The PHENIX graphical interface

Nathaniel Echols
NEchols@lbl.gov
help@phenix-online.org
http://www.phenix-online.org
The **PHENIX** Project

**Lawrence Berkeley Laboratory**
- Paul Adams, Pavel Afonine, Nat Echols, Richard Gildea, Ralf Grosse-Kunstleve, Jeff Headd, Nigel Moriarty, Nicholas Sauter, Peter Zwart

**Los Alamos National Laboratory**
- Tom Terwilliger, Li-Wei Hung

**Cambridge University**
- Randy Read, Airlie McCoy, Gabor Bunkoczi, Rob Oeffner

**Duke University**
- Jane & David Richardson, Vincent Chen, Swati Jain, Gary Kapral, Chris Williams, Bryan Arendall, Bradley Hintze

An NIH/NIGMS funded Program Project
Automation of Structure Solution

Data quality

Experimental Phasing

Molecular Replacement

Phase improvement

Model building

Structure completion

Refinement

Ligand identification & Fitting

Validation Deposition
Automation of Structure Solution

Data quality
- phenix.xtriage
- Labelit
- phenix.automr
- phenix.refine

Experimental Phasing
- phenix.autosol

Phase improvement
- Model building

Molecular Replacement

Structure completion
- Refinement

Ligand identification & Fitting
- phenix.ligandfit
- phenix.elbow

Validation
- Deposition
- molprobity
- ligand screen

Why Automation?

- Can speed up the process and can help reduce errors
- Software can try more possibilities than we are typically willing to bother with
- Makes difficult cases more feasible for experts
- Routine structure solution cases are accessible to a wider group of (structural) biologists
- Multiple trials or use of different parameters can be used to estimate uncertainties

- What is required:
  - Software carrying out individual steps
  - Integration between the steps (collaboration between developers)
  - Algorithms to decide which is best from a list of possible results
    - The computer has to make the decisions
  - Strategies for structure determination and decision-making
PHENIX resources online

- help@phenix-online.org: user support
- bugs@phenix-online.org: bug reports
- phenixbb@phenix-online.org: message board (subscribers only)
- Regular stable releases, and “nightly” builds
- Supported on:
  - Linux (RedHat, Fedora)
  - Mac OSX
  - Windows (in progress)
- Extensive documentation
Obtaining PHENIX

• Free to academic users; simple online registration required *(please use your academic email address!)*

• Regular official releases (typically 2-8 months)

• Nightly builds

• Regular releases

• Supported on:
  • Linux (RedHat, Fedora)
  • Mac OSX

• Extensive documentation

http://www.phenix-online.org/download/nightly_builds.cgi
Command line tools

- Very simple and intuitive syntax
- Data validation
  `phenix.xtriage porin_fp.mtz`
- **Automated structure solution**
  `phenix.autosol data=peak.sca seq_file=nsf-d2.seq`
- **Automated model building**
  `phenix.autobuild data=scale.mtz model=mr.pdb seq_file=correct.seq`
- **Automated ligand fitting**
  `phenix.ligandfit data=nsf-d2.mtz model=noligand.pdb ligand=atp.pdb`
- **Structure refinement**
  `phenix.refine nsf-d2.mtz nsf.pdb`
- **Building ligand coordinates and restraints**
  `phenix.elbow --smiles="C12CC3CC(C2)CC(C1)C3"`
Why a GUI?

- Condense and summarize output
- *Human beings make poor text parsers*
- Many results are inherently graphical - better to plot data than show a table
- Higher-level automation:
  - Simplify transitions between programs, and automatically pick relevant files
  - Suggest appropriate next steps (or run them immediately)
- Track and organize results for a project
PHENIX GUI: Major features and design goals

• Automatic interface generation based on underlying configuration files
• All features available in command line versions should be present in the GUI as well
• Integration with Coot and PyMOL for nearly all programs
  • Validation GUI directly controls graphics windows
• Graphical presentation of current progress (where appropriate) and results
• Drag-and-drop of files (from desktop or between windows) supported in most interfaces
• Visual atom selections (mainly for phenix.refine)
• Customization of program behavior and project details
• Simple transitions between programs: start AutoBuild directly from AutoSol, etc.
• Run processes either directly in GUI or independently (“detached”)
• Track and display appropriate citations for programs used
• Automatic bug reports for Python errors - sent directly to me
Central interface

- Project management and application list

Many user-adjustable settings in here

Project-specific settings (including default files)

Simple utilities (including download from PDB)
• Projects are mainly used to track related jobs and store results
• Some project-specific settings available (X-ray data, hydrogen addition)
• Creating a project in a directory will add a subdirectory “.phenix” to store internal data
• Tutorial setup with sample data now integrated with project management

Things to avoid:
• Making your home directory a project directory
• Nesting project directories
• Moving project directories - use “Move project” in the Projects menu for this
Project history

Display can be limited to specific programs with this menu

This icon indicates that the job failed (for any reason)

This icon indicates an aborted job

“Flagging” a job marks it as important

This button opens a summary panel (next slide)
Project history: job summaries

- Overview of input and output files, and statistics

![Project 'scratch': job #55](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>Refine</th>
<th>Run on May 02 2011 10:28 am</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Title: [phenix.refine]

Directory: /private/var/tmp/Refine_55

Config file: .phenix/project_data/refine_55.eff

Statistics:
- R-work: 0.1857
- R-free: 0.1656
- RMSbonds: 0.007
- RMSangles: 0.917

Input files: (SP = /var/tmp)

- /private/var/tmp/1yjp.mtz
  - Reflections file, File with R(free... Input model
- /private/var/tmp/1yjp.pdb

Output files:

- scratch_refine_55.eff: Effective parameters for this run
- scratch_refine_55.geo: Geometry restraints before refi...
- scratch_refine_55.log: phenix.refine log file
- scratch_refine_55.mtz: Map coefficients for Coot
- scratch_refine_55.pdb: Refined model
Project directory layout

• Inside each project directory:

  .phenix/
  project.phil
  job_history.phil
  tmp/
  defaults/
  project_data/
  job_[X].phil
  refine_1.eff
  refine_1.log
  refine_1.pkl

  basic project info
  job history (without details)
  folder for temporary files
  various default files (if defined)
  data for individual jobs
  record of input/output files and statistics
  job runtime files (including log and saved result)

• You should not need to modify any of these files
New feature (January 2012): project groups

- An additional sorting layer, primarily for managing common files and settings - mostly* optional

* Tutorials are automatically added to a “PHENIX tutorials” group - any other project will not be part of a group unless explicitly specified
User preferences

- Settings for overall behavior and individual programs
Overview of available programs

- **Central interface** ("phenix" command)
- **AutoSol, AutoMR, AutoBuild, and LigandFit wizards**
- **phenix.refine** (and associated utilities)
- **Xtriage** - comprehensive assessment of data quality
- **Phaser** - advanced interface for MR and SAD
- **Validation** - most of Molprobity, and more
- **Several map-related interfaces**
- **Reflection file editor** - combine files, create or extend R-free flags
- **REEL** - graphical restraints editor (*Nigel Moriarty*)
- **40+ programs currently available, more coming soon**
### Overview of available programs

#### Molecular replacement

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoMR (simple interface)</strong></td>
<td>Automated molecular replacement with Phaser – search for a single ensemble</td>
</tr>
<tr>
<td><strong>AutoMR (advanced interface)</strong></td>
<td>Automated molecular replacement with Phaser – supports multiple ensembles and components</td>
</tr>
<tr>
<td><strong>Phaser-MR (simple interface)</strong> [alpha]</td>
<td>Automated molecular replacement with Phaser – search for a single ensemble</td>
</tr>
<tr>
<td><strong>Phaser-MR</strong></td>
<td>Maximum-likelihood molecular replacement</td>
</tr>
<tr>
<td><strong>MRage – automated pipeline</strong>  [alpha]</td>
<td>Integrated model identification, preparation, and parallel MR search</td>
</tr>
<tr>
<td><strong>MR-Rosetta (beta)</strong></td>
<td>AutoMR combined with Rosetta model improvement and AutoBuild for difficult structures</td>
</tr>
<tr>
<td><strong>Sculptor</strong></td>
<td>Modify a molecular replacement search model</td>
</tr>
<tr>
<td><strong>Sculptor – Coot interface</strong></td>
<td>Extension to Coot GUI for running Sculptor interactively</td>
</tr>
<tr>
<td><strong>Ensembler</strong></td>
<td>Create ensemble of models for molecular replacement</td>
</tr>
<tr>
<td><strong>Parallel Phaser</strong> [alpha]</td>
<td>Evaluate many different search models independently across multiple processors</td>
</tr>
</tbody>
</table>

#### Experimental phasing

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoSol</strong></td>
<td>Automated experimental phasing with model-building</td>
</tr>
<tr>
<td><strong>Hybrid Substructure Search</strong></td>
<td>Dual-space identification of heavy-atom sites</td>
</tr>
<tr>
<td><strong>Phaser-EP</strong></td>
<td>Maximum-likelihood SAD experimental phasing</td>
</tr>
</tbody>
</table>

#### Model building

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoBuild</strong></td>
<td>Automated model-building and refinement</td>
</tr>
<tr>
<td><strong>Phase and build</strong></td>
<td>Faster auto-building combined with density modification</td>
</tr>
<tr>
<td><strong>Find Helices and Strands</strong></td>
<td>Fast chain tracing</td>
</tr>
<tr>
<td><strong>Fit Loops</strong></td>
<td>Fast placement of missing loops in electron density</td>
</tr>
<tr>
<td><strong>Morph model</strong></td>
<td>Model improvement for poor molecular replacement solutions</td>
</tr>
</tbody>
</table>
### Overview of available programs

<table>
<thead>
<tr>
<th>Maps</th>
<th>Ligands</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Calculate maps</strong></td>
<td><strong>LigandFit</strong></td>
<td><strong>phenix.refine</strong></td>
</tr>
<tr>
<td>Utility for creating likelihood-weighted maps using model phases, output as MTZ file or CCP4-format maps</td>
<td>Fitting of ligands into electron density</td>
<td>Automated X-ray and neutron refinement</td>
</tr>
<tr>
<td><strong>FFT map coefficients</strong></td>
<td><strong>eLBOW</strong></td>
<td><strong>DEN refinement [alpha]</strong></td>
</tr>
<tr>
<td>Generate map file(s) in CCP4 or XPLOR format from an MTZ file containing one or more sets of map coefficients</td>
<td>electronic Ligand Building and Optimization Workbench: ligand restraints generation and optimization</td>
<td>Deformable elastic network refinement using simulated annealing, for low-resolution and molecular replacement structures</td>
</tr>
<tr>
<td><strong>AutoBuild – create omit map</strong></td>
<td><strong>eLBOW (wizard-style) [alpha]</strong></td>
<td><strong>ReadySet</strong></td>
</tr>
<tr>
<td>Simulated annealing and iterative-build omit maps using the AutoBuild wizard</td>
<td>Simplified interface to eLBOW</td>
<td>Utility for preparing PDB files for refinement – automatically generate restraints and add hydrogens</td>
</tr>
<tr>
<td><strong>Density modification [alpha]</strong></td>
<td><strong>Ligand identification</strong></td>
<td><strong>Covalent ligand setup [alpha]</strong></td>
</tr>
<tr>
<td>Simple density modification by solvent flipping</td>
<td>Automated ligand search using database of 200 most frequent ligands, or user input</td>
<td>Generate link restraints for various covalent ligands, such as carbohydrate groups</td>
</tr>
<tr>
<td><strong>RESOLVE density modification</strong></td>
<td><strong>Guided ligand replacement</strong></td>
<td></td>
</tr>
<tr>
<td>Simple interface for running density modification only using AutoBuild and RESOLVE</td>
<td>Ligand fitting based on an existing protein–ligand complex</td>
<td></td>
</tr>
<tr>
<td><strong>Isomorphous difference map</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Fo–Fo map from isomorphous datasets</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Superpose maps</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Superpose two PDB files and transform the associated map coefficients to the new orientation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Cut out density</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extract an arbitrary user-defined region from map coefficients file</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Multi-crystal averaging</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Density modification with multi-crystal averaging of maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Find difference map peaks and holes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identify local maxima and minima in mFo–DFc map (and anomalous map if available) and flag waters with excess density</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Overview of available programs

### Reflection tools
- **Xtriage**
  Analysis of data quality and crystal defects
- **Reflection file editor**
  Utility for merging and converting reflections
- **Calculate F(model)**
  Utility for generating structure factors from a PDB file
- **Import CIF structure factors**
  Convert data deposited in PDB to MTZ format
- **French & Wilson data correction**
  Statistics-based handling of negative intensities
- **Model-based phases**
  Utility to obtain PHI, FOM, and Hendrickson-Lattman coefficients from data and PDB file
- **3D data viewer**
  OpenGL visualization of reflection data in reciprocal space
- **2D data viewer**
  Visualization of reflection data in slices through reciprocal space

### Model tools
- **PDB Tools**
  Utility for simple modifications of PDB files, including geometry regularization
- **Combine PDB files**
  Merge a model split across multiple files, with automatic chain renumbering and clash check
- **Superpose PDB files**
  Simple structure alignment program
- **Find NCS operators**
  Identify non-crystallographic symmetry in model, heavy-atom sites, or electron density map
- **Apply NCS operators**
  Transform a molecule by NCS matrices to generate complete structure
- **Add conformations [alpha]**
  Add alternate conformations in bulk, for an entire model or user-defined atom selection

### Utilities
- **Calculate map correlations**
  Simple calculation of CC between two sets of map coefficients in MTZ format, accounting for origin shifts
- **Calculate model-map correlations**
  Simple calculation of CC between a PDB file and map coefficients in MTZ format, accounting for origin shifts
- **Generate "Table 1" for journal**
  Extraction of final model statistics for publication, including data processing logfiles
- **Diffraction image viewer**
  Simple viewer for detector images, similar to ADXV

---

---

---

---

---
“Expert level” control in dialog windows

Configuration interfaces can be dynamically adjusted to show only the most basic/popular options, or more detail:

Changing the “user level” shows or hides advanced controls; the default level can be set in the Preferences.
Keyword search for parameters

Most documentation covers the command-line parameters; the corresponding GUI controls may be easiest to find with the search.

It helps to be as specific as possible: searching for “mask” alone finds 19 parameters, “optimize mask” (shown) finds 5, and “optimize_mask” (a specific parameter name) finds 2.
Coot/PyMOL integration

- Coot must have Python support (default on Mac)
- Specific paths to executables usually required on Linux
  - Preferences->Graphics->Full path to Coot [...PyMOL]
- Most results can be opened directly in graphics apps
- Any PDB file listed in GUI can also be opened
- AutoSol, AutoBuild, and phenix.refine will update Coot continuously while running
File management

• Bulk file input possible on both Mac + Linux
Automatic bug reports

- Sent via email to Phenix developers - please submit!
Utility functions (“Other tools” in main GUI)

- Convenient access to very small and fast programs
reflection file editor

Combine and manipulate reflection files in any format, output as MTZ. Capable of extending old R-free sets, and generating new sets as thin shells (for refinement in presence of NCS). For use with fully processed data only - reflections will be merged and h,k,l indices altered as required.

(All functionality is also available on the command line as iotbx.reflection_file_editor, but we recommend using the GUI for this unless you are scripting an automation pipeline.)
phenix.data_viewer: visualizing reflections in 3D

Useful for identifying pathologies and other dataset properties

Anomalous data in P21212, showing missing reflections (white) and systematic absences (violet)
Data analysis with *phenix.xtriage*

- Analysis of signal-to-noise, data quality, Wilson plot, translational NCS, twinning, symmetry issues, and more

**Wilson plot and B-factors for a typical protein crystal dataset** ($PHENIX/examples/porin-twin$)

**Anomalous signal vs. resolution for an excellent SeMet dataset** ($PHENIX/examples/p9-sad$)

*AutoSol* and *AutoBuild* run Xtriage almost immediately, and results can be viewed from those GUIs. However, it may save time and effort to run Xtriage yourself first.
Identifying twinned structures in Xtriage

Twinning can’t be detected by looking at diffraction images, but it changes the distribution of intensity values in predictable ways.

Good data (p9-sad example): observed intensity distributions are close to expected values.

Twinned data (porin-twin example): NZ test curve is sigmoidal, L test curve is shifted upwards.

Intensity distributions can also be affected by pseudotranslation (especially NZ test); make sure you look at all of the evidence for twinning!
Identifying twinned structures in Xtriage

The twin fraction for all possible twin laws will be estimated; usually one of these is obviously different.

Two twin laws from the porin-twin example are shown; in this case $h,-h-k,-l$ is the actual twin law for this crystal. This can be used in phenix.refine, which will determine the true twin fraction based on the refined model.

The validation GUI (or phenix.model_vs_data) will also try to determine if your structure is twinned based on the R-factors with and without a twin law.
AutoSol: an experimental phasing pipeline

1. Anomalous and/or isomorphous data
2. Correct for anisotropy (if needed)
3. Scale data
4. Locate substructure with phenix.hyss
5. Score substructure hands (analysis of maps)
6. Phasing (Phaser or Solve)
7. Statistical density modification (Resolve)
8. Initial model building (Resolve)

Substructure Completion

The AutoSol graphical interface

Files added by dragging from desktop into the window

Very little input required for simple SAD experiments, but multiple datasets and methods are also supported

All phasing results (sites and maps) linked to building programs and external graphics windows
How Competitive is Automated Solution?

Tom Terwilliger, Paul Adams
AutoMR: Phaser made easy

- Streamlined setup of ensembles and composition

Single-click transitions to building, refinement, MR-SAD GUIs

Phaser: Airlie McCoy, Gabor Bunkoczi, Rob Oeffner, Randy Read; AutoMR: Tom Terwilliger
Phaser-MR for advanced users and difficult cases

• Includes all features of command-line program

- Supports all MR modes (automatic or manual)
- Most keywords found here
- Can use a low-resolution map as a search model
- One-click re-use of partial solutions from past runs
- Any reflection file format permitted

Phaser: Airlie McCoy, Gabor Bunkoczi, Rob Oeffner, Randy Read
phenix.maps GUI

Very simple interface for creating simple maps (including anomalous difference maps) in MTZ or XPLOR format*

*To save disk space, Phenix does not write XPLOR or CCP4 maps by default; however, most programs in the GUI will convert MTZ map coefficients to CCP4 format when you click the “Open in PyMOL” button.


Fill missing F(obs) with F(calc): often improves 2mFo-DFc maps, but watch out for bias! (phenix.refine and Refmac both do this)
Combines with `phenix.ready_set` for adding hydrogen/deuterium and generating restraints - not fully automatic yet

Automatic re-use of parameters in subsequent refinement jobs

`phenix.refine`: Pavel Afonine et al.; `phenix.ready_set`: Nigel Moriarty
phenix.refine: graphical extensions

- `phenix.find_tls_groups`: highly parallel automatic TLS setup (similar to TLSMD), available as interactive component

Buttons to launch find_tls_groups or TLSMD web server

Visualization of atom selections

`phenix.find_tls_groups`: Pavel Afonine
Integrating refinement and validation

- Constant feedback during refinement enables immediate detection of potential problems.

Molprobity validation scores

Coot model and maps updated at each step

R-factors and geometry deviations by cycle
Visualizing validation problems

- Outlier lists recenter Coot view; Probe dots automatically loaded
Advanced validation tools

- Combines Molprobity with phenix.model_vs_data; run automatically after phenix.refine
Graphical comparison of statistics versus the PDB

Colored bars are one-dimensional histograms showing distribution of values for structures at similar resolution.

The black polygon shows where the statistics for the user's structure fall in each histogram.

The structure used to generate this figure has good geometry relative to the PDB, but very poor R-factors.
Parallel validation of multiple structures

• Identifies points of difference between structures of the same protein, with optional map superpositioning

Comparison of sidechain rotamers across all chains (green = most common, yellow = minority, red = outlier)

Double-clicking any cell in the grid zooms Coot/PyMOL

(Collaboration with Herb Klei, BMS)
Works in progress and future plans

• **Improved Windows support**
• Fully automated molecular replacement
• Simplified GUI for eLBOW (ligand restraints)
• *LABELIT* GUI (indexing of diffraction images)
• You can preview new developments by checking “Enable alpha-test programs and features” in the preferences
• *Suggestions? Email NEchols@lbl.gov*
# Acknowledgments

**Lawrence Berkeley Laboratory**
- Paul Adams, Pavel Afonine, Richard Gildea, Ralf Grosse-Kunstleve, Jeff Headd, Nigel Moriarty, Nicholas Sauter, Peter Zwart

**Los Alamos National Laboratory**
- Tom Terwilliger, Li-Wei Hung

**Cambridge University**
- Randy Read, Airlie McCoy, Laurent Storoni, Gabor Bunkoczi, Robert Oeffner

**Duke University**
- Jane Richardson & David Richardson, Ian Davis, Vincent Chen, Chris Williams, Bryan Arendall, Laura Murray, Gary Kapral, Swati Jain, Bradley Hintze

**Others**
- Alexandre Urzhumtsev
- Luc Bourhis
- Herb Klei
- Garib Murshudov & Alexi Vagin
- Paul Emsley, Kevin Cowtan, Bernhard Lohkamp, William Scott, Charles Ballard
- Warren DeLano
- David Abrahams
- PHENIX Testers & Users: Brent Appleton, Joel Bard, Scott Classen, Ben Eisenbraun, James Fraser, Felix Frolow, Christine Gee, Miguel Ortiz-Lombardia, Blaine Mooers, Bob Nolte, Engin Ozkan, Daniil Prigozhin, Miles Pufall, Richard Rymer, Edward Snell, Eugene Valkov, Erik Vogan, Frank von Delft, Andre White, and many more

**Funding:**
- NIH/NIGMS: P01GM063210, P50GM062412, P01GM064692, R01GM071939
- PHENIX Industrial Consortium
- Lawrence Berkeley Laboratory

**Authors of open-source software packages:**
- wxPython, matplotlib, numpy, ksDSSP, MUSCLE, PULCHRA