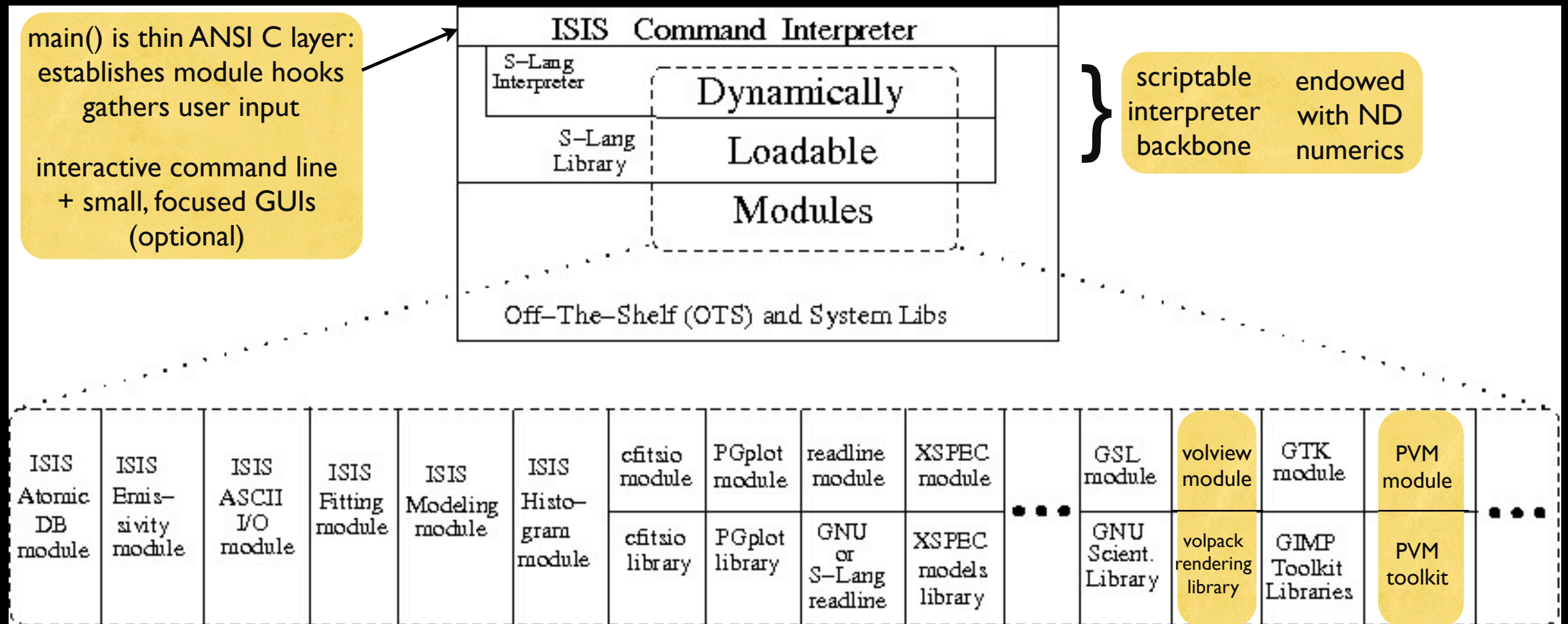
... so I can decide

... how to best SHAPE & SEE it

- Quickly (by end-user / scientist)
- With little or no low-level coding
- Through ad-hoc mixtures of intrinsic functions,
- Plugin modules, scripts, interactive commands,
- Or even compiled code (dynamically loaded)

Pick Your Favorite: MatLab, R, S-Lang, Python, ...

10 Years In ISIS



Bulk of functionality provided by modules

Most can be downloaded/used outside of ISIS

Spectral Modeling Innovations

- Programmable Atomic Database API
- Model == arbitrary interpreted math expression (not text)
- Model evaluation caching
- Hooks to customize nearly every aspect of fitting
- Natural multi-wavelength: X-ray, Radio, Visible, IR, Gamma
- Volume rendering : leading to spatial + spectral fitting
- Data Mining GUI-lets
- Nearly Transparent Parallelism: conf limits, model fit eval, etc
- Read/write access to FORTRAN common blocks
- Automated Vector-Parallel Bindings Generation
- HDF5 I/O

Years Ahead Of Other Groups : Very Small Team

Beyond XSPEC: Toward Highly Configurable
Astrophysical Analysis (Noble & Nowak, 2008, PASP)

Modules: Evolutionary Advantage

Harder to extend venerable/monolithic app with arbitrary features than to add new modules to extensible system

Consider addition of HDF5 to Genome Workbench

Large C++ codebase; Long-ish NCBI institutional cycle

Design/write/test/document/release of ENTIRE APP?

Contrast With: `import("hdf5")`

Leverages Internet: “All problems shallow” (Raymond/Torvalds)
Someone, somewhere likely to have written a module ...

∴ Feature sets evolve MUCH more rapidly

Modules: Developer Advantage

Most of porting problem solved in interpreter, by someone else

Then the common refrain

“Small team, supported only on Linux”

begins to fade away.

JAVA Solves Portability, Too, BUT :

Anyone write interactive numerical interpreters in JAVA?
(Byte) Compiled, static typing == not for itinerant bio coders

Modules: Yet More Advantages

More Stable: internal core of app unchanged as features added
(better partition of test/doc/release workload)

Functional Orthogonality == needs smaller mental map
(internal core need not be mastered by module developer)

More Nimble Schedule: modules released as ready, not app

More Nimble Response: modules can be loaded JIT, per use

Case Study: 2D + 2D Data Mining

Array Selection
With
`where()`

```
isis> arr = [1, 2, 3, 4, 1]
isis> i = where( arr < 2)
isis> print(i)
0, 4
```

Like R “`subset()`” but more concise?

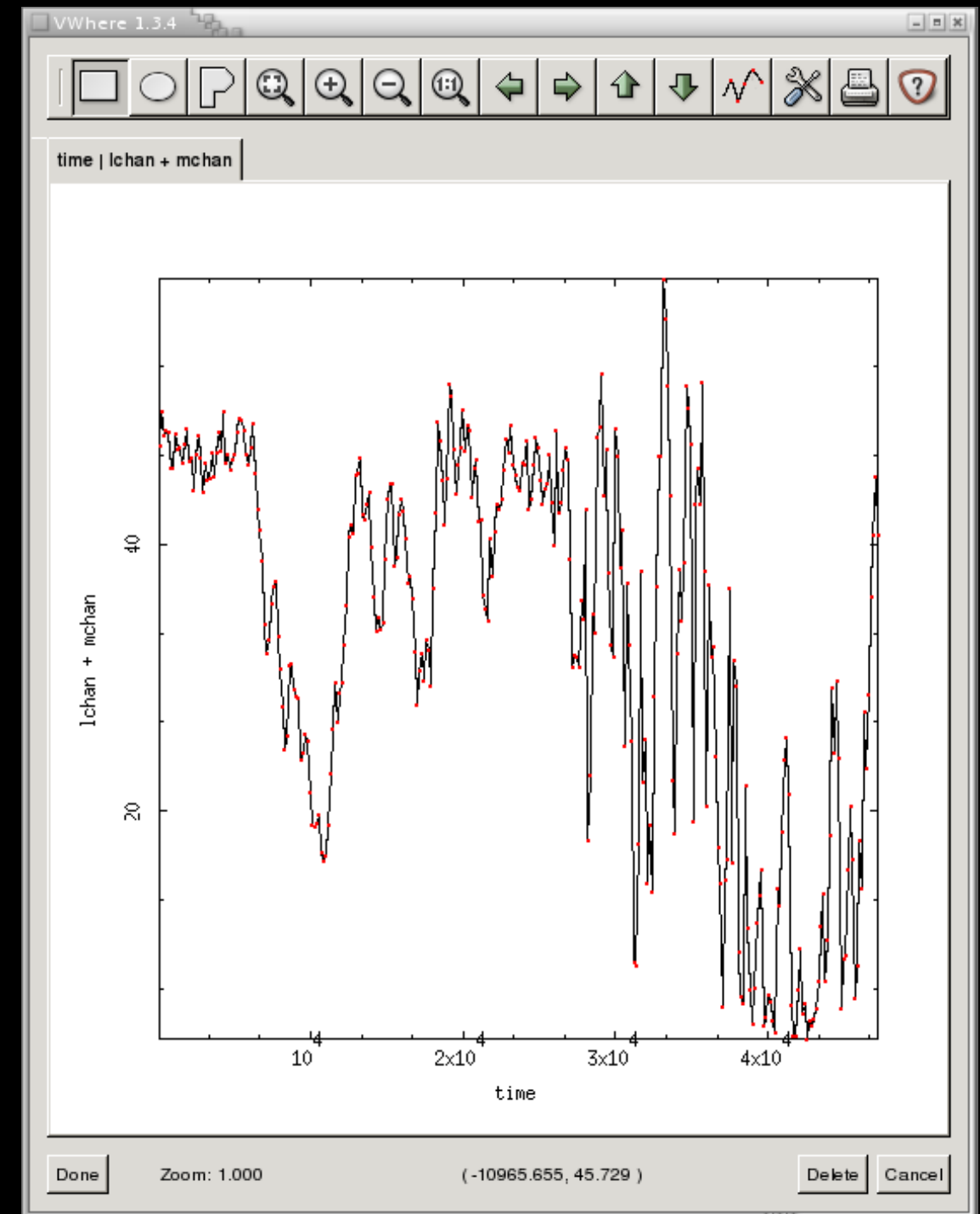
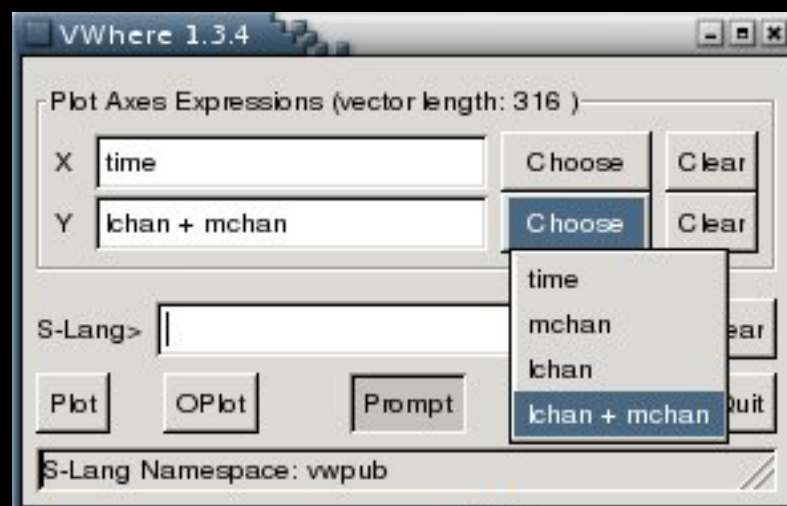
Surprisingly Powerful Feature

VWhere : Visual Where Function

```
isis> table = read("cygnus_x1")  
isis> table  
Struct_Type with 17 fields
```

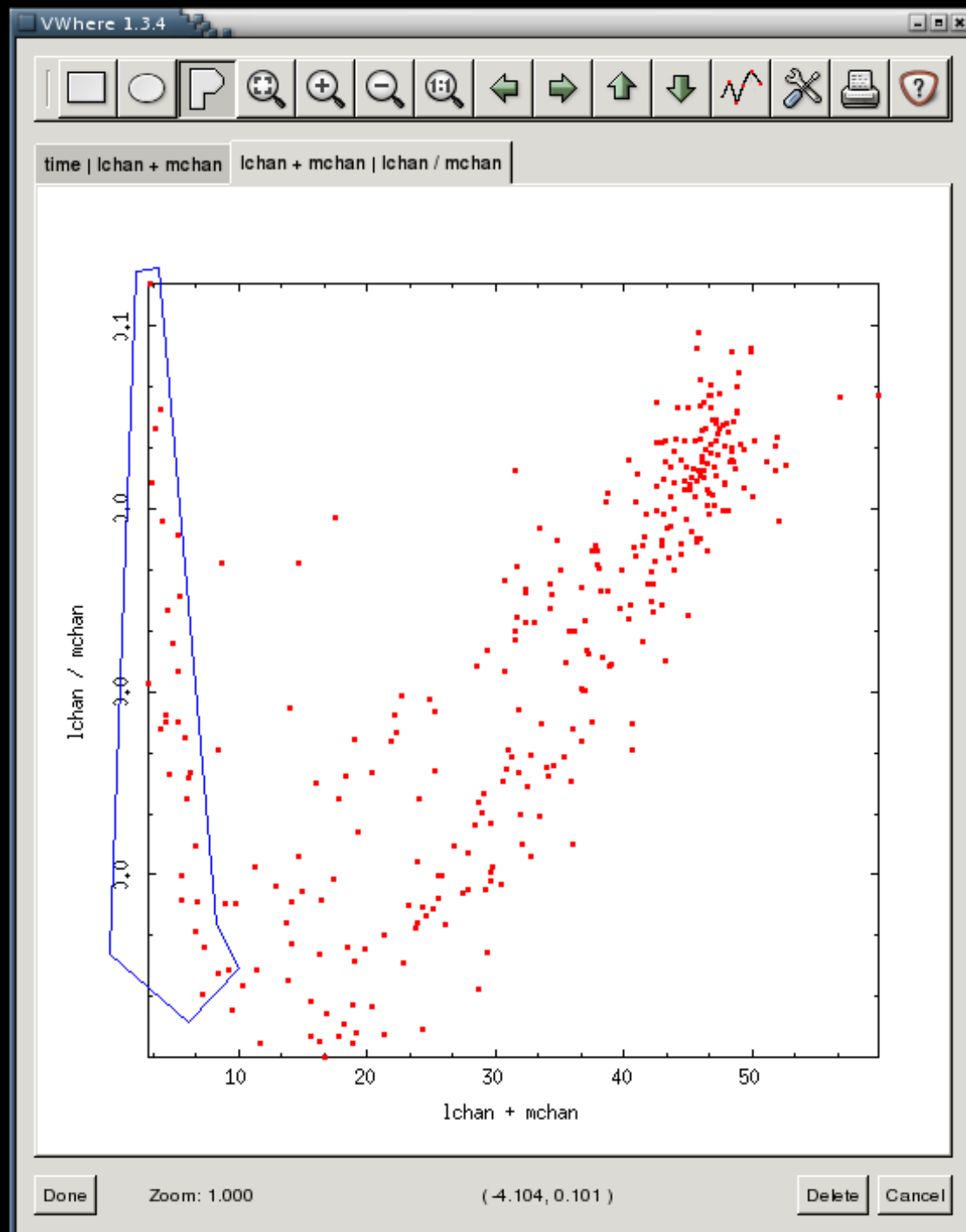
```
isis> points = vwhere(table)
```

GUI-let launched from interactive cmd line

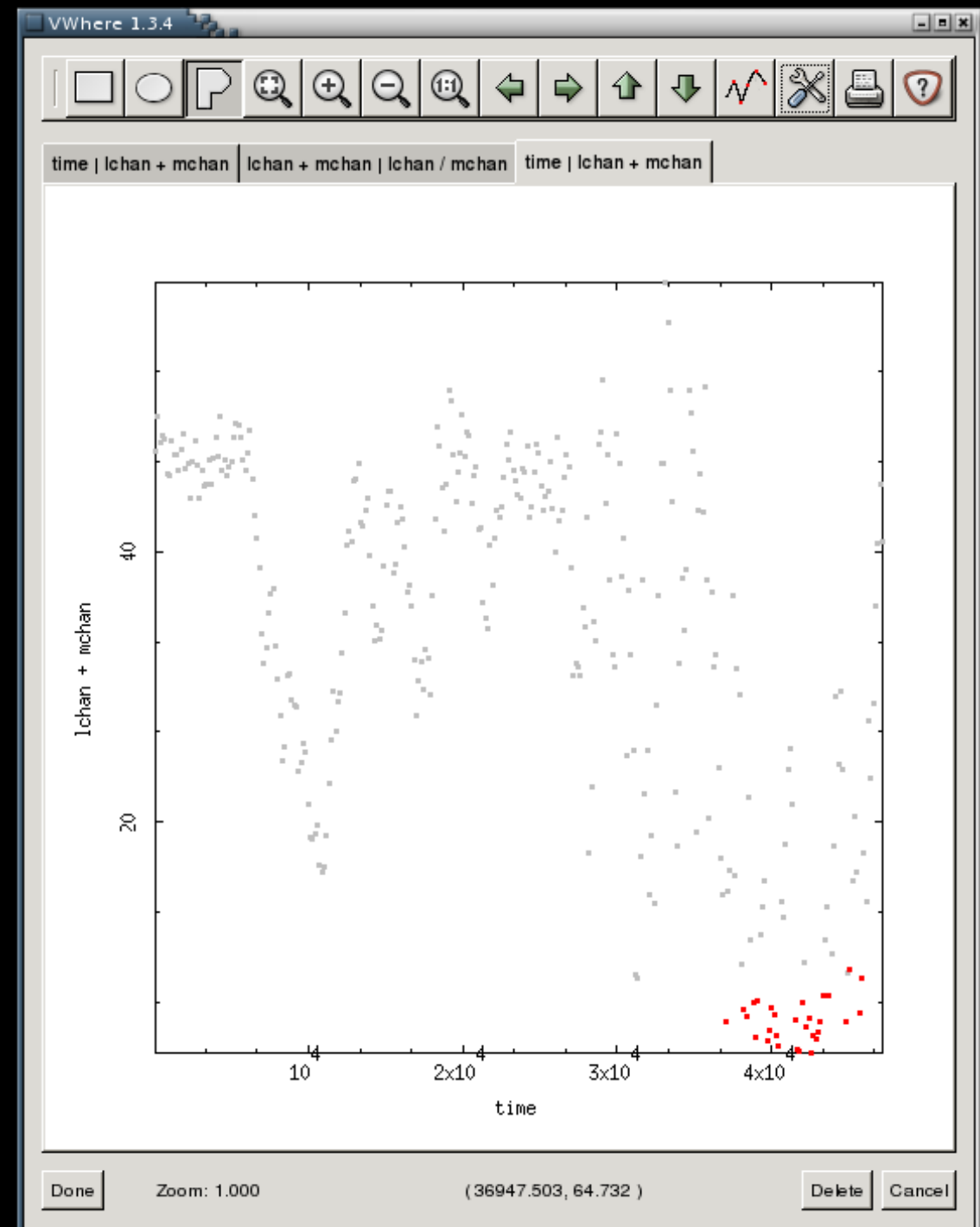


Create new axes on fly with arbitrary numerical expressions,
including calls to C, C++, Fortran modules

Points selected via regions
on plots of 2 axes



Effect instantly seen on
plots of other axes



Returns same array of indices as where()

VWhere Benefits

Faster than file-based command-line tools (10x or more)

Filters accumulated in memory, not files

No file litter

More intuitive : no tool filter syntax needed

Far more powerful, too (extensible math)

Same Paradigm

ND arrays, shaped & seen
in arbitrary ways

Combines power of
interactive analysis
cmd line with GUI
modules, where useful

Case Study: Parallel Model Fitting

You calculate p-value

We calculate χ^2

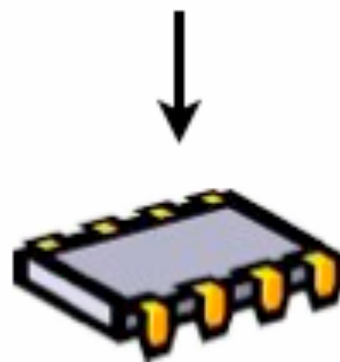
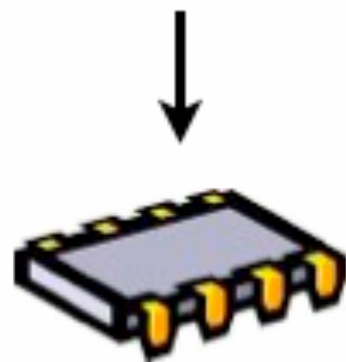
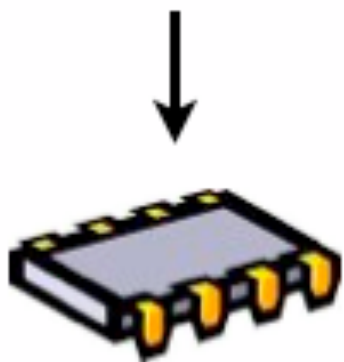
```
isis> load_data("my_data.pha")
isis> model("warmabs + warmabs(2) + hotabs")
isis> set_params(...)
isis> fit
```



warmabs

warmabs(2)

hotabs



Model components
mathematically independent

Embarrassingly parallel

```
isis> require("pmodel")
```

Evaluations farmed to
independent CPUs / hosts

ISIS unaware model is parallel:
internals unchanged
fit performed as if serial

Parallelism in ISIS: 5+ years before any others

Case Study: SLIRP Wrapper Generator

Dramatically reduce effort to call C/C++/Fortran from S-Lang.

Can vectorize funcs for natural S-Lang array usage.

Optionally with OpenMP for multicore.

Read / write access to Fortran common blocks.

Turnkey mechanism for scientists to include their custom models into modern analysis system.

Quick way for developer to generate wrapper modules for OTS libs being considered for use.

SLIRP Example : ASCII Volume

Problem: Visualize 320x320x320 cube of Doppler velocities, from ASCII file

Solution: Convert to real-values with `atof()`

Problem II: Will be slow, `atof()` not vectorized

Solution II : replace it with vector-parallel version

```
linux% slirp -make -openmp atof.h && make
```

SLIRP Example : Performance

Using only 100^3 strings : 33x smaller dataset

```
isis> avol = array_map(String_Type, &sprintf, "%d", [1:100*100*100])
isis> tic; dvol = array_map(Double_Type, &atof, avol); toc
13.754

isis> import("atof")
isis> tic; pdvol = atof(avol); toc
0.1442
```

S-Lang intrinsic : 13.8 sec

Vector-parallel replacement: 0.144 sec

95X faster on dual-core 1.8 GHz

Random Browse of Bio S/W

NCBI: CDTree, NCBI Toolbox, Genome Workbench

Allen Brain Institute: Brain Explorer, etc

Broad Institute: IGV, Arachne, ALLPATHS

... and many more ...

Overwhelmed by sheer volume of data

Impressed by maintenance of provenance

Layman's Concern With

Sense of qualitative, visualization orientation

3D molecules, 2D gene trees, Contig assemblies, ...

Then what?

How does it lead to quantitative analysis?

Stated Differently: Difficult to see data flow between
between disparate but similar tools

“Small team, supported only on Linux”

Tough to swallow by R, MatLab, Python, etc crowd

Conclusion: Concern Turns To Joy



Deeply Resonant
With Themes
Of This Talk

Flexible, Modular, Extensible, Reproducible

∴ Beautiful

But where is FireHose? Why no (Bio)Python?