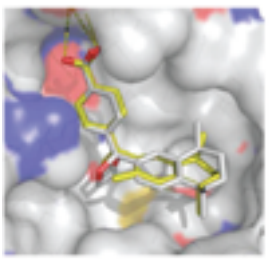
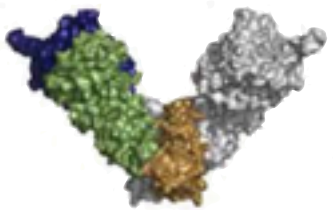
Modeling of proteins and their assemblies with MODELLER and IMP

http://salilab.org/
Dr. Benjamin Webb
Sali Lab
University of California San Francisco

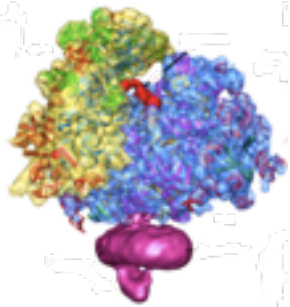
Modeling structures



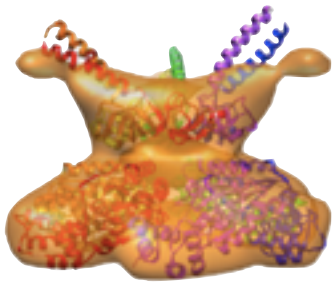
RXRa



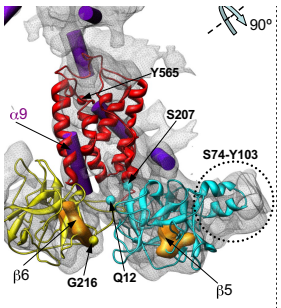
HSP90



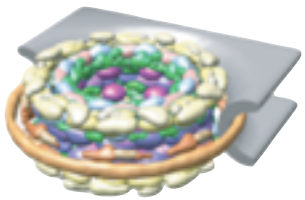
ribosome



26S proteasome



RyR1

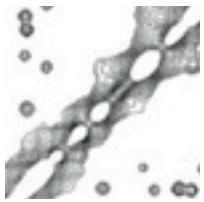


NPC

atom positions → residue positions → member orientations → member positions

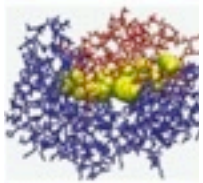
Need to be able to simultaneously process structure on all these scales.

Data sources



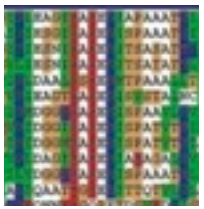
NMR

structure prediction

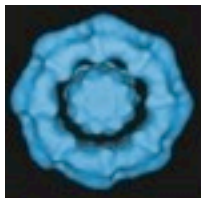


site-directed mutagenesis

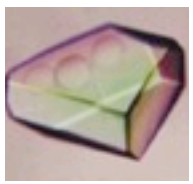
affinity purification



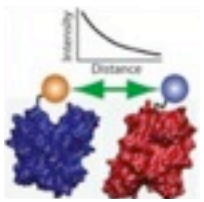
bioinformatics



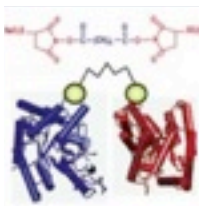
cryo-EM



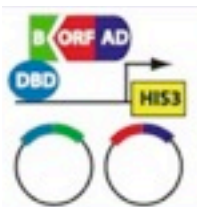
X-ray structures



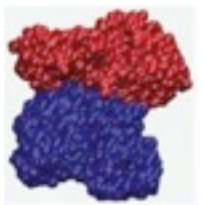
FRET



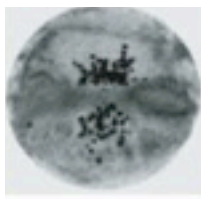
crosslinking



yeast two-hybrid



computational docking



immuno-EM

atom positions → residue positions → member orientations → member positions

Need to be able to simultaneously process all the different data sources.

Satisfaction of spatial restraints

- Represent system at appropriate level(s) of resolution
  - e.g. atoms, residues, domains, proteins
- Convert each data source into spatial restraints
  - e.g. harmonic distance (“spring”)
- Sum all restraints into a scoring function
- Generate models that are consistent with all restraints by optimizing the scoring function
  - e.g. conjugate gradients, molecular dynamics, Monte Carlo
- Our MODELLER and IMP packages both apply this method
  - MODELLER: comparative modeling of proteins
  - IMP: integrative assembly modeling

Why Protein Structure Prediction?

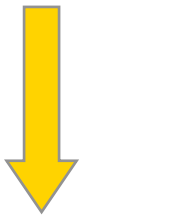
	Y 2008
Sequences	5,000,000
Structures	49,000

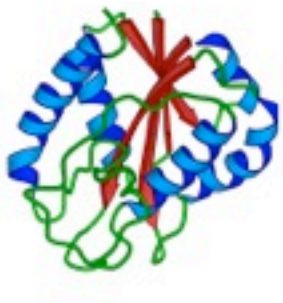
We have an experimentally determined atomic structure for only ~1% of the known protein sequences.

Principles of protein structure

D. Baker & A. Sali. *Science* **294,** 93-97, 2001.

GFCHIKAYTRLIMVG…





Folding

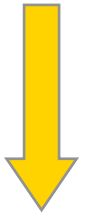
(physics)

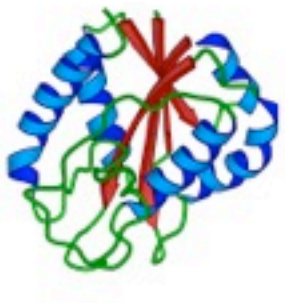
***Ab initio* prediction**

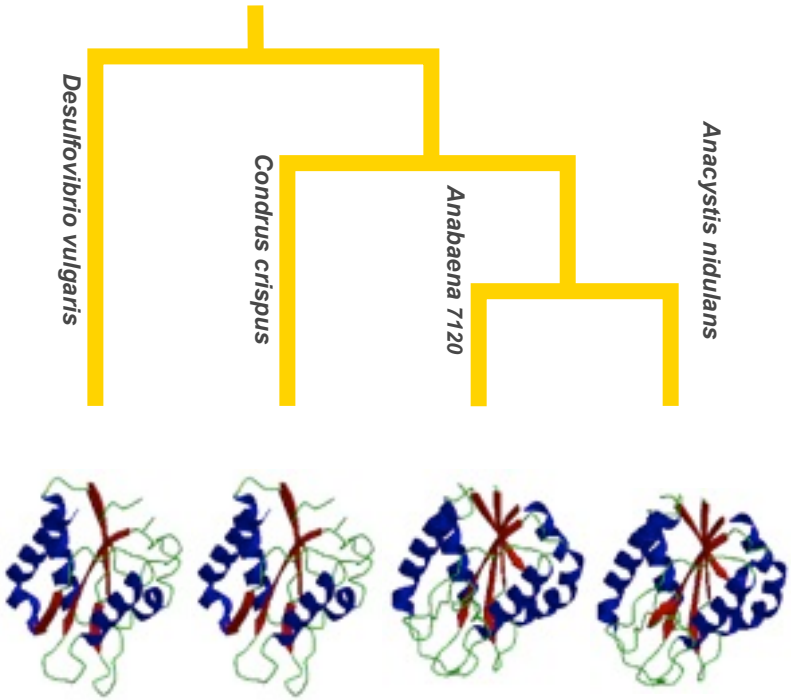
Principles of protein structure

D. Baker & A. Sali. *Science* **294,** 93-97, 2001.

GFCHIKAYTRLIMVG…







Folding

(physics)

***Ab initio* prediction**

Evolution

("statistical" rules)

Threading
Comparative Modeling

The "comparative modeling" principle

The EMBO Journal vol.5 no.4 pp.823–826, 1986

The relation between the divergence of sequence and structure in proteins

Cyrus Chothia[1] and Arthur M.Lesk[2]

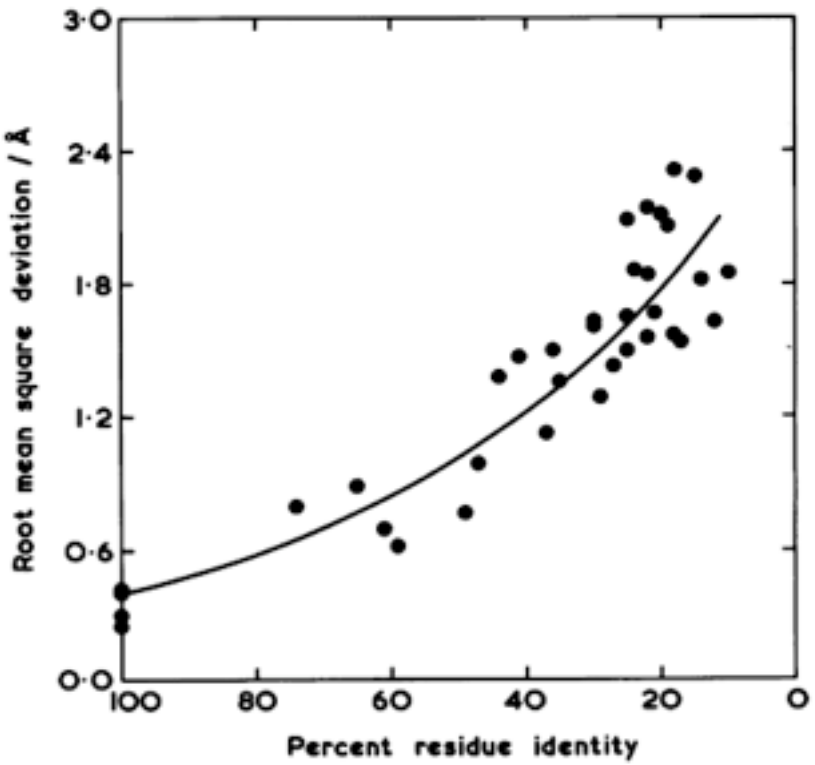


Fig. 2. The relation of residue identity and the r.m.s. deviation of the backbone atoms of the common cores of 32 pairs of homologous proteins (see Table II).

Comparative modeling overview

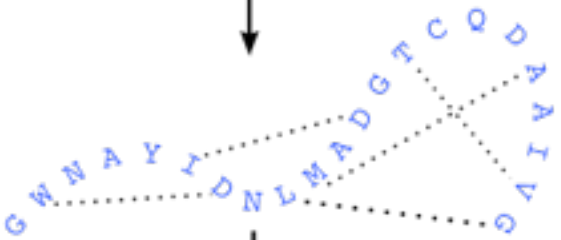
- How does it work?
  - Extract information from known structures (one or more templates), and use to build the structure for the ‘target’ sequence
  - Should also consider information from other sources: physical force fields, statistics (e.g. PDB mining)
- Classes of methods for comparative modeling
  - Assembly of rigid bodies (core, loops, sidechains)
  - Segment matching
  - Satisfaction of spatial restraints

Comparative modeling by satisfaction of spatial restraints - MODELLER

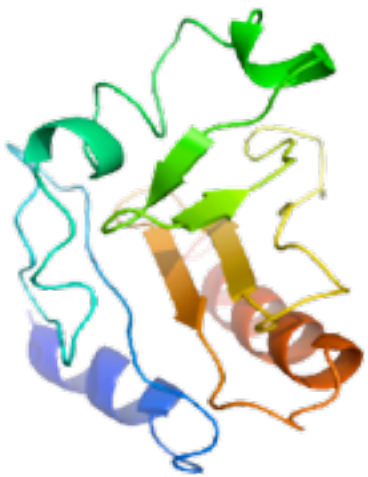
1. Align sequence with structures

Template structure(s) SWQTYVDTNLVGTGAVTQA--AI
Target sequence -GWNAYIDNLMADGTCQDAAIVG

2. Extract spatial restraints



3. Satisfy spatial restraints

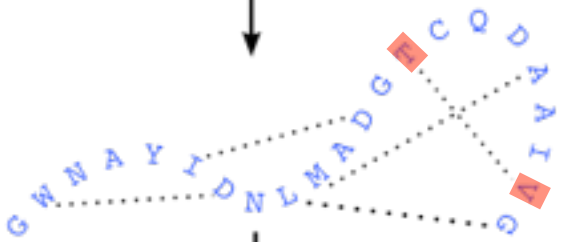


A. Šali & T. Blundell. J. Mol. Biol. 234, 779, 1993.
J.P. Overington & A. Šali. Prot. Sci. 3, 1582, 1994.
A. Fiser, R. Do & A. Šali, Prot. Sci., 9, 1753, 2000.

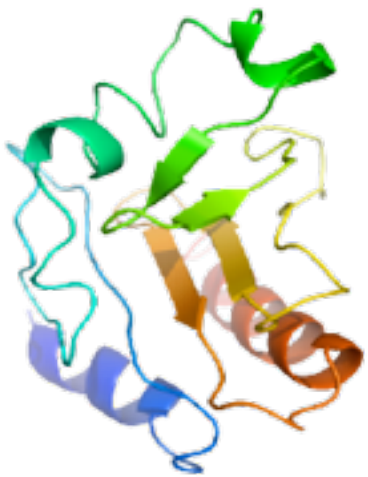
Comparative modeling by satisfaction of spatial restraints - MODELLER

1. Align sequence with structures

2. Extract spatial restraints



3. Satisfy spatial restraints



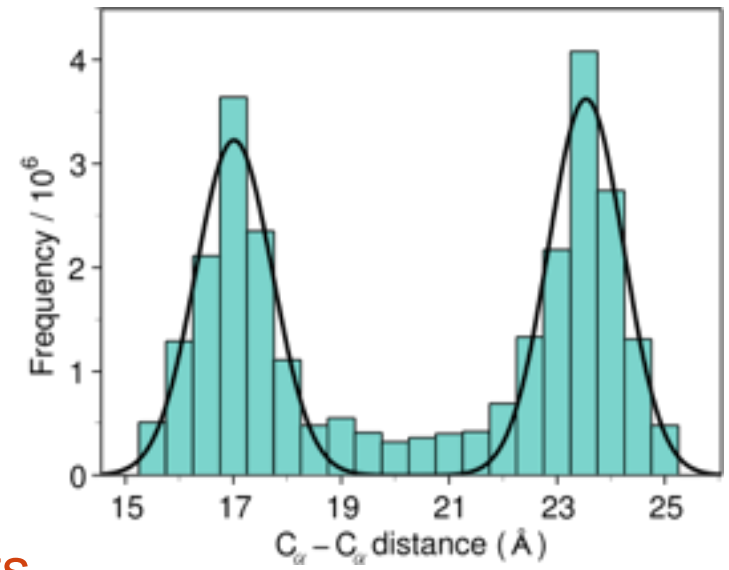
A. Šali & T. Blundell. J. Mol. Biol. 234, 779, 1993.
J.P. Overington & A. Šali. Prot. Sci. 3, 1582, 1994.
A. Fiser, R. Do & A. Šali, Prot. Sci., 9, 1753, 2000.

1. Align sequence with structures

- First, must determine the template structures
  - Simplistically, try to align the target sequence against every known structure's sequence
  - In practice, this is too slow, so heuristics are used (e.g. BLAST)
  - Profile or HMM searches are generally more sensitive in difficult cases (e.g. Modeller's profile.build method, or PSI-BLAST)
  - Could also use threading or other web servers
- Alignment to templates often uses global dynamic programming
  - Sequence-sequence: relies purely on a matrix of observed residue-residue mutation probabilities ('align')
  - Sequence-structure: gap insertion is penalized within secondary structure (helices etc.) ('align2d')
  - Other features, profile-profile, and/or user-defined ('salign') or use an external program

2. Extract spatial restraints

- Spatial restraints incorporate homology information, statistical preferences, and physical knowledge
  - Template Cα- Cα internal distances
  - Backbone dihedrals (φ/ψ)
  - Sidechain dihedrals given residue type of both target and template
  - Force field stereochemistry (bond, angle, dihedral)
  - Statistical potentials
  - Other experimental constraints
  - etc.



3. Satisfy spatial restraints

- All information is combined into a single objective function
  - Restraints and statistics are converted to an “energy” by taking the negative log
  - Force field (CHARMM 22) simply added in
- Function is optimized by conjugate gradients and simulated annealing molecular dynamics, starting from the target sequence threaded onto template structure(s)
- Multiple models are generally recommended; ‘best’ model or cluster or models chosen by simply taking the lowest objective function score, or using a model assessment method such as Modeller’s own DOPE or GA341, or external programs such as PROSA or DFIRE

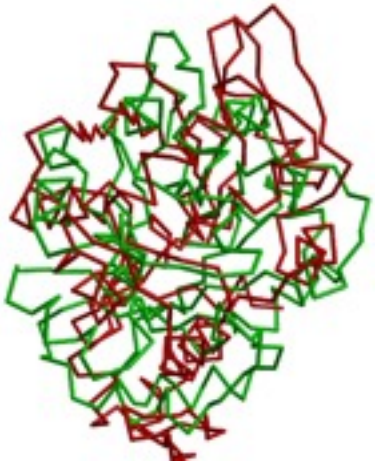
Typical errors in comparative models

MODEL

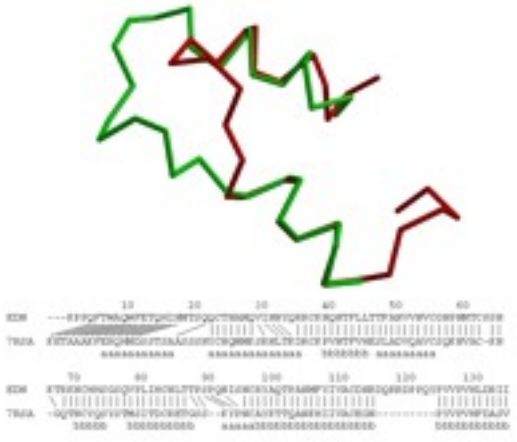
X-RAY

TEMPLATE

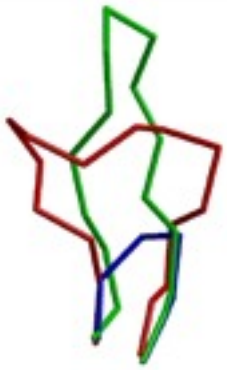
Incorrect template



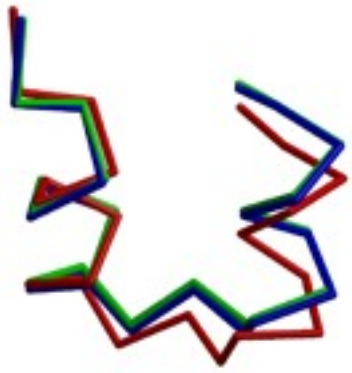
Misalignment



Region without a template



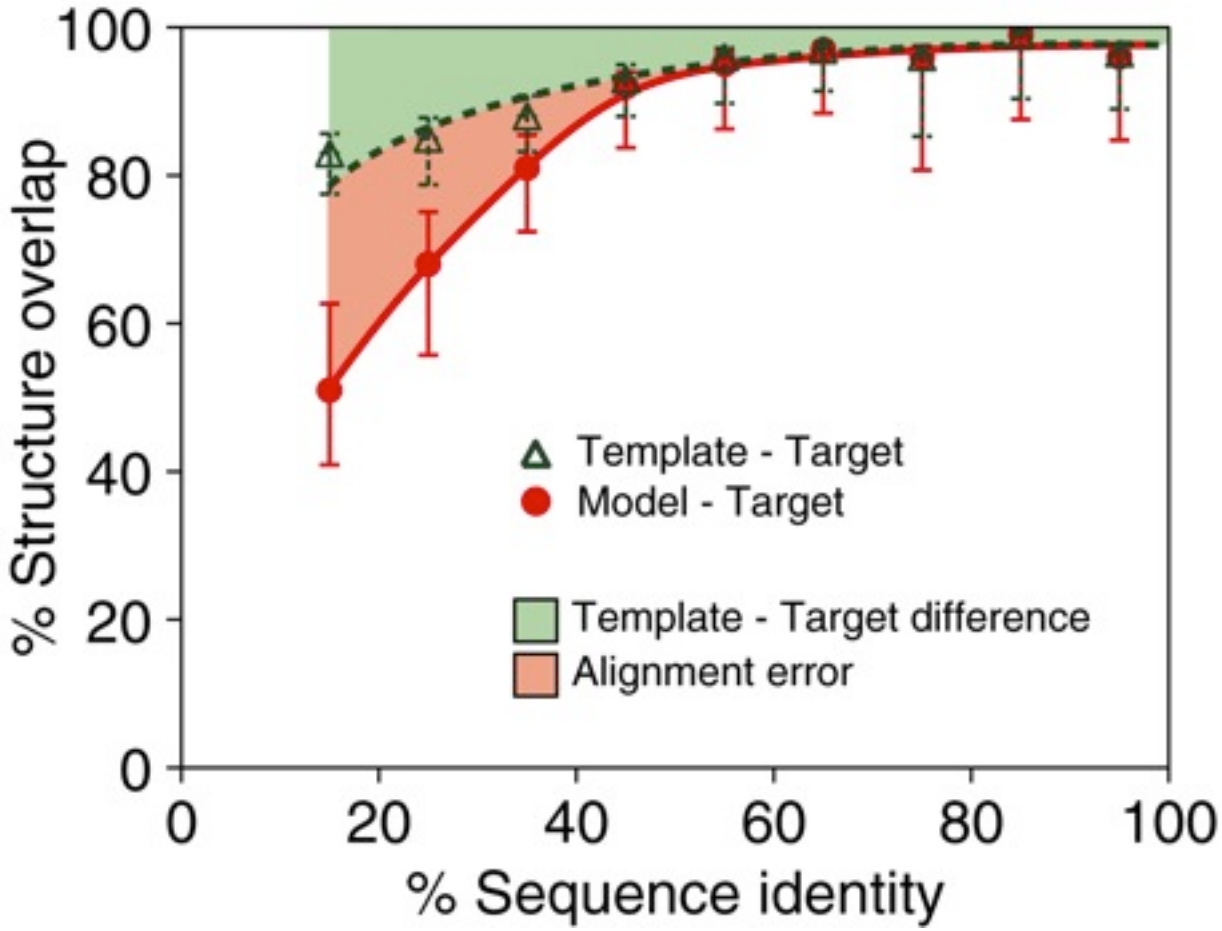
Distortion/shifts in aligned regions



Sidechain packing

Marti-Renom et al. Annu.Rev.Biophys.Biomol.Struct. 29, 291-325, 2000.

Model Accuracy as a Function of Target-Template Sequence Identity

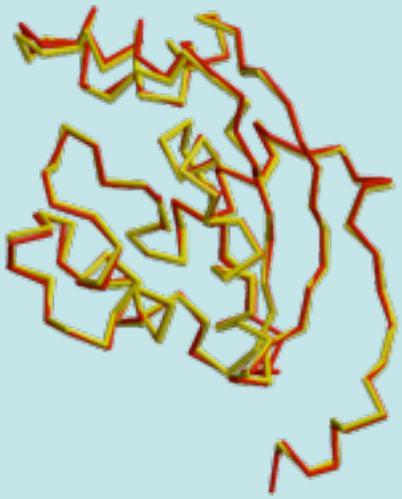


Sánchez, R., Šali, A. Proc Natl Acad Sci U S A. 95 pp13597-602. (1998).

Model accuracy

HIGH ACCURACY

NM23
Seq id 77%
Cα equiv 147/148
RMSD 0.41Å



Scope for improvement:
Sidechains

X-RAY / MODEL

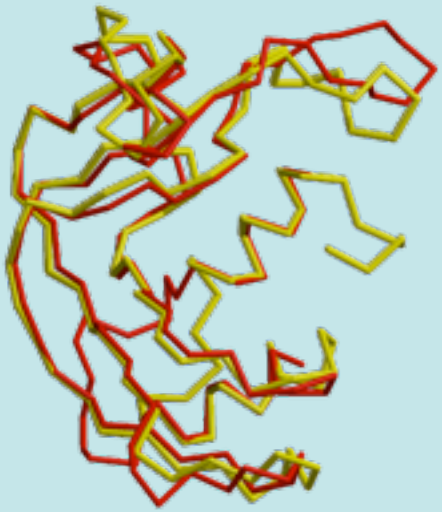
MEDIUM ACCURACY

CRABP
Seq id 41%
Cα equiv 122/137
RMSD 1.34Å

Sidechains
Core backbone
Loops

LOW ACCURACY

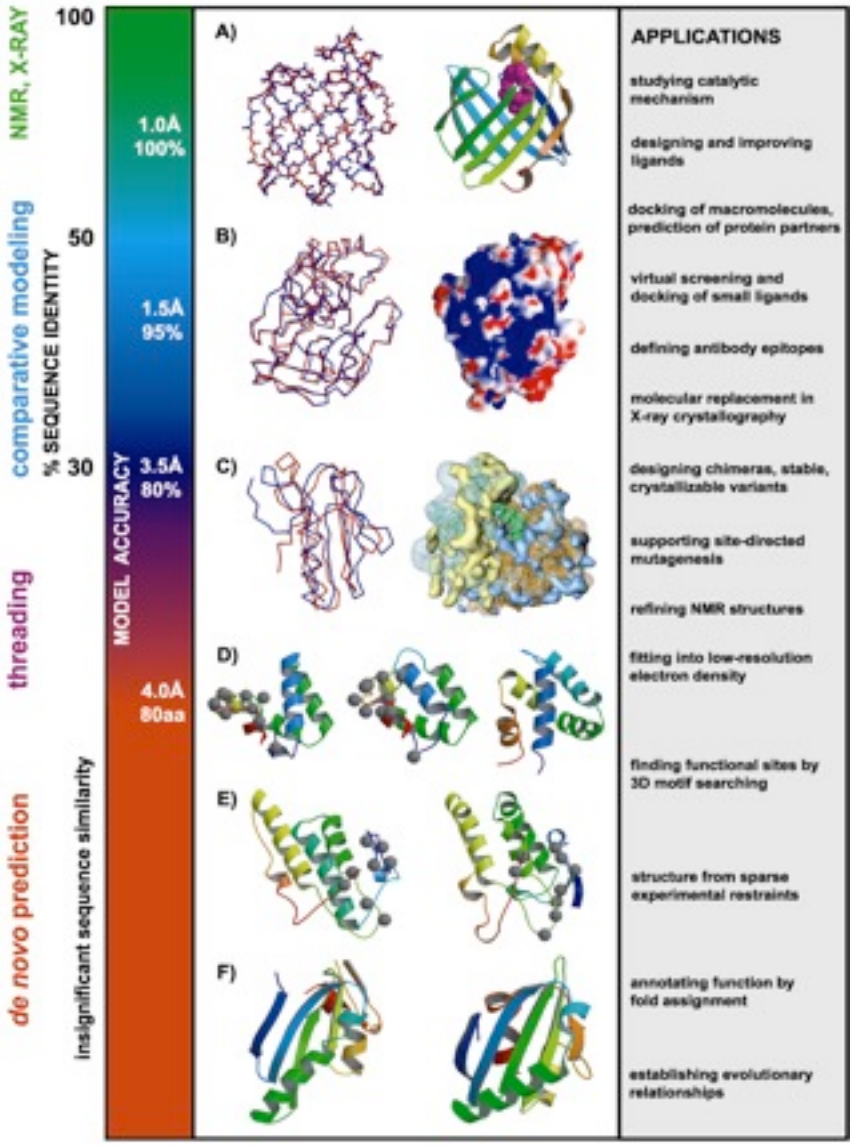
EDN
Seq id 33%
Cα equiv 90/134
RMSD 1.17Å



Sidechains
Core backbone, Loops
Alignment,
Fold assignment

Marti-Renom et al. Annu.Rev.Biophys.Biomol.Struct. 29, 291-325, 2000.

Applications of protein structure models

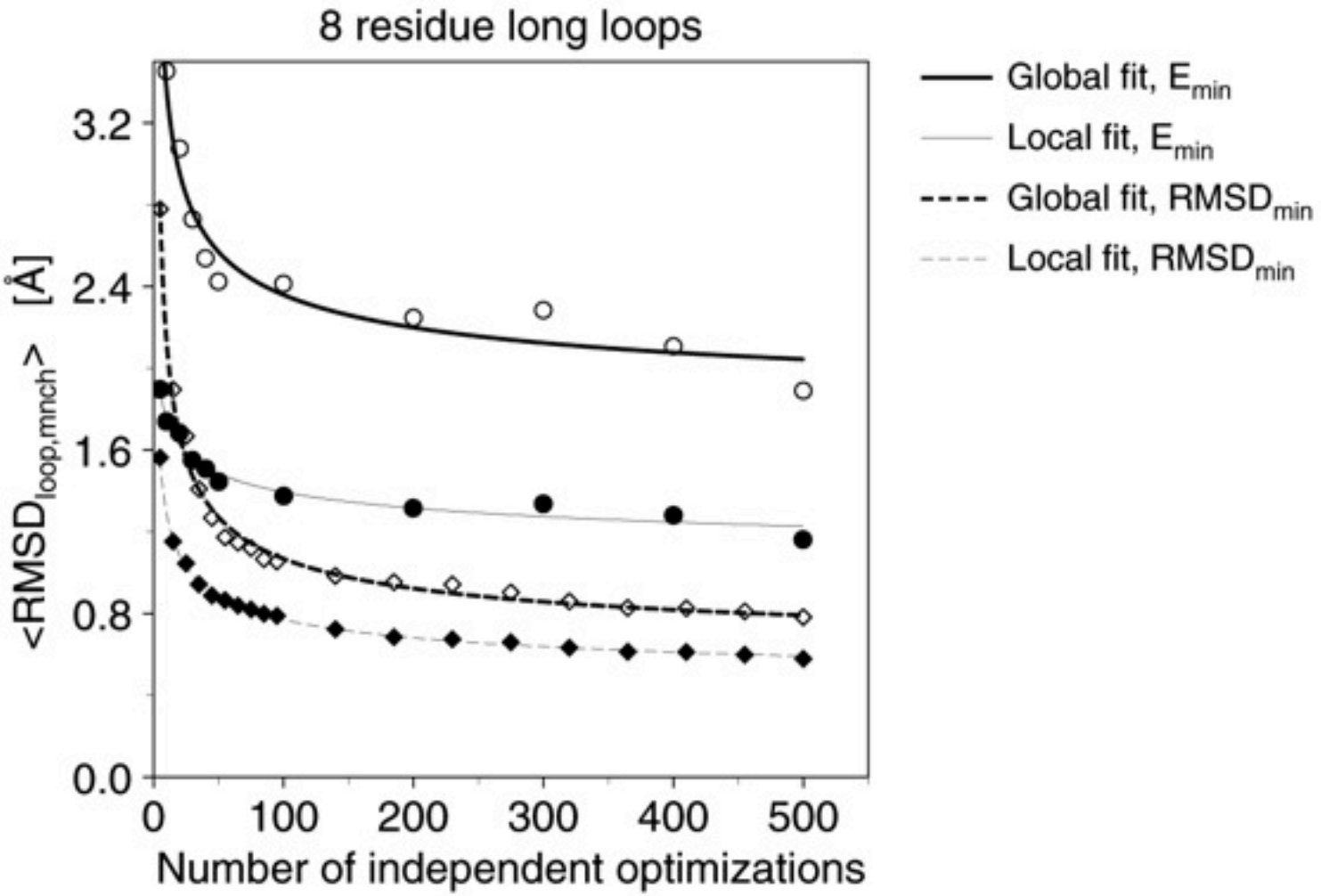


D. Baker & A. Sali. Science 294, 93, 2001.

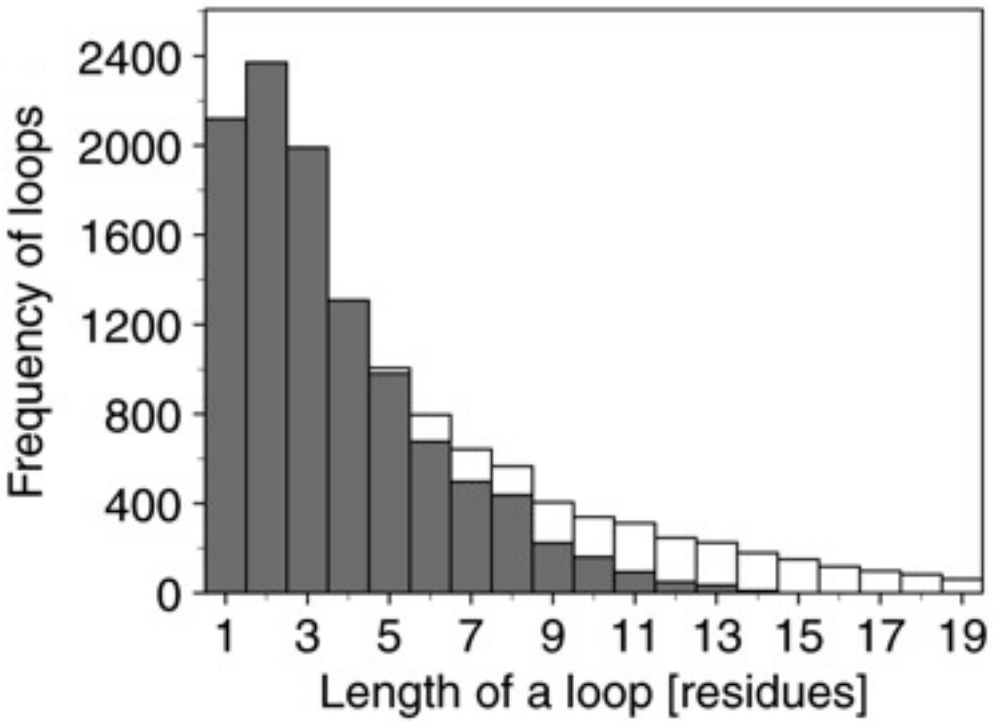
Loop modeling

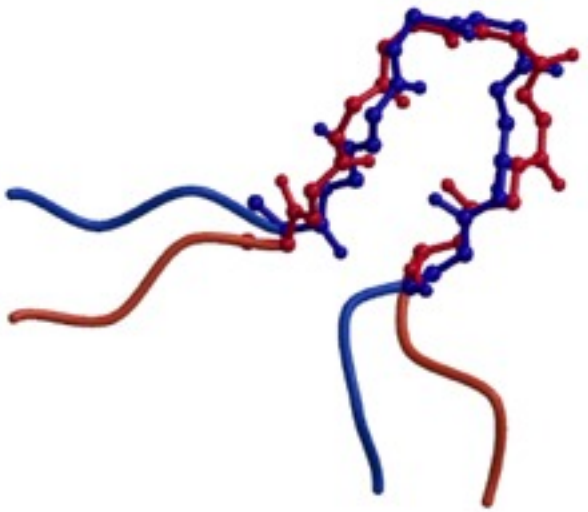
- Often, there are parts of the sequence which have no detectable templates
- “Mini folding problem” – these loops must be sampled to get improved conformations
- Database searches only complete for 4-6 residue loops
- Modeller uses conformational search with a custom energy function optimized for loop modeling (statistical potential derived from PDB)
  - Fiser/Melo protocol (‘loopmodel’)
  - Newer DOPE + GB/SA protocol (‘dope_loopmodel’)

Accuracy of loop models as a function of amount of optimization



Fraction of loops modeled with medium accuracy (<2Å)





Obtaining the Modeller software

- Modeller can be obtained from our website: http://salilab.org/modeller/
- Available for Mac, Windows, Linux and some other Unix systems
- Free for academics, but does need a license
- The website also links to more detailed tutorials, the online manual, users' mailing list, publications, etc.
- Example files for this tutorial can be found at http://salilab.org/modeller/erice.zip

Running Modeller

Running Modeller

- Modeller is actually a powerful library of functions and Python classes for handling protein structures and alignments

Running Modeller

- Modeller is actually a powerful library of functions and Python classes for handling protein structures and alignments
  - Pro: not just limited to comparative modeling; you can add your own functionality (e.g. custom energy terms) in C or Python, or use the Python module from other programs

Running Modeller

- Modeller is actually a powerful library of functions and Python classes for handling protein structures and alignments
  - Pro: not just limited to comparative modeling; you can add your own functionality (e.g. custom energy terms) in C or Python, or use the Python module from other programs
  - Pro: can also superpose structures, search sequence databases, fit against EM data, etc.

Running Modeller

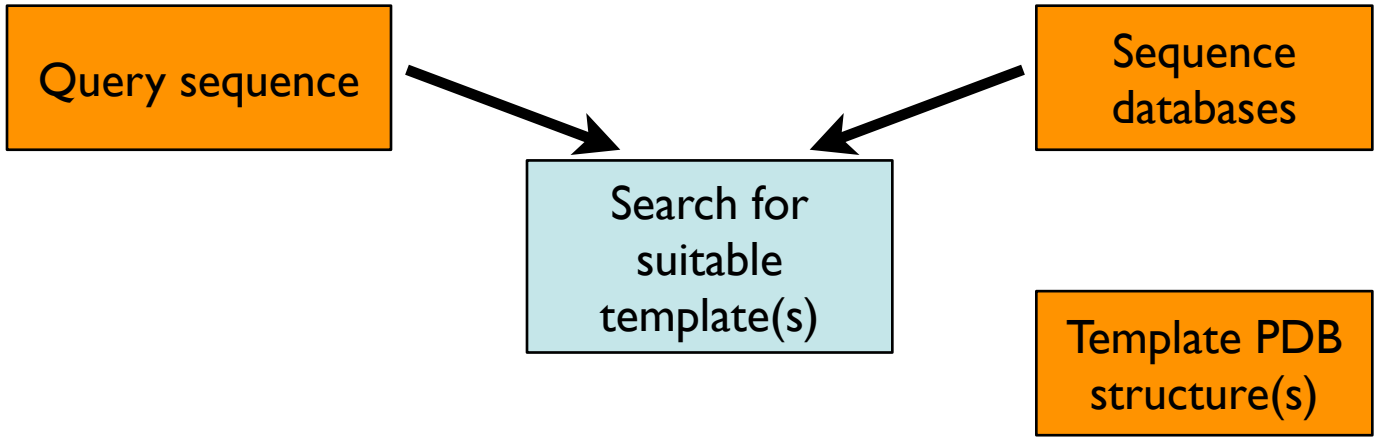
- Modeller is actually a powerful library of functions and Python classes for handling protein structures and alignments
  - Pro: not just limited to comparative modeling; you can add your own functionality (e.g. custom energy terms) in C or Python, or use the Python module from other programs
  - Pro: can also superpose structures, search sequence databases, fit against EM data, etc.
  - Con: there is no point and click interface; to build a model, you must write a short Python script…
    but for most applications, these scripts are very simple, and you can use the examples as your templates

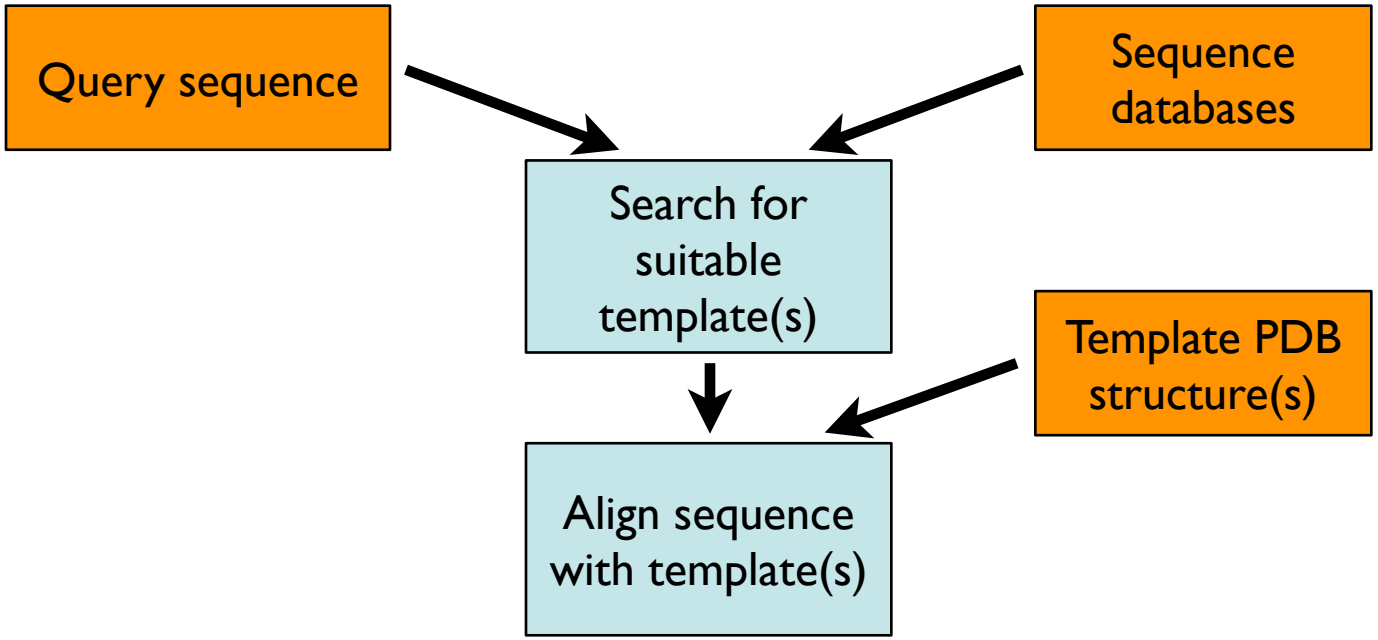
Query sequence

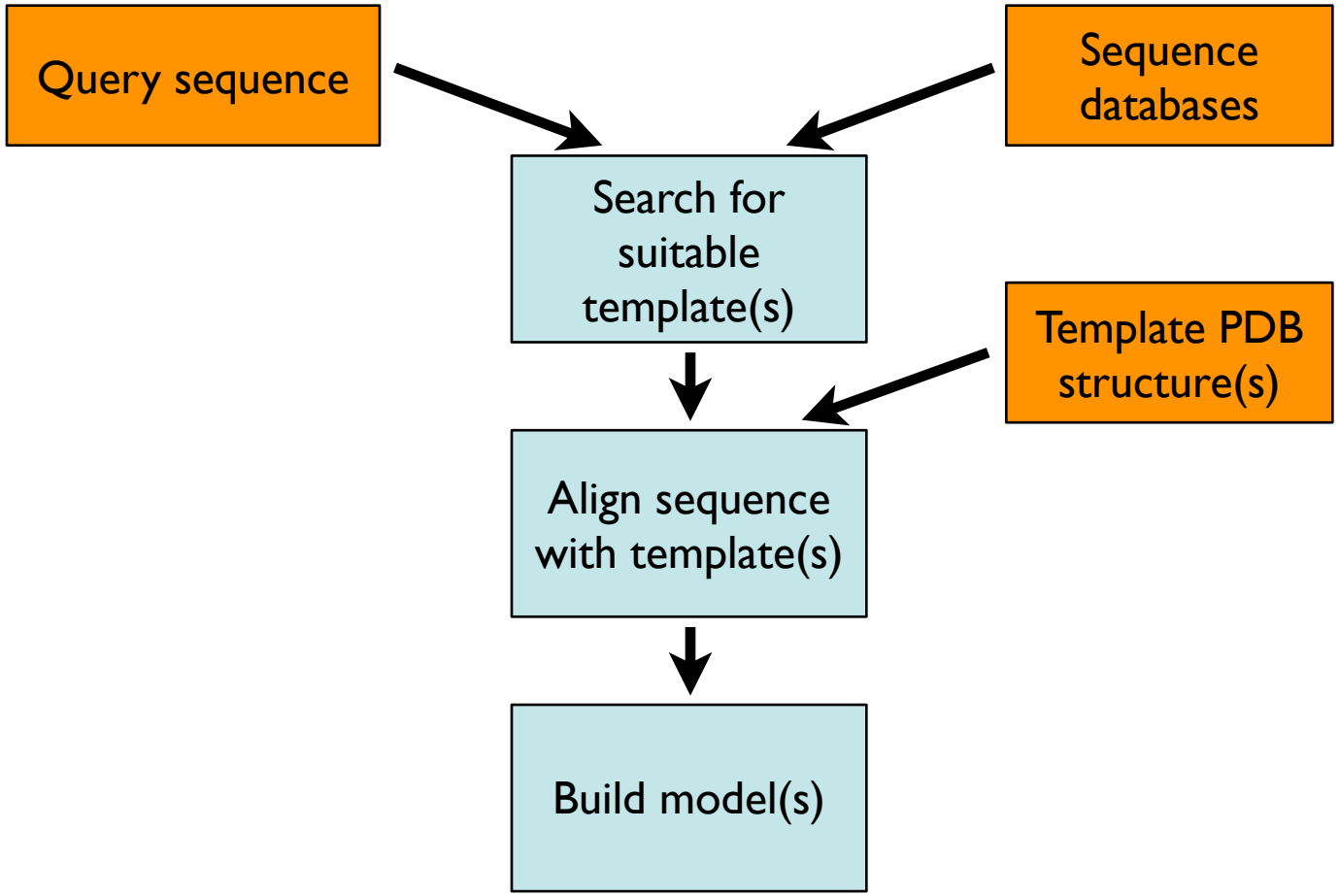
Query sequence

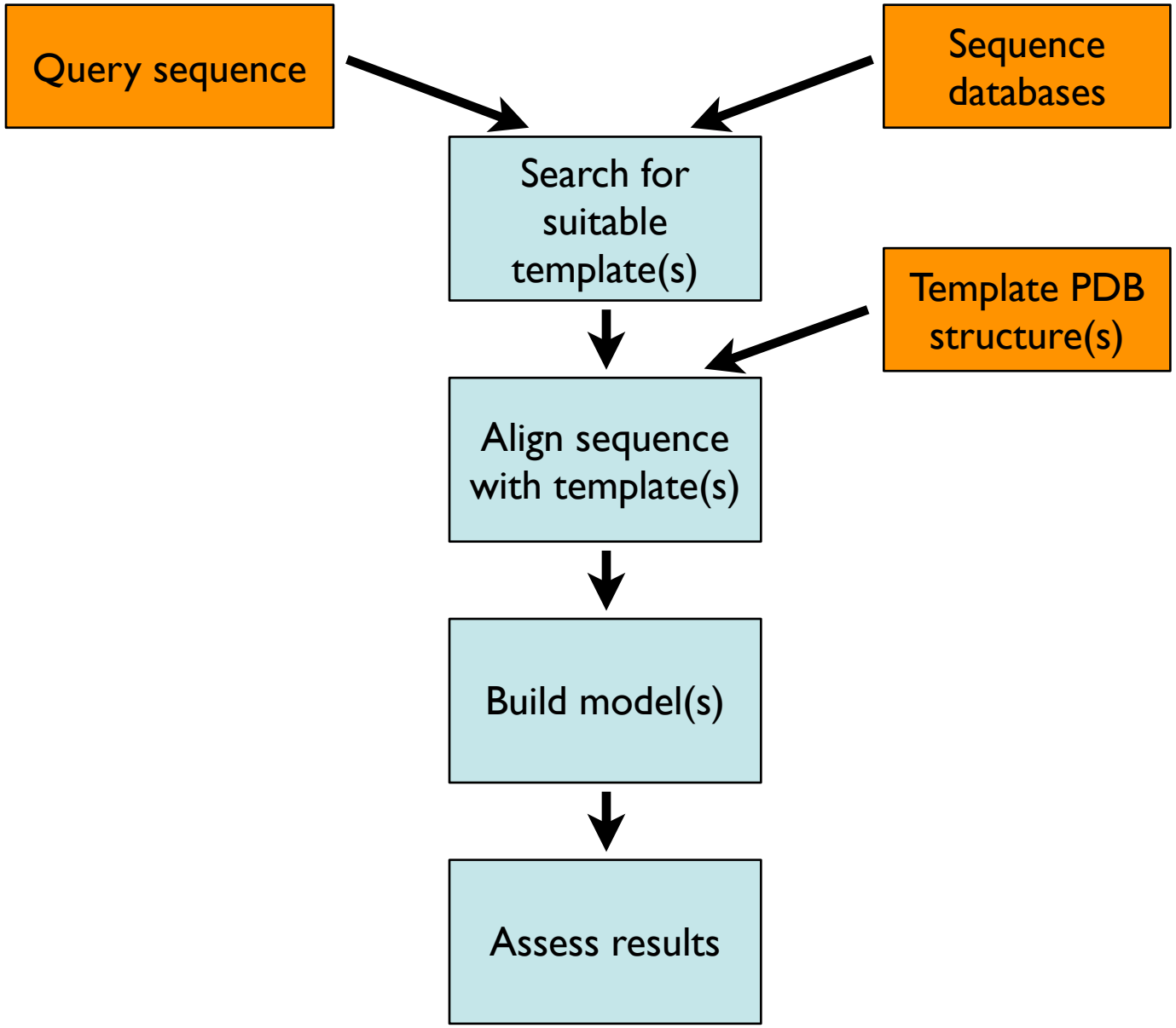
Sequence databases

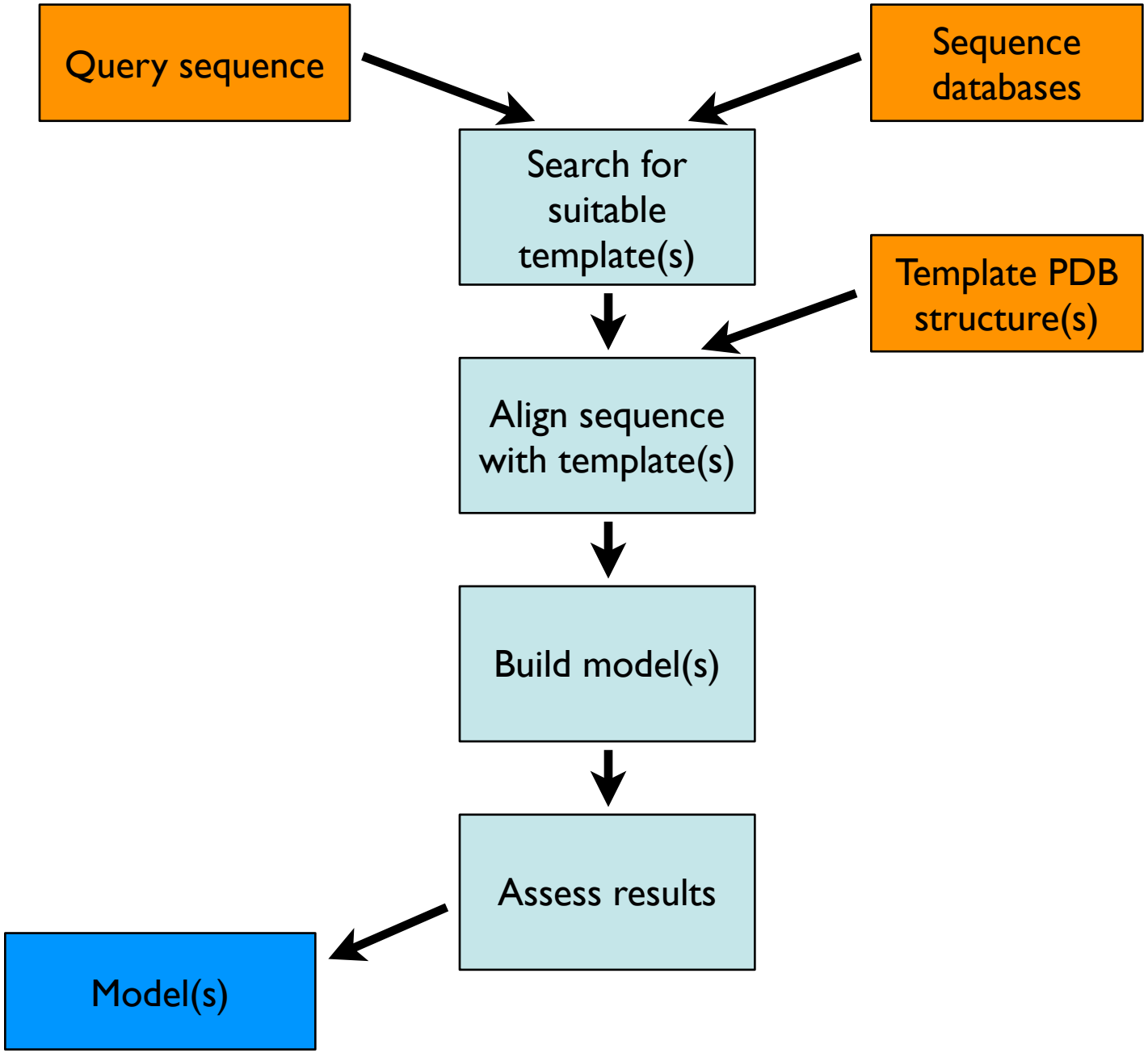
Template PDB structure(s)

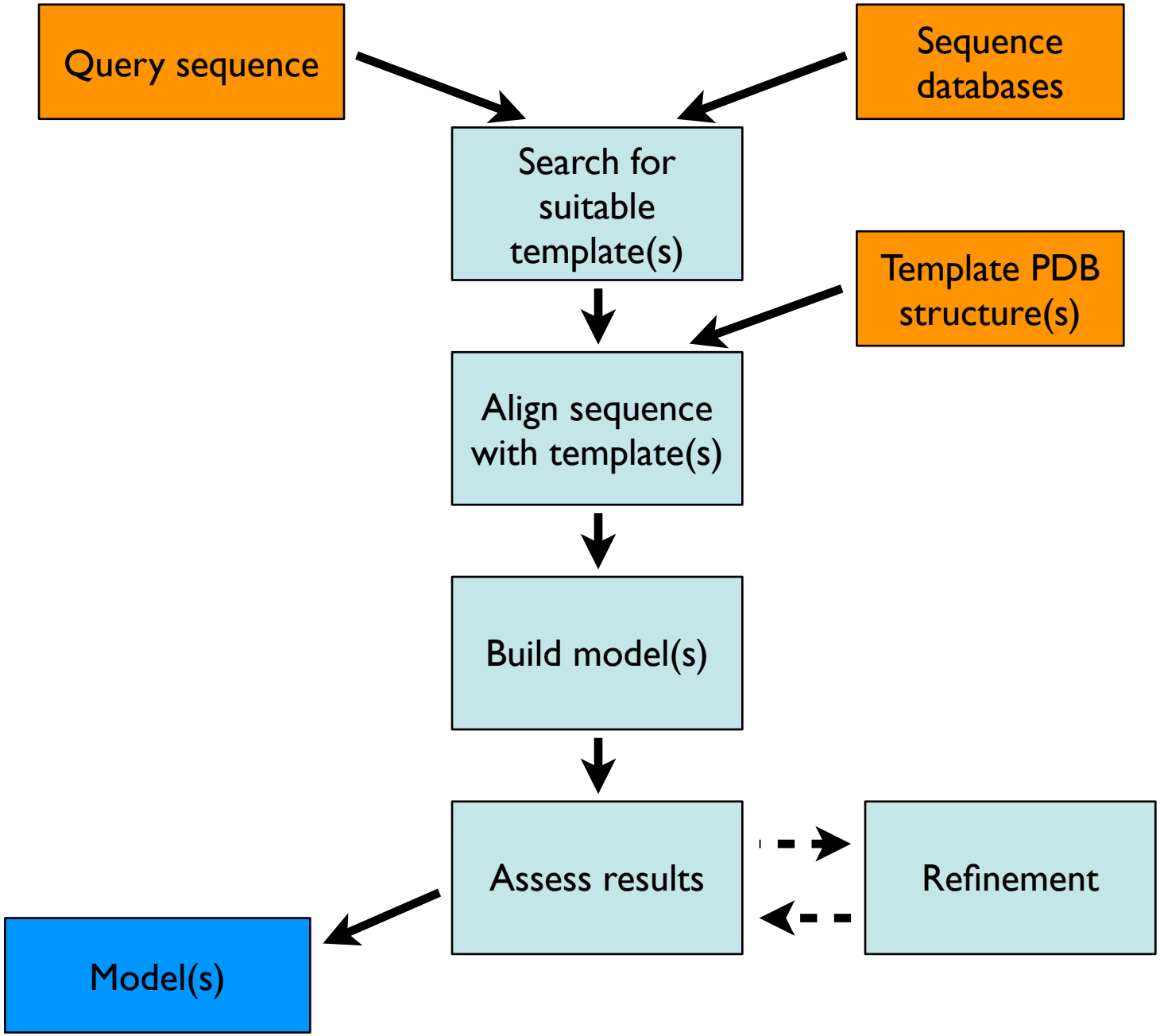

Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)

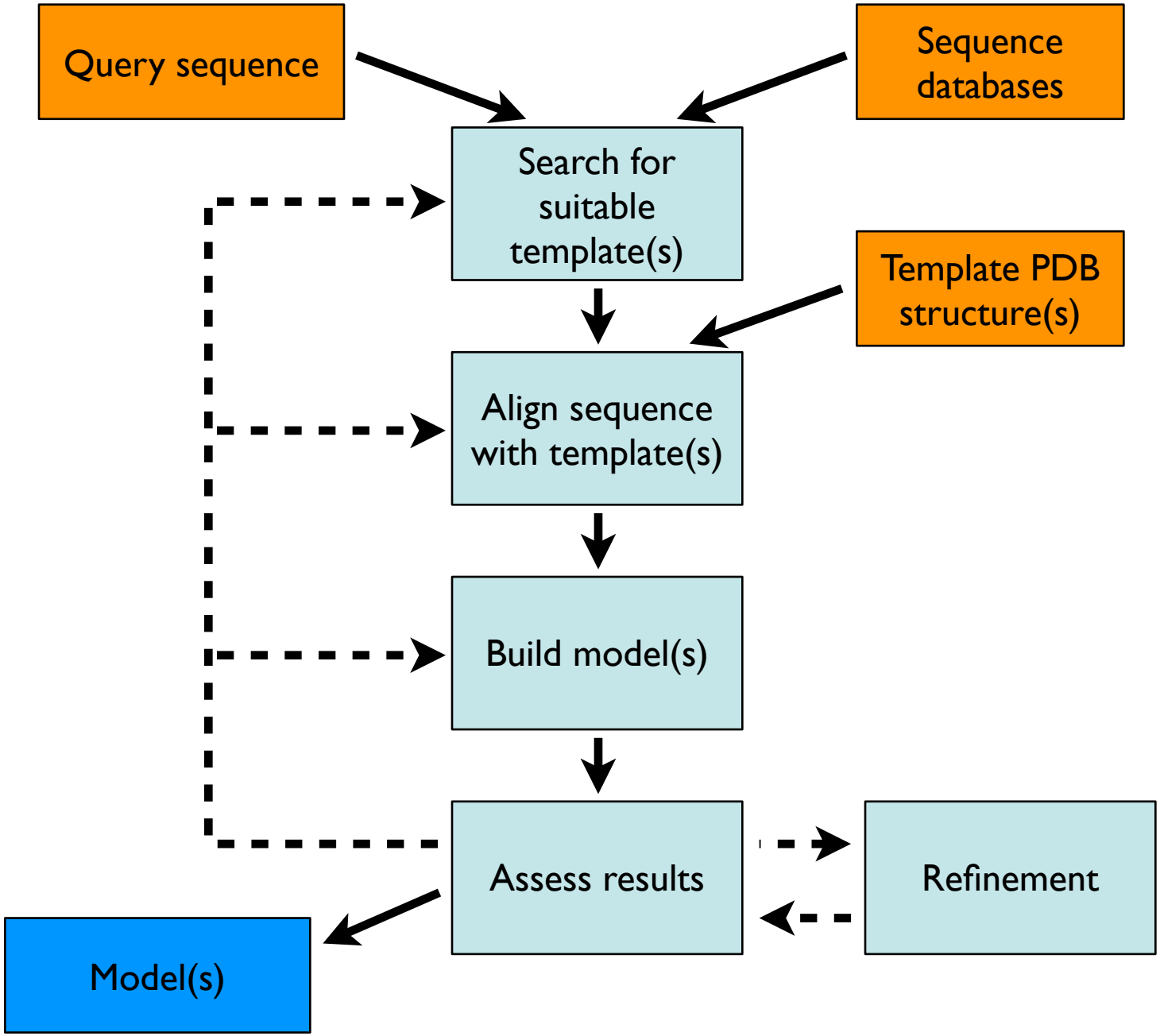

Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)

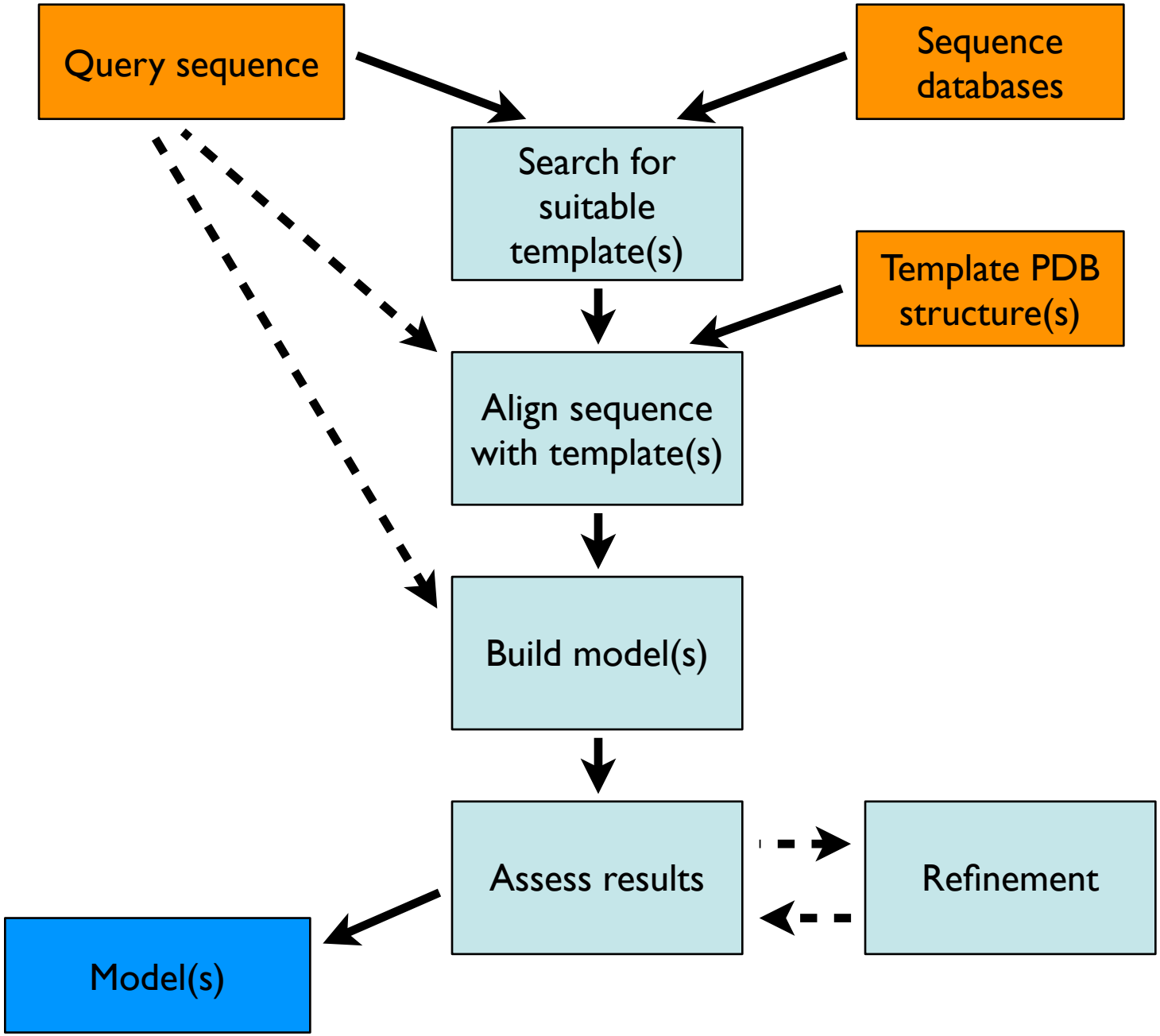

Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Model(s)


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

Prepare the query sequence

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- This is an ‘alignment’ file, even though it contains only one sequence

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- This is an ‘alignment’ file, even though it contains only one sequence
- Look up ‘alignment file format’ in the Modeller manual index for more information

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

'align code': a unique identifier for the sequence. Often PDB code + chain ID (e.g. 1xyzA)

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- This is an 'alignment' file, even though it contains only one sequence
- Look up 'alignment file format' in the Modeller manual index for more information

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

This line identifies that this is a sequence with unknown structure (more later)

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- This is an ‘alignment’ file, even though it contains only one sequence
- Look up ‘alignment file format’ in the Modeller manual index for more information

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

the amino acid sequence (case sensitive), terminated by a '*' character

- This is an 'alignme[illegible] sequence
- Look up 'alignment file format' in the Modeller manual index for more information

Prepare the query sequence

- Put the sequence in PIR format and save as a new file, TvLDH.ali:

```
>P1;TvLDH
sequence:::::::::
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- This is an ‘alignment’ file, even though it contains only one sequence
- Look up ‘alignment file format’ in the Modeller manual index for more information

Prepare the query sequence

Prepare the query sequence

- All Modeller input files are ‘plain text’

Prepare the query sequence

- All Modeller input files are ‘plain text’
- I always work from a terminal window and use ‘vi’ to edit my text files, but…

Prepare the query sequence

- All Modeller input files are ‘plain text’
- I always work from a terminal window and use ‘vi’ to edit my text files, but…
  - emacs works too

Prepare the query sequence

- All Modeller input files are ‘plain text’
- I always work from a terminal window and use ‘vi’ to edit my text files, but…
  - emacs works too
  - If you want to use a graphical text editor, make sure you save the file in plain text (not Unicode or Word .doc)

Prepare the query sequence

- All Modeller input files are ‘plain text’
- I always work from a terminal window and use ‘vi’ to edit my text files, but…
  - emacs works too
  - If you want to use a graphical text editor, make sure you save the file in plain text (not Unicode or Word .doc)
  - For Mac’s TextEdit application, use ‘Make Plain Text’ from the Format menu

Prepare the query sequence

- All Modeller input files are ‘plain text’
- I always work from a terminal window and use ‘vi’ to edit my text files, but…
  - emacs works too
  - If you want to use a graphical text editor, make sure you save the file in plain text (not Unicode or Word .doc)
  - For Mac’s TextEdit application, use ‘Make Plain Text’ from the Format menu
  - For Windows, use Notepad or be very careful to save in plain text otherwise (and watch out for Windows “helpfully” adding or hiding file extensions)

Prepare the databases

- Get `pdb_95.pir.gz` from the “Data file downloads” page at the Modeller website
- This is another ‘alignment’ file in PIR format
  - Contains the sequence for each structure in PDB, clustered to remove redundancy
  - No gaps, so not really an “alignment”

Excerpt of pdb_95.pir

```
>P1;205lA
structureX:205l:1    :A:162  :A:T4 LYSOZYME:OBACTERIA PHAGE T4: 2.10:-1.00
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTAAKSAAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTF
RTGTWDAYK*

>P1;830cA
structureX:830c:104  :A:271  :A:MMP-13:HOMO SAPIENS:1.60:-1.00
YNVFPRTLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHDGIADIMISFGIKEHGDFYPF
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

Excerpt of pdb_95.pir

This sequence corresponds to an X-ray structure

```
>P1;205lA
structureX:205l:1     :A:161
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTAAKSAAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTF
RTGTWDAYK*

>P1;830cA
structureX:830c:104   :A:271   :A:MMP-13:HOMO SAPIENS:1.60:-1.00
YNVFPRTLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHDGIADIMISFGIKEHGDFYPF
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

Excerpt of pdb_95.pir

```
>P1;205lA
structureX:205l:1     :A:162   :A:T4 LYSOZYME:OBACTERIA PHAGE T4: 2.10:-1.00
MNIFEMLRIDEGLRLKIYKDTEGYYT
LRNAKLKPVYDSLDAVRRAALINMVF
RTGTWDAYK*

>P1;830cA
structureX:830c:104   :A:271   :A:MMP-13:HOMO SAPIENS:1.60:-1.00
YNVFPRTLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHDGIADIMISFGIKEHGDFYPF
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

The corresponding structure can be found in the 205l PDB file

Excerpt of pdb_95.pir

```
>P1;2051A
structureX:2051:1     :A:162   :A:T4 LYSOZYME:OBACTERIA PHAGE T4: 2.10:-1.00
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTAAKSAAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTF
RTGTWDAYK*

>P1;830cA
structureX:830c:104   :A:271   :A:MMP-13:HOMO SAPIENS:1.60:-1.00
YNVFPRTLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHDGIADIMISFGIKEHGDFYPF
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

The sequence corresponds to residues 1 through 162 in the A chain of the PDB

Excerpt of pdb_95.pir

```
>P1;205lA
structureX:205l:1     :A:162   :A:T4 LYSOZYME:OBACTERIA PHAGE T4: 2.10:-1.00
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTAAKSAAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTF
RTGTWDAYK*

>P1;830cA
structureX:830c:104   :A:27
YNVFPRTLKWSKMNLTYRIVNYTPDM
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

Name, organism, resolution and R-factor (if known) of the structure

Excerpt of pdb_95.pir

```
>P1;205lA
structureX:205l:1    :A:162  :A:T4 LYSOZYME:OBACTERIA PHAGE T4: 2.10:-1.00
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTAAKSAAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTF
RTGTWDAYK*

>P1;830cA
structureX:830c:104  :A:271  :A:MMP-13:HOMO SAPIENS:1.60:-1.00
YNVFPRTLKWSKMNLTYRIVNYTPDMTHSEVEKAFKKAFKVWSDVTPLNFTRLHDGIADIMISFGIKEHGDFYPF
DGPSGLLAHAFPPGPNYGGDAHFDDDETWTSSSKGYNLFLVAAHEFGHSLGLDHSKDPGALMFPIYTYTFMLPDD
DVQGIQSLYGPGDE*

(etc.)
```

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Prepare the databases

- Our first Modeller script, `convert_db.py`:

Load the Modeller Python module

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

Request verbose output from Modeller in the log file

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

Create a new Modeller environment. Most other Modeller objects need an environment.

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

Create a new Modeller sequence database (initially empty)

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

- Just a regular
- Converts the p
  which is faster

Convert the existing pdb_95.pir database from PIR format to binary format (Modeller is smart enough to automatically uncompress pdb_95.pir.gz first). Discard sequences shorter than 30 or longer than 4000 residues, and clean the sequences by removing non-standard residues.

Prepare the databases

- Our first Modeller script, `convert_db.py`:

```
from modeller import *

log.verbose()
env = environ()

sdb = sequence_db(env)
sdb.convert(seq_database_file='pdb_95.pir',
            seq_database_format='PIR',
            chains_list='ALL', minmax_db_seq_len=(30, 4000),
            clean_sequences=True, outfile='pdb_95.hdf5')
```

- Just a regular Python script
- Converts the pdb_95.pir file into a binary equivalent, which is faster to use (only need to do this once)

Running the script

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file
- Open a terminal window, go to the directory containing the sequence and script, and run the script with

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file
- Open a terminal window, go to the directory containing the sequence and script, and run the script with

```
python convert_db.py > convert_db.log
```

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file
- Open a terminal window, go to the directory containing the sequence and script, and run the script with

```
python convert_db.py > convert_db.log
```

- Alternatively, if Python is not set up on your system, you can run the Modeller binary directly (it includes a copy of Python 2.3):

Running the script

- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file
- Open a terminal window, go to the directory containing the sequence and script, and run the script with

```
python convert_db.py > convert_db.log
```

- Alternatively, if Python is not set up on your system, you can run the Modeller binary directly (it includes a copy of Python 2.3):

```
mod9v8 convert_db.py
```

Running the script

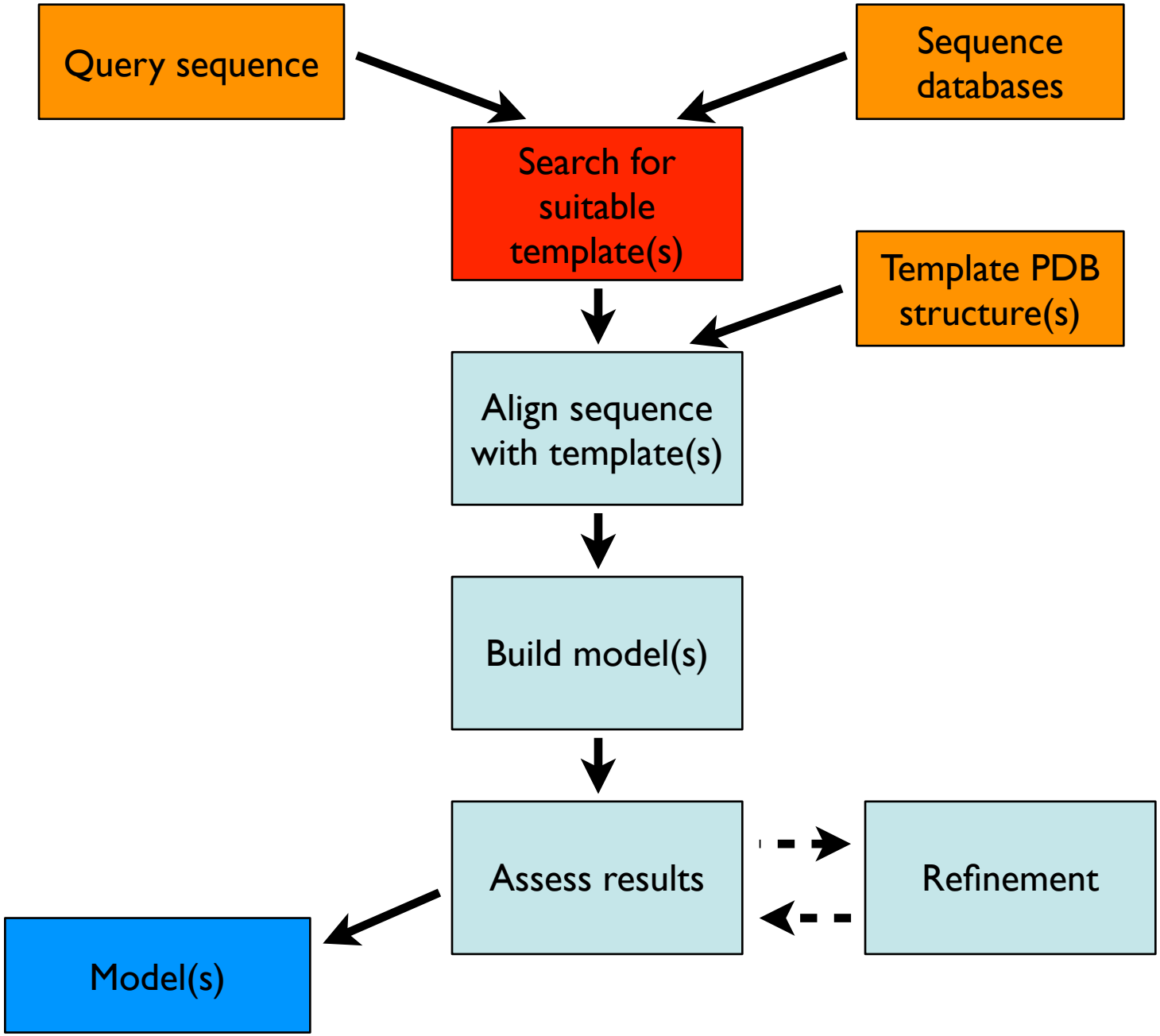
- Put the Python script, sequence file and sequence database in the same directory/folder
- Both Modeller and Python are very strict about syntax – check that you have commas, brackets etc. in the right places in your script and sequence file
- Open a terminal window, go to the directory containing the sequence and script, and run the script with

```
python convert_db.py > convert_db.log
```

- Alternatively, if Python is not set up on your system, you can run the Modeller binary directly (it includes a copy of Python 2.3):

```
mod9v8 convert_db.py
```

- Any errors (Python exceptions) come out on standard error


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

Search for template(s)

- Read in the query sequence (TvLDH.ali) and search for similar sequences in the pdb_95 database
- Use Modeller's profile.build() method
  - Similar to BLAST, but uses rigorous dynamic programming
  - Widens the single sequence into a 'profile' containing the sequence and similar sequences (e.g. homologs)
  - This is a simple 'sequence-sequence' search
  - Note that profiles are sensitive than sequences for searching since they provide more information (rather than "first residue is a G" they say "first residue is 90% likely to be a G, but 10% to be an A")

- build_profile.py:

```
from modeller import *

log.verbose()
env = environ()

# Read in the sequence database in binary format
sdb = sequence_db(env, seq_database_file='pdb_95.hdf5',
                  seq_database_format='BINARY', chains_list='ALL')

# Read in the target sequence in PIR alignment format
aln = alignment(env)
aln.append(file='TvLDH.ali', alignment_format='PIR', align_codes='ALL')

# Convert the input sequence "alignment" into a one-sequence profile
prf = aln.to_profile()

# Scan sequence database to pick up similar sequences and add them to the profile
prf.build(sdb, matrix_offset=-450, rr_file='${LIB}/blosum62.sim.mat',
          gap_penalties_1d=(-500, -50), n_prof_iterations=1,
          check_profile=False, max_aln_evalue=0.01)

# Write out the profile in text format
prf.write(file='build_profile.prf', profile_format='TEXT')
```

Excerpt of build_profile.prf

```
# Number of sequences:     41
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -500.0   -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S     0   335     1   335     0     0     0    0.     0.0
    2 1a5zA      X     1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X     1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X     1   327     7   331     6   325   316   42.    0.0
    5 1civA      X     1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X     1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X     1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X     1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X     1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X     1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X     1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X     1   321    64   239    53   234   164   26.   0.45E-03
   24 1lldA      X     1   313    13   242     9   233   216   31.   0.72E-07
   25 5mdhA      X     1   333     2   332     1   331   328   44.    0.0
   26 7mdhA      X     1   351     6   334    14   339   325   34.    0.0
   27 1mldA      X     1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X     1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X     1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X     1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X     1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X     1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X     1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X     1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X     1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X     1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X     1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X     1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X     1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X     1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X     1   327     1   325     1   319   318   45.    0.0
```

Excerpt of build_profile.prf

S = sequence, X = sequence with structure

```
# Number of sequences:      41
# Length of profile  :     335
# N_PROF_ITERATIONS  :       1
# GAP_PENALTIES_1D   :    -500.0    -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S      0   335     1   335     0     0     0    0.     0.0
    2 1a5zA      X      1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X      1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X      1   327     7   331     6   325   316   42.    0.0
    5 1civA      X      1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X      1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X      1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X      1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X      1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X      1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X      1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X      1   321    64   239    53   234   164   26.   0.45E-03
   24 1lldA      X      1   313    13   242     9   233   216   31.   0.72E-07
   25 5mdhA      X      1   333     2   332     1   331   328   44.    0.0
   26 7mdhA      X      1   351     6   334    14   339   325   34.    0.0
   27 1mldA      X      1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X      1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X      1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X      1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X      1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X      1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X      1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X      1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X      1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X      1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X      1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X      1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X      1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X      1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X      1   327     1   325     1   319   318   45.    0.0
```

Excerpt of build_profile.prf

```
# Number of sequences:     41
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -500.0   -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S     0   335     1   335     0     0     0    0.    0.0
    2 1a5zA      X     1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X     1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X     1   327     7   331     6   325   316   42.    0.0
    5 1civA      X     1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X     1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X     1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X     1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X     1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X     1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X     1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X     1   321    64   239    53   23
   24 1lldA      X     1   313    13   242     9   23
   25 5mdhA      X     1   333     2   332     1   33
   26 7mdhA      X     1   351     6   334    14   33
   27 1mldA      X     1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X     1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X     1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X     1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X     1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X     1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X     1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X     1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X     1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X     1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X     1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X     1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X     1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X     1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X     1   327     1   325     1   319   318   45.    0.0
```

Position and length of the match in the two sequences

Excerpt of build_profile.prf

```
# Number of sequences:     41
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -500.0   -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S     0   335     1   335     0     0     0    0.     0.0
    2 1a5zA      X     1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X     1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X     1   327     7   331     6   325   316   42.    0.0
    5 1civA      X     1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X     1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X     1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X     1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X     1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X     1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X     1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X     1   321    64   239    53   234   164   26.   0.45E-03
   24 1lldA      X     1   313    13   242     9   233   216   31.   0.72E-07
   25 5mdhA      X     1   333     2   332     1   331   328   44.    0.0
   26 7mdhA      X     1   351     6   334    14   339   325   34.    0.0
   27 1mldA      X     1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X     1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X     1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X     1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X     1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X     1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X     1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X     1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X     1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X     1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X     1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X     1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X     1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X     1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X     1   327     1   325     1   319   318   45.    0.0
```

Sequence identity

Excerpt of build_profile.prf

```
# Number of sequences:     41
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -500.0   -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S     0   335     1   335     0     0     0    0.     0.0
    2 1a5zA      X     1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X     1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X     1   327     7   331     6   325   316   42.    0.0
    5 1civA      X     1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X     1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X     1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X     1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X     1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X     1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X     1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X     1   321    64   239    53   234   164   26.   0.45E-03
   24 1lldA      X     1   313    13   242     9   233   216   31.   0.72E-07
   25 5mdhA      X     1   333     2   332     1   331   328   44.    0.0
   26 7mdhA      X     1   351     6   334    14   339   325   34.    0.0
   27 1mldA      X     1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X     1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X     1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X     1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X     1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X     1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X     1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X     1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X     1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X     1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X     1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X     1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X     1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X     1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X     1   327     1   325     1   319   318   45.    0.0
```

e-value of the match

Excerpt of build_profile.prf

```
# Number of sequences:     41
# Length of profile  :    335
# N_PROF_ITERATIONS  :      1
# GAP_PENALTIES_1D   :   -500.0   -50.0
# MATRIX_OFFSET      : -450.0
# RR_FILE            : ${LIB}/blosum62.sim.mat
    1 TvLDH      S     0   335     1   335     0     0     0    0.     0.0
    2 1a5zA      X     1   312    75   242    63   229   164   28.   0.19E-07
    3 2a92A      X     1   316     8   191     6   186   174   26.   0.41E-04
    4 1b8pA      X     1   327     7   331     6   325   316   42.    0.0
    5 1civA      X     1   374     6   334    33   358   325   35.    0.0
    6 2cmdA      X     1   312     7   320     3   303   289   27.   0.37E-05
    7 3d5tA      X     1   321     4   325     3   316   312   44.    0.0
    8 2dfdA      X     1   314     5   198     2   190   186   26.   0.60E-06
    9 2ewdA      X     1   317     1   301     1   288   270   26.   0.33E-07
   10 1ez4B      X     1   318    69   239    55   230   159   31.   0.79E-06
   11 1guzA      X     1   305    13   301     8   280   265   25.   0.65E-08
...
   23 1llcA      X     1   321    64   239    53   234   164   26.   0.45E-03
   24 1lldA      X     1   313    13   242     9   233   216   31.   0.72E-07
   25 5mdhA      X     1   333     2   332     1   331   328   44.    0.0
   26 7mdhA      X     1   351     6   334    14   339   325   34.    0.0
   27 1mldA      X     1   313     5   198     1   189   183   26.   0.13E-05
   28 1o6zA      X     1   303     7   320     3   287   278   26.   0.61E-05
   29 1oc4A      X     1   315     5   191     4   186   174   28.   0.41E-04
   30 1ojuA      X     1   294    78   320    68   285   218   28.   0.99E-05
   31 1pzgA      X     1   327    74   191    71   190   114   30.   0.37E-06
   32 1smkA      X     1   313     7   202     4   198   188   34.    0.0
   33 1sovA      X     1   316    81   256    76   248   160   27.   0.21E-02
   34 1t2dA      X     1   315     5   256     4   250   238   25.   0.15E-03
   35 1u5aA      X     1   306    97   256    84   240   155   26.   0.92E-03
   36 1ur5A      X     1   299    13   191     9   171   158   31.   0.57E-02
   37 1uxkA      X     1   296    13   191    10   169   155   32.   0.95E-02
   38 1v6aA      X     1   332    94   300   103   304   197   25.   0.28E-03
   39 2v65A      X     1   326    94   300    97   298   197   24.   0.59E-03
   40 1y6jA      X     1   289    77   191    58   167   109   33.   0.75E-05
   41 1y7tA      X     1   327     1   325     1   319   318   45.    0.0
```

Choose a template

- Simplest option: choose one of those from the build_profile output that has a ‘zero’ e-value, i.e. 1b8pA, 1civA, 3d5tA, 5mdhA, 7mdhA, 1smkA, or 1y7tA
- Modeller has a “compare structures” function we can use to compare the templates with each other (not with the unknown sequence) to aid in picking the best template

- compare.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

- compare.py:

Look for PDB files first in the current directory, then the 'atom_files' subdirectory

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

- compare.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

Read a single chain from each PDB file

- compare.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

Add the corresponding sequence to the alignment

- compare.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

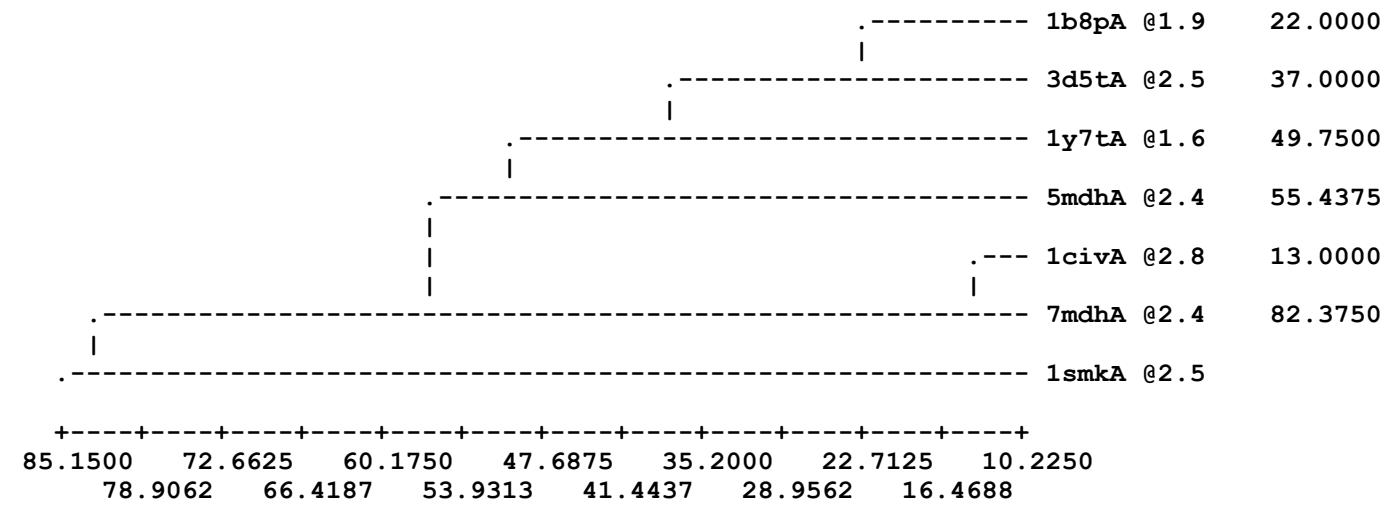
Use a simple sequence alignment as a starting point for a structure alignment

- compare.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

# Make a simple 1:1 alignment of 7 template structures
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1y7t', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('3d5t', 'A'),
                     ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)

# Sequence alignment
aln.malign()

# Structure alignment
aln.malign3d()

# Report details of the sequence/structure alignment
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

Excerpt from log file

```
Sequence identity comparison (ID_TABLE):
   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

         1b8pA @11y7tA @11civA @25mdhA @27mdhA @23d5tA @21smkA @2
1b8pA @1      327     201     146     152     152     249      50
1y7tA @1       61     327     158     170     160     210      58
1civA @2       45      48     374     140     304     148      55
5mdhA @2       46      52      42     333     140     164      58
7mdhA @2       46      49      87      42     351     148      50
3d5tA @2       78      65      46      51      46     321      50
1smkA @2       16      19      18      19      16      16     313
```

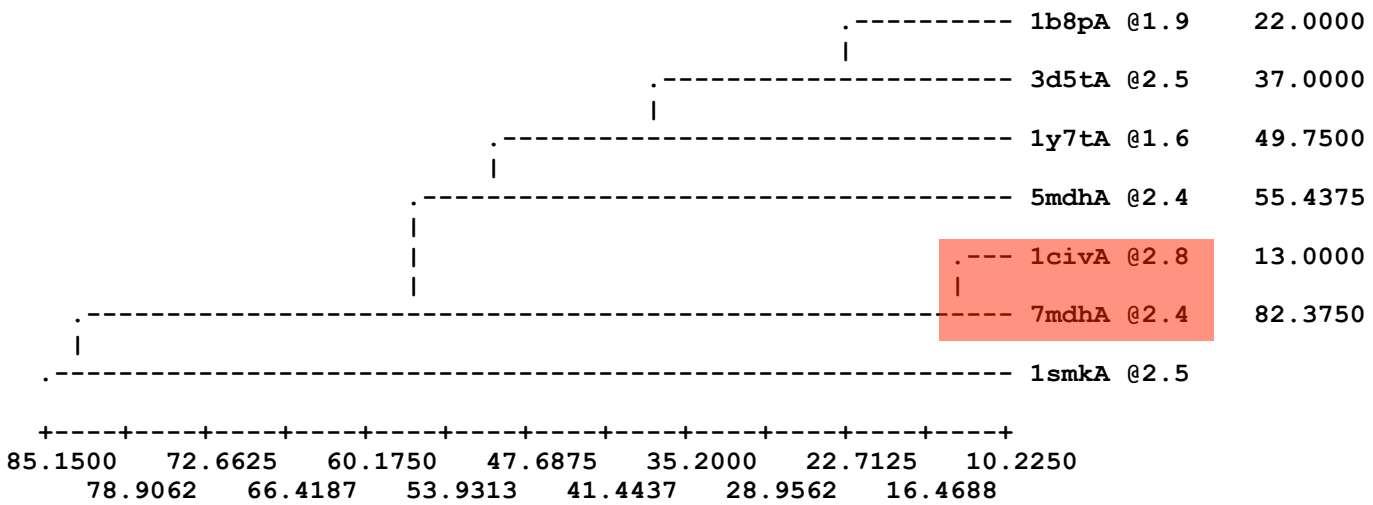


Excerpt from log file

1civA and 7mdhA are nearly identical (both sequence and structure) but 7mdhA is a higher resolution structure - eliminating 1civA

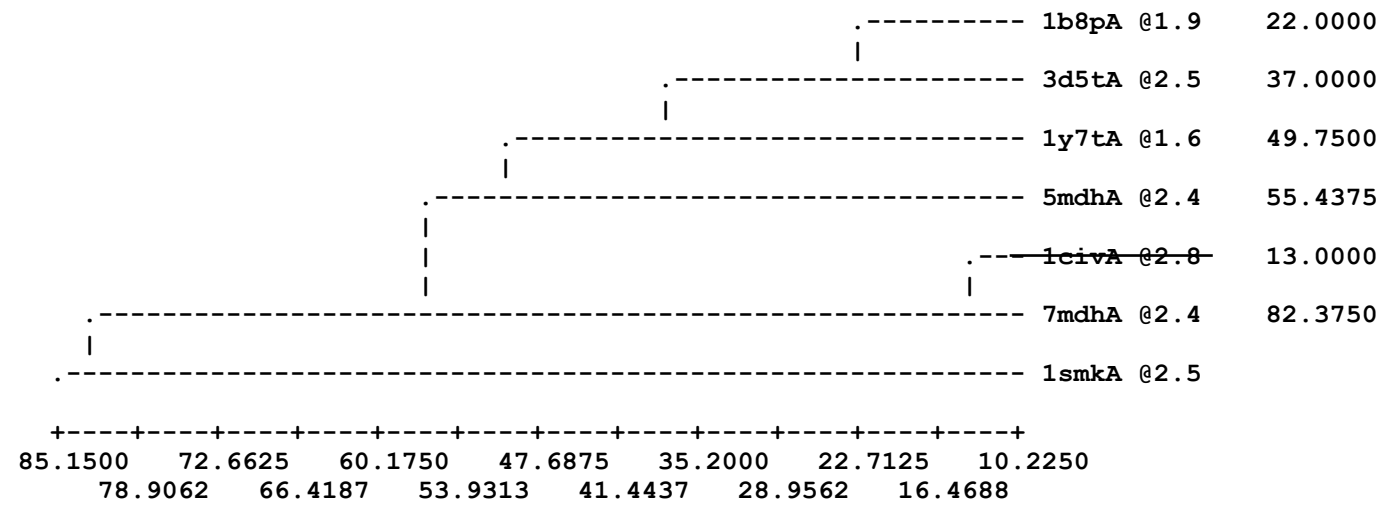
```
Sequence identity comparison (ID_TABLE)
   Diagonal       ... number of residue
   Upper triangle ... number of identic
   Lower triangle ... % sequence identi
```

	1b8pA @1	1y7tA @1	1civA @2	5mdhA @2	7mdhA @2	3d5tA @2	1smkA @2
1b8pA @1	327	201	146	152	152	249	50
1y7tA @1	61	327	158	170	160	210	58
1civA @2	45	48	374	140	304	148	55
5mdhA @2	46	52	42	333	140	164	58
7mdhA @2	46	49	87	42	351	148	50
3d5tA @2	78	65	46	51	46	321	50
1smkA @2	16	19	18	19	16	16	313



Excerpt from log file

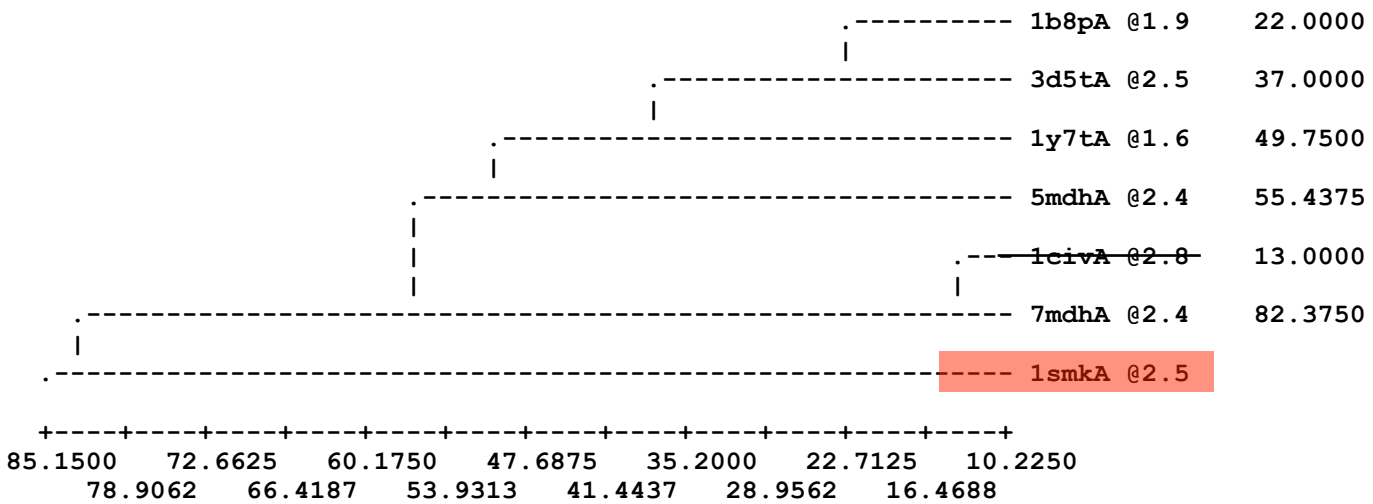
```
Sequence identity comparison (ID_TABLE):
   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

         1b8pA @11y7tA @11civA @25mdhA @27mdhA @23d5tA @21smkA @2
1b8pA @1      327     201     146     152     152     249      50
1y7tA @1       61     327     158     170     160     210      58
1civA @2       45      48     374     140     304     148      55
5mdhA @2       46      52      42     333     140     164      58
7mdhA @2       46      49      87      42     351     148      50
3d5tA @2       78      65      46      51      46     321      50
1smkA @2       16      19      18      19      16      16     313
```



Excerpt from log file

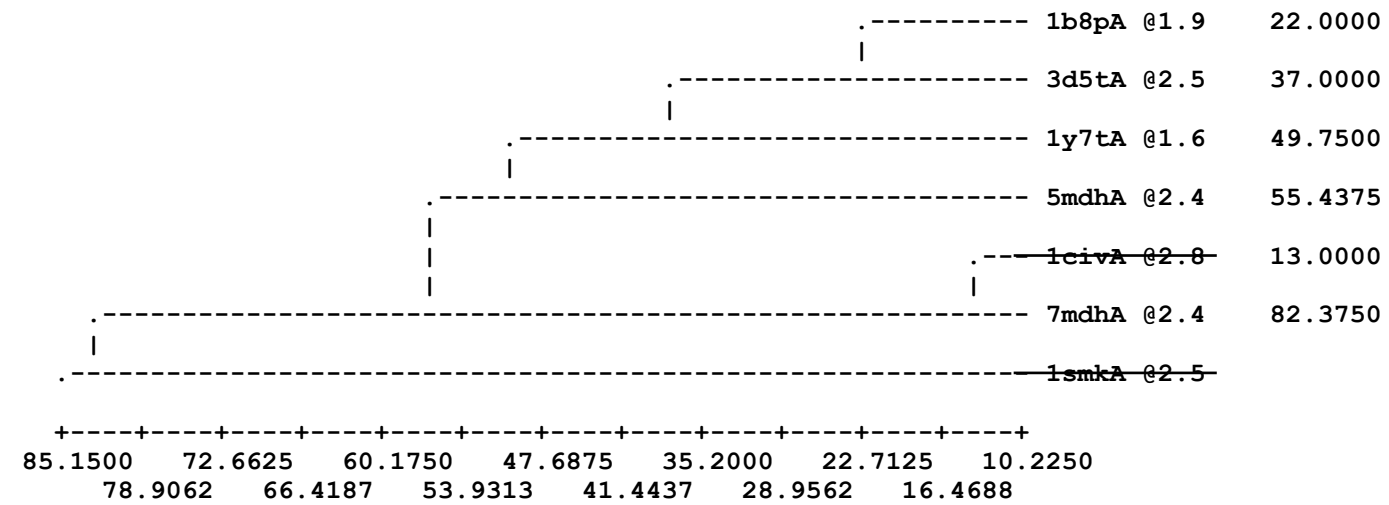
```
Sequence identity comparison (ID_TABLE):
   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

          1b8pA @11y7tA @11civA @25mdhA
1b8pA @1      327     201     146      1
1y7tA @1       61     327     158      1
1civA @2       45      48     374      1
5mdhA @2       46      52      42    333     140     164      58
7mdhA @2       46      49      87     42     351     148      50
3d5tA @2       78      65      46     51      46     321      50
1smkA @2       16      19      18     19      16      16     313
```

1smkA is very different from the other templates, but it is also the one with the lowest sequence identity to the query sequence (34%)



Excerpt from log file

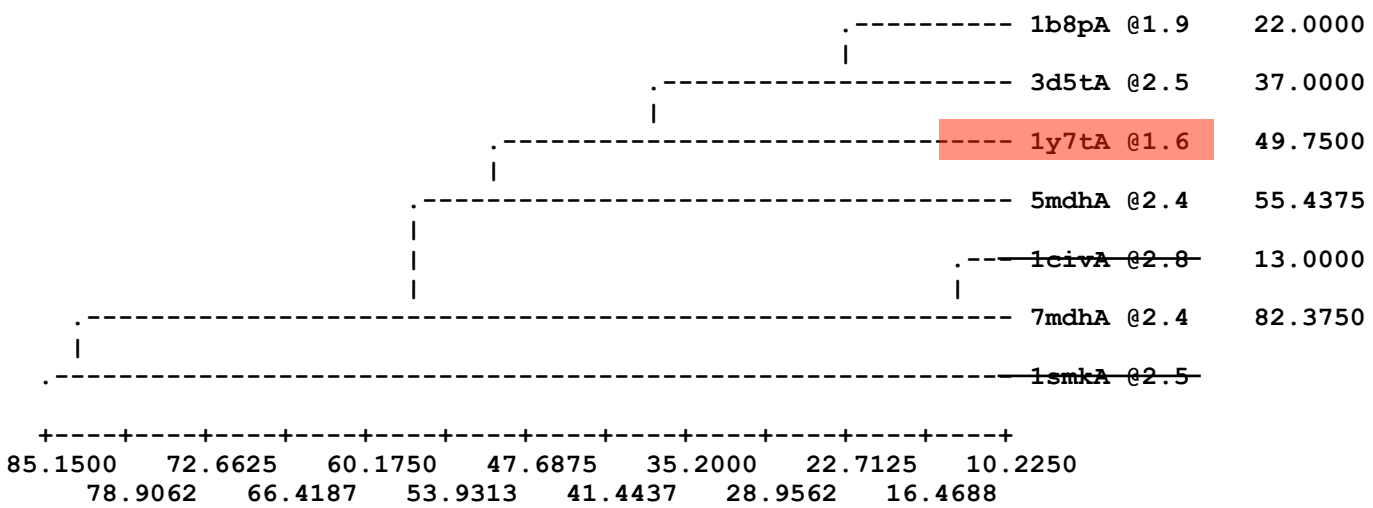
```
Sequence identity comparison (ID_TABLE):
   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

         1b8pA @11y7tA @11civA @25mdhA @27mdhA @23d5tA @21smkA @2
1b8pA @1      327     201     146     152     152     249      50
1y7tA @1       61     327     158     170     160     210      58
1civA @2       45      48     374     140     304     148      55
5mdhA @2       46      52      42     333     140     164      58
7mdhA @2       46      49      87      42     351     148      50
3d5tA @2       78      65      46      51      46     321      50
1smkA @2       16      19      18      19      16      16     313
```

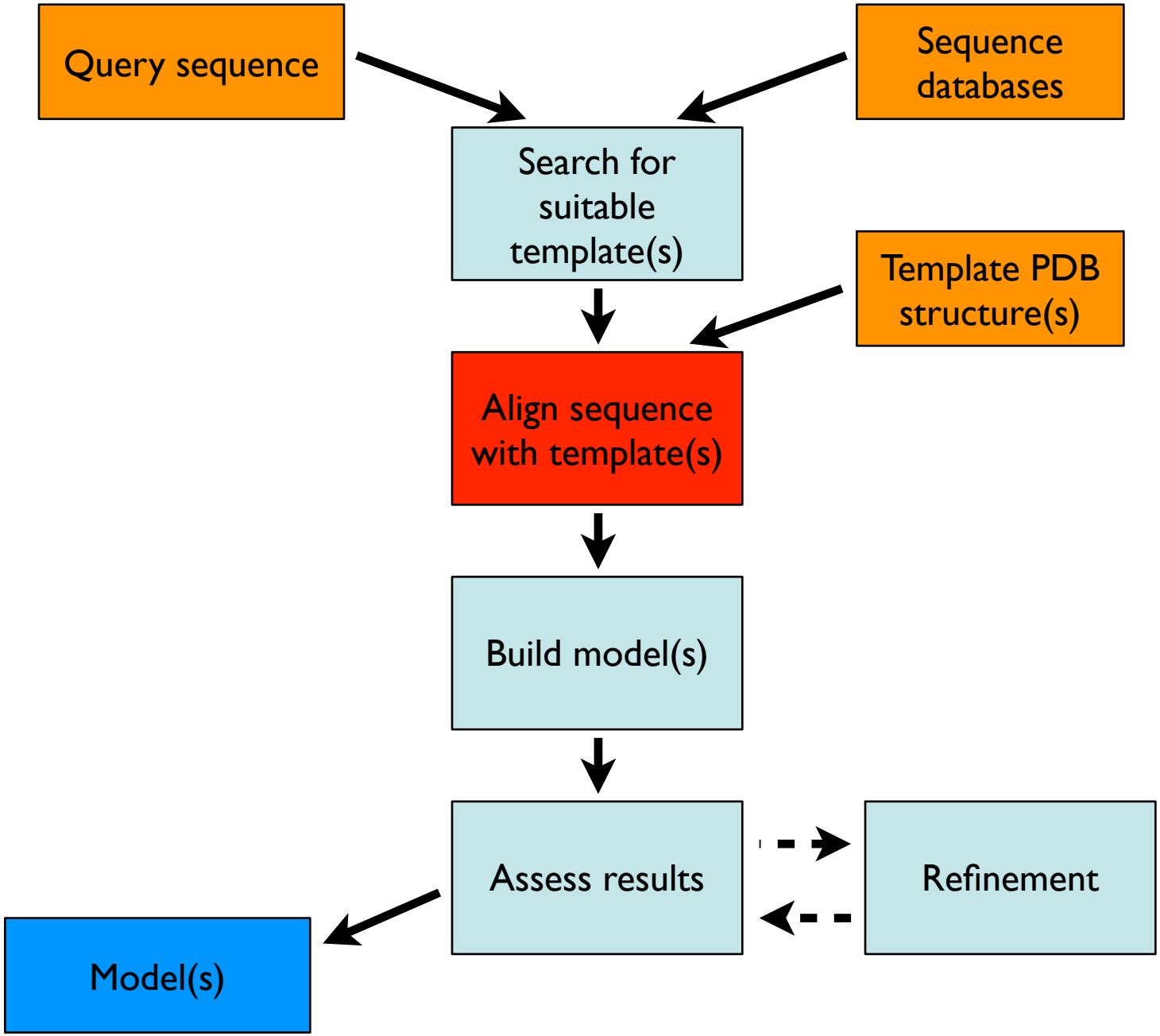


Excerpt from log file

```
Sequence identity comparison (ID_TABLE):
   Diagonal       ... number of residues;
   Upper triangle ... number of identical residues;
   Lower triangle ... % sequence identity, id/min(length).

         1b8pA @11y7tA @11civA @25mdhA @27mdhA @23d5tA @21smkA @2
1b8pA @1      327     201     146     152     152     249      50
1y7tA @1       61     327     158     170     160     210      58
1civA @2       45      48     374     140     304     148      55
5mdhA @2       46      52      42     333     140     164      58
7mdhA @2       46      49      87
3d5tA @2       78      65      46
1smkA @2       16      19      18
```

Of the remaining structures, 1y7tA has the highest identity to the query sequence (45%) and also the best crystallographic resolution (1.6Å)




Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

- align2d.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

aln = alignment(env)

# Read in the 1y7t template structure and add to alignment
mdl = model(env, file='1y7t', model_segment=('FIRST:A','LAST:A'))
aln.append_model(mdl, align_codes='1y7tA', atom_files='1y7t')

# Add in the TvLDH sequence
aln.append(file='TvLDH.ali', align_codes='TvLDH')

# Sequence/structure alignment
aln.align2d(max_gap_length=40)

# Write out resulting alignment in both PIR and PAP formats
aln.write(file='TvLDH-1y7tA.ali', alignment_format='PIR')
aln.write(file='TvLDH-1y7tA.pap', alignment_format='PAP')
```

- align2d.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

aln = alignment(env)

# Read in the 1y7t template structure and add to alignment
mdl = model(env, file='1y7t', model_segment=('FIRST:A','LAST:A'))
aln.append_model(mdl, align_codes=

# Add in the TvLDH sequence
aln.append(file='TvLDH.ali', align_codes='TvLDH')

# Sequence/structure alignment
aln.align2d(max_gap_length=40)

# Write out resulting alignment in both PIR and PAP formats
aln.write(file='TvLDH-1y7tA.ali', alignment_format='PIR')
aln.write(file='TvLDH-1y7tA.pap', alignment_format='PAP')
```

Remember that align2d uses structural information: it prefers not to add gaps in the template within helices, beta strands

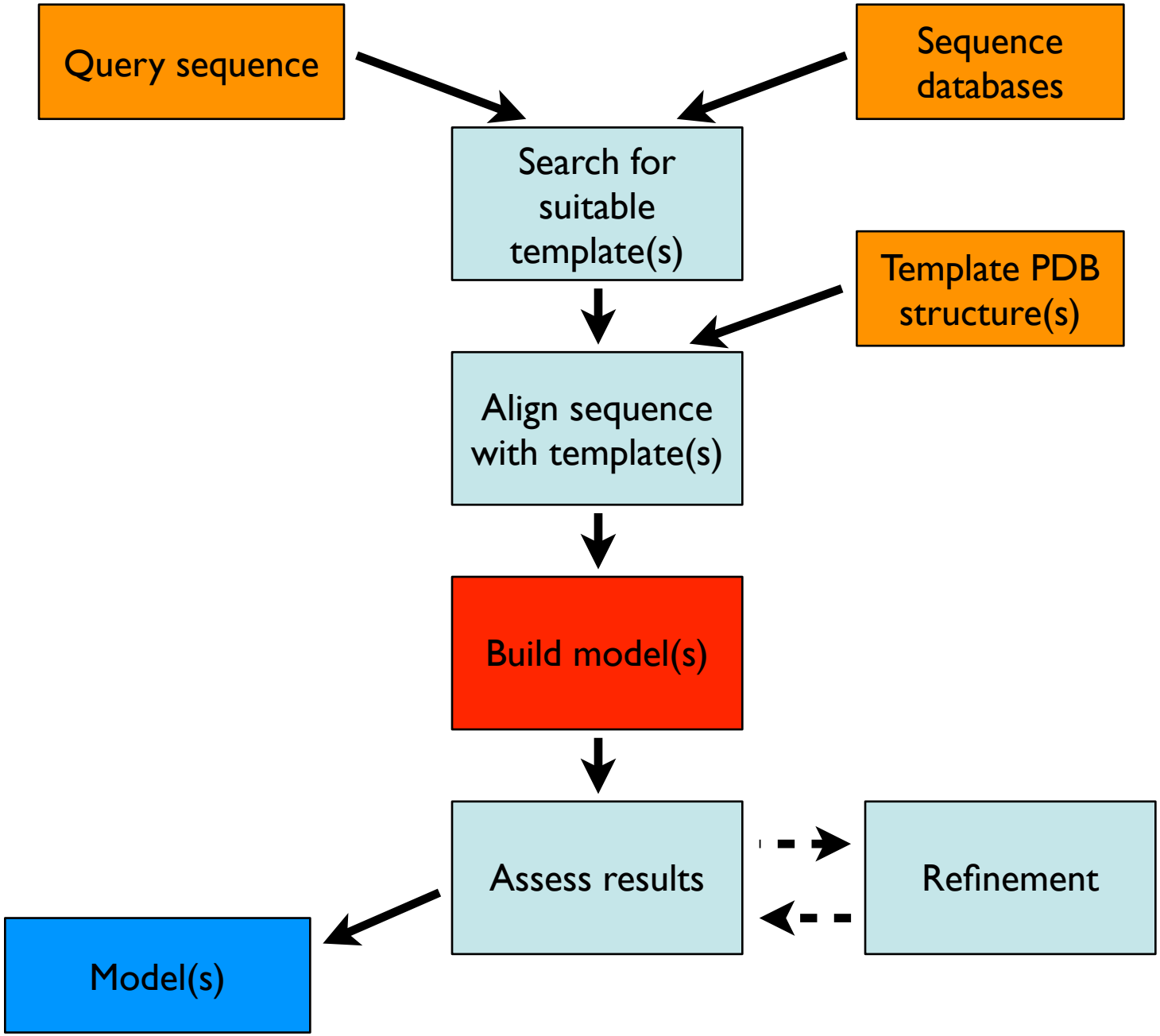
- align2d.py:

```
from modeller import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

aln = alignment(env)

# Read in the 1y7t template structure and add to alignment
mdl = model(env, file='1y7t', model_segment=('FIRST:A','LAST:A'))
aln.append_model(mdl, align_codes='1y7tA', atom_files='1y7t')

# Add in the TvLDH sequence
aln.append(file='TvLDH.ali', align_codes='TvLDH')

# Sequence/structure alignment
aln.align2d(max_gap_length=40)

# Write out resulting alignment in both PIR and PAP formats
aln.write(file='TvLDH-1y7tA.ali', alignment_format='PIR')
aln.write(file='TvLDH-1y7tA.pap', alignment_format='PAP')
```

Output .ali file

```
>P1;1y7tA
structureX:1y7t:   0 :A:+327 :A:MALATE DEHYDROGENASE:THERMUS THERMOPHILUS: 1.65: 0.20
MKAPVRVAVTGAAGQIGYSLLFRIAAGEMLGKDQPVILQLLEIPQAMKALEGVVMELEDCAFPLLAGLEATDDPK
VAFKDADYALLVGAAPRKAGMERRDLLQVNGKIFTEQGRALAEVAKKDVKVLVVGNPANTNALIAYKNAPGLNPR
NFTAMTRLDHNRAKAQLAKKTGTGVDRIRRMTVWGNHSSTMFPDLFHAEV--DGR--PALELVDMEWYEKVFIPT
VAQRGAAIIQARGASSAASAANAAIEHIRDWALGTPEGDWVSMAVPS-QG-EYGIPEGIVYSFPVTA-KDGAYRV
VEGLEINEFARKRMEITAQELLDEME-QVKALG-LI*

>P1;TvLDH
sequence::     : :     : :::-1.00:-1.00
MSEAAHVLITGAAGQIGYILSHWIASGELYG-DRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPK
AAFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPE
NFSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKK
IGHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHV
VEGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

- Ready for modeling
- Note the structureX: line; Modeller will be able to read the A chain from the 1y7t PDB file to get corresponding structure when it needs it

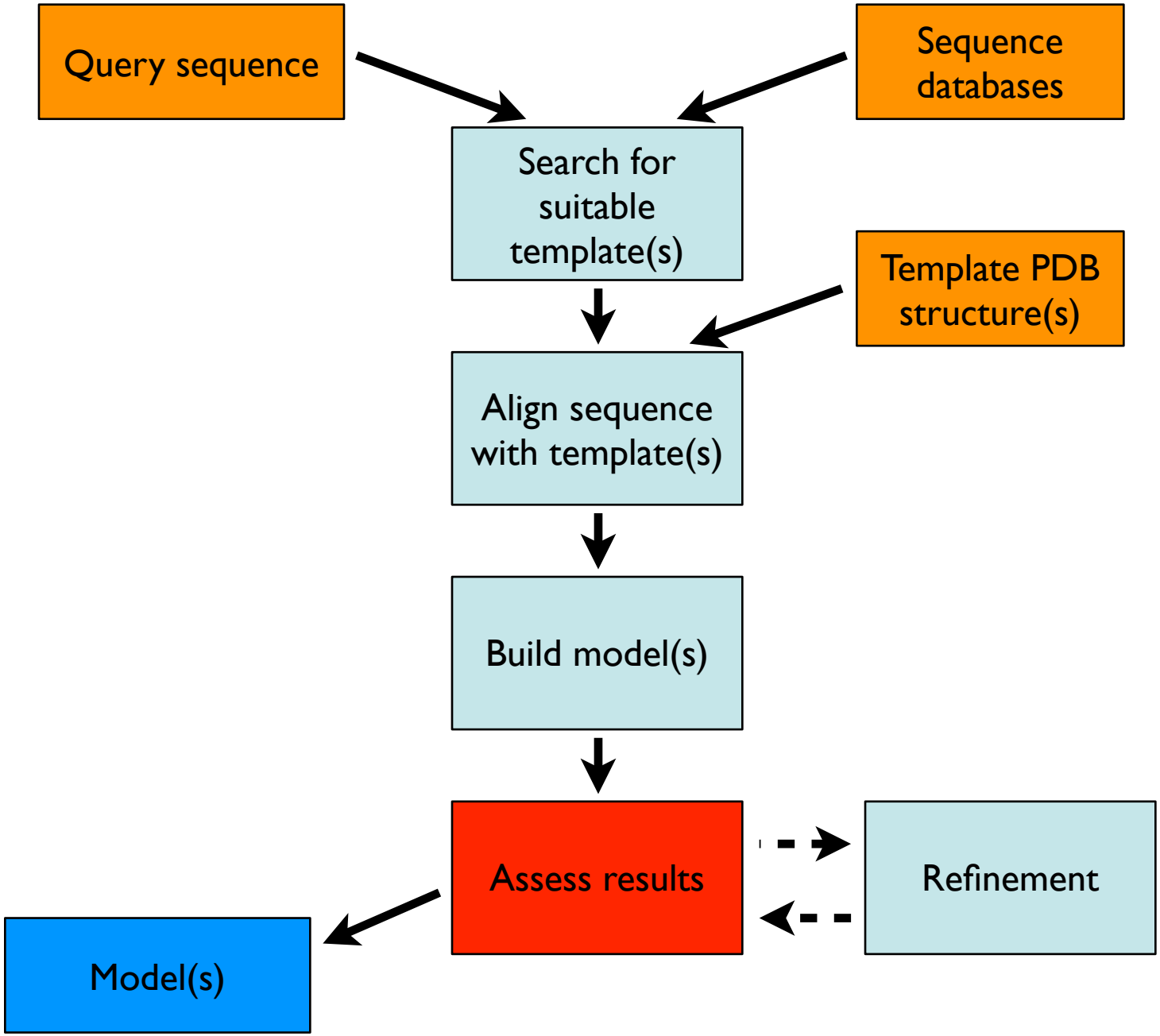

Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

- model-single.py:

```
from modeller import *
from modeller.automodel import *

env = environ()
env.io.atom_files_directory = ['.', 'atom_files']

a = automodel(env, alnfile='TvLDH-1y7tA.ali',
              knowns='1y7tA', sequence='TvLDH',
              assess_methods=assess.normalized_dope)

a.starting_model = 1
a.ending_model = 5

a.make()
```

- The automodel Python class provides an automated way to build comparative models given an alignment

Output: log file and PDB models


Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

Assess

- How do we know if the model is a good one?
- Check log file for restraint violations and Modeller score (molpdf)
  - Not reliable since the scoring function is not perfect!
- Use another assessment score on the final model
  - Fold assignment: GA341
  - Statistical potential: DOPE
  - Other programs (e.g. Prosa)
- Fit the model to some other experimental data not used in the modeling procedure

DOPE assessment

- Excerpt from modeling log file:

```
>> Summary of successfully produced models:
Filename                          molpdf Normalized DOPE score
----------------------------------------------------------------
TvLDH.B99990001.pdb            1675.31470       -1.08121
TvLDH.B99990002.pdb            1974.06335       -1.00076
TvLDH.B99990003.pdb            1638.27026       -0.96732
TvLDH.B99990004.pdb            1753.61841       -0.94745
TvLDH.B99990005.pdb            1736.45630       -0.99304
```

- Normalized DOPE is a z score, and the most reliable of Modeller's own assessment methods
- Scores of -1 or less generally indicate native-like structures
- Thus, all of the generated models are probably OK; the first is probably the best one

Fit to other experimental data

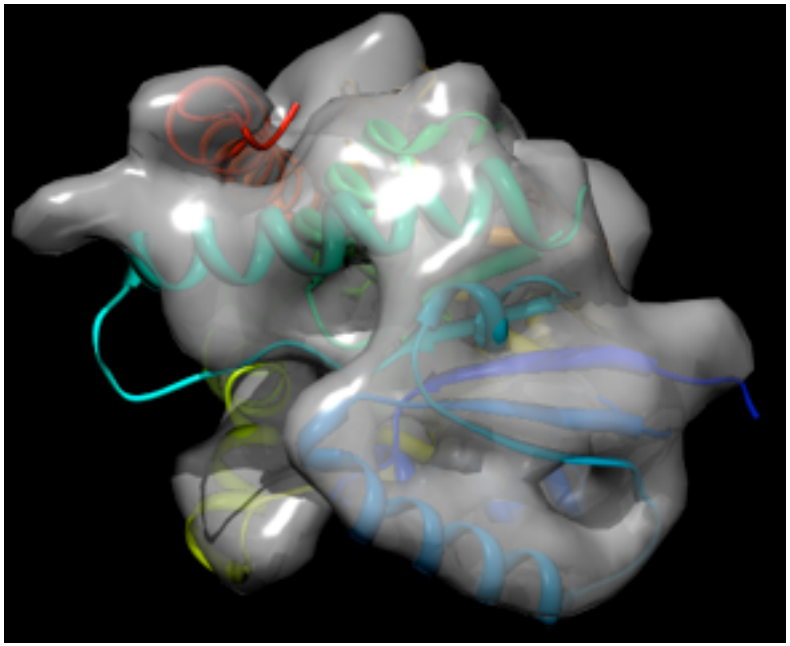
- Often even though there is no X-ray crystal structure, we have some other experimental information
- For example, we have a lower-resolution cryo-EM map of the protein
- Modeller can fit a protein into a cryo-EM map and assess the cross correlation

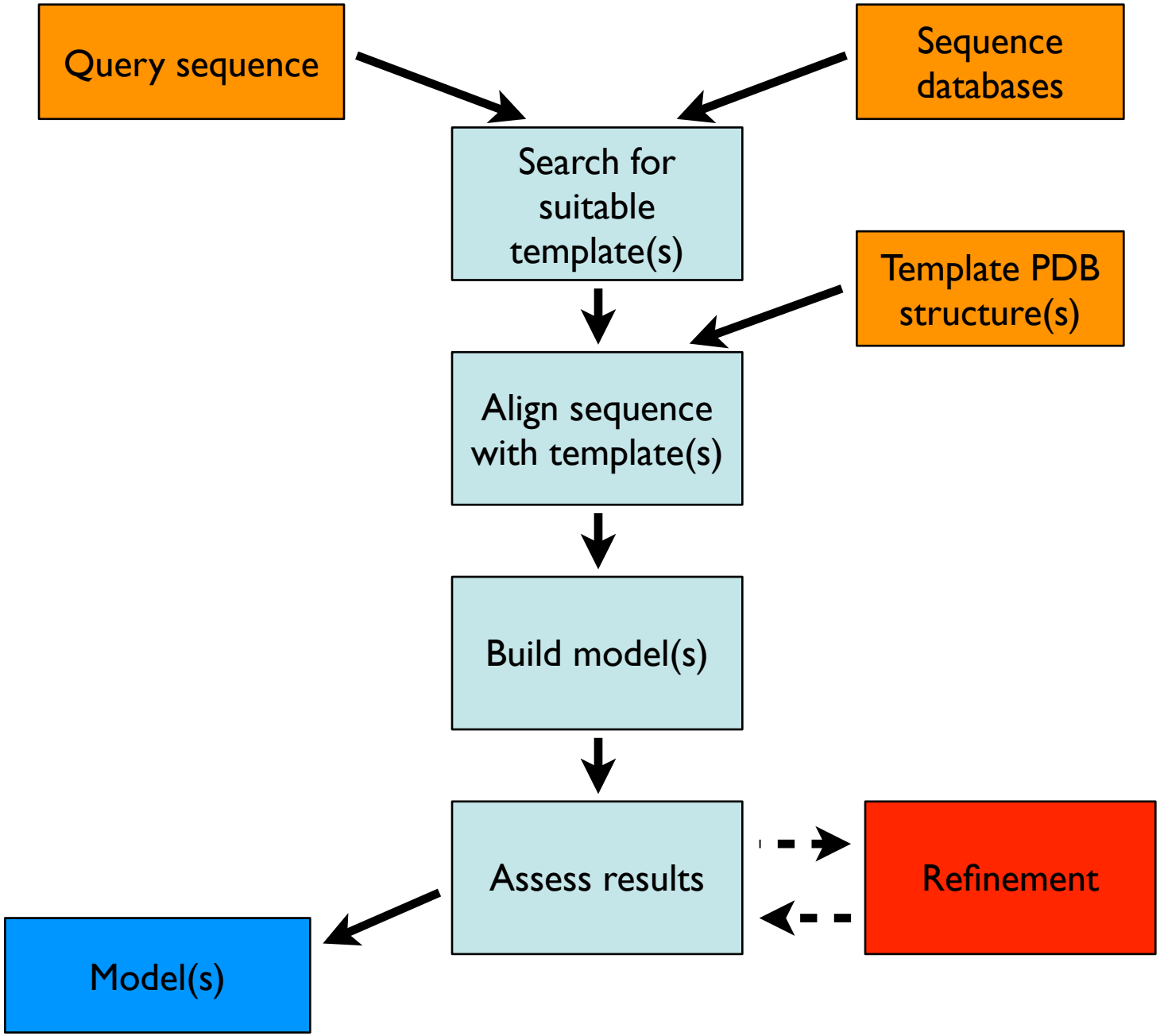
- modem_mc.py:

```
from modeller import *

log.verbose()
env = environ()

# Read in cryo-EM density map
den = density(env, file='TvLDH.10A.mrc', em_density_format='MRC',
              resolution=10.0, density_type='GAUSS',
              px=-26.742, py=-9.5205, pz=-10.375)

# Fit the PDB file into the map by MC simulated annealing
den.grid_search(em_density_format='MRC', num_structures=1,
                em_pdb_name='TvLDH.B99990001.pdb', chains_num=[1],
                start_type='CENTER', number_of_steps=20,
                angular_step_size=30., temperature=100.,
                best_docked_models=1, translate_type='RANDOM',
                em_fit_output_file='modem.log')
```

EM fit results

- Output:
  - TvLD_1_1.pdb, the best fit of the protein into the map
  - Log file, containing the cross-correlation score (good fit, 0.9387)




Query sequence
Sequence databases
Search for suitable template(s)
Template PDB structure(s)
Align sequence with template(s)
Build model(s)
Assess results
Refinement
Model(s)

Refinement: loop modeling

- Loop modeling refines small sections of the protein using a statistical potential (not template information)
- From the fit into the EM map, we could conclude that residues 93-100 need to be refined, since they don't fit well
- Multiple models should be generated to sample conformational space (longer loops, more models)

- loop.py:

```
from modeller import *
from modeller.automodel import *

env = environ()

# Build 5 loop models from TvLD_1_1.pdb, and assess each one with DOPE
a = loopmodel(env, inimodel='TvLD_1_1.pdb',
              sequence='TvLDH', loop_assess_methods=assess.DOPE)
a.loop.starting_model = 1
a.loop.ending_model = 5
a.make()
```

- loop.py:

```
from modeller import *
from modeller.automodel import *




env = environ()

# Build 5 loop models from TvLD_1_1.pdb, and assess each one with DOPE
a = loopmodel(env, inimodel='TvLD_1_1.pdb',
              sequence='TvLDH', loop_assess_methods=assess.DOPE)
a.loop.starting_model = 1
a.loop.ending_model = 5
a.make()
```

- loop.py:

```
from modeller import *
from modeller.automodel import *

# Override the regular loopmodel class to select our own loops
class MyLoop(loopmodel):
    # This routine picks the residues to be refined by loop modeling
    def select_loop_atoms(self):
        # One loop from residues 93 to 100 inclusive
        return selection(self.residue_range('93:', '100:'))

env = environ()

# Build 5 loop models from TvLD_1_1.pdb, and assess each one with DOPE
a = loopmodel(env, inimodel='TvLD_1_1.pdb',
              sequence='TvLDH', loop_assess_methods=assess.DOPE)
a.loop.starting_model = 1
a.loop.ending_model = 5
a.make()
```

- loop.py:

```
from modeller import *
from modeller.automodel import *

# Override the regular loopmodel class to select our own loops
class MyLoop(loopmodel):
    # This routine picks the residues to be refined by loop modeling
    def select_loop_atoms(self):
        # One loop from residues 93 to 100 inclusive
        return selection(self.residue_range('93:', '100:'))

env = environ()

# Build 5 loop models from TvLD_1_1.pdb, and assess each one with DOPE
a =    MyLoop(env, inimodel='TvLD_1_1.pdb',
              sequence='TvLDH', loop_assess_methods=assess.DOPE)
a.loop.starting_model = 1
a.loop.ending_model = 5
a.make()
```

- loop.py:

```
from modeller import *
from modeller.automodel import *

# Override the regular loopmodel class to select our own loops
class MyLoop(loopmodel):
    # This routine picks the residues to be refined by loop modeling
    def select_loop_atoms(self):
        # One loop from residues 93 to 100 inclusive
        return selection(self.residue_range('93:', '100:'))

env = environ()

# Build 5 loop models from TvLD_1_1.pdb, and assess each one with DOPE
a = MyLoop(env, inimodel='TvLD_1_1.pdb',
           sequence='TvLDH', loop_assess_methods=assess.DOPE)
a.loop.starting_model = 1
a.loop.ending_model = 5
a.make()
```

Advanced topics

- Modeller can also
  - Perform more sensitive searches for templates (sequence-profile, profile-profile, similar to PSI-BLAST)
  - Incorporate ligands, RNA/DNA into built models
  - Build structures of multi-chain proteins (homo or hetero)
  - Add extra restraints to the modeling process (such as known distances, e.g. from FRET)
  - Use multiple templates to build a model
- Note: you don't have to use Modeller for template search, alignment, assessment or refinement
  - If you know your template (e.g. from BLAST) just format the alignment for Modeller and skip straight to the model-building step

Web services and databases

- ModBase
  - contains pre-built structures for many existing sequences (potentially, about half of all single-chain sequences)
- ModWeb
  - automates all steps of modeling; builds single-chain models given a sequence (no ligands, multi-chain, or refinement)
- ModLoop
  - refines given regions in an existing PDB file

MODBASE

Database of comparative protein structure models

http://salilab.org/modbase

- Current contents: 10,741,814 models from 2,477,564 unique sequences

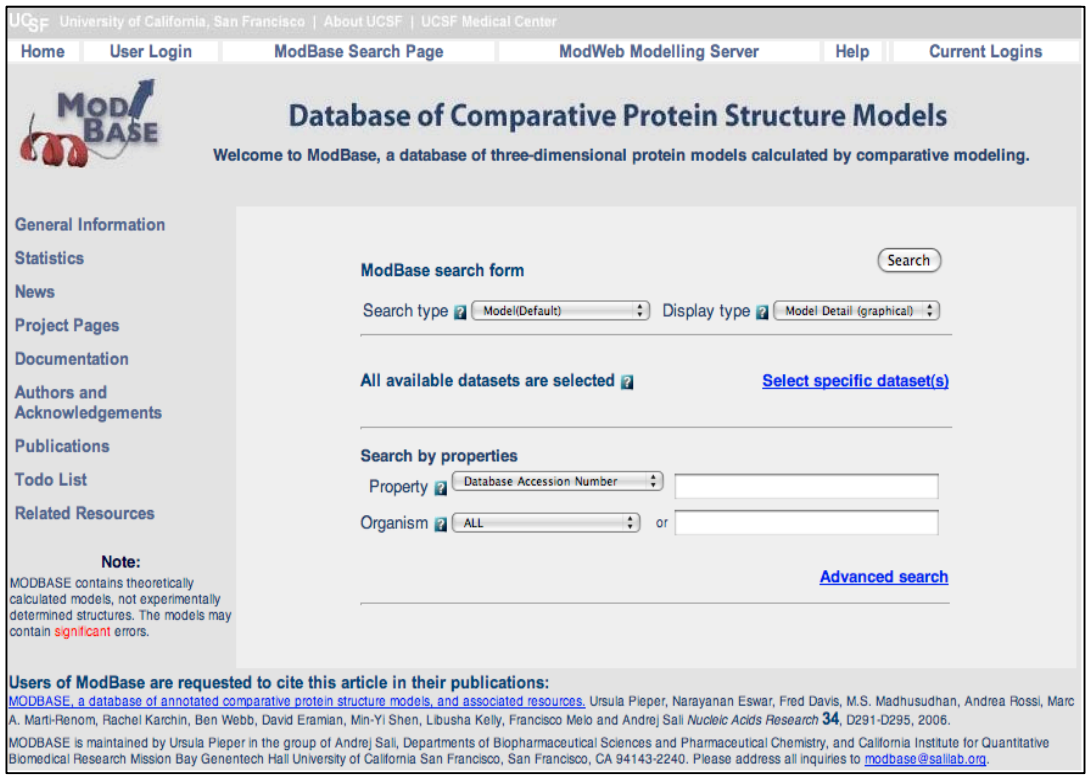
MODBASE

Database of comparative protein structure models

http://salilab.org/modbase

- Current contents: 10,741,814 models from 2,477,564 unique sequences

Search form



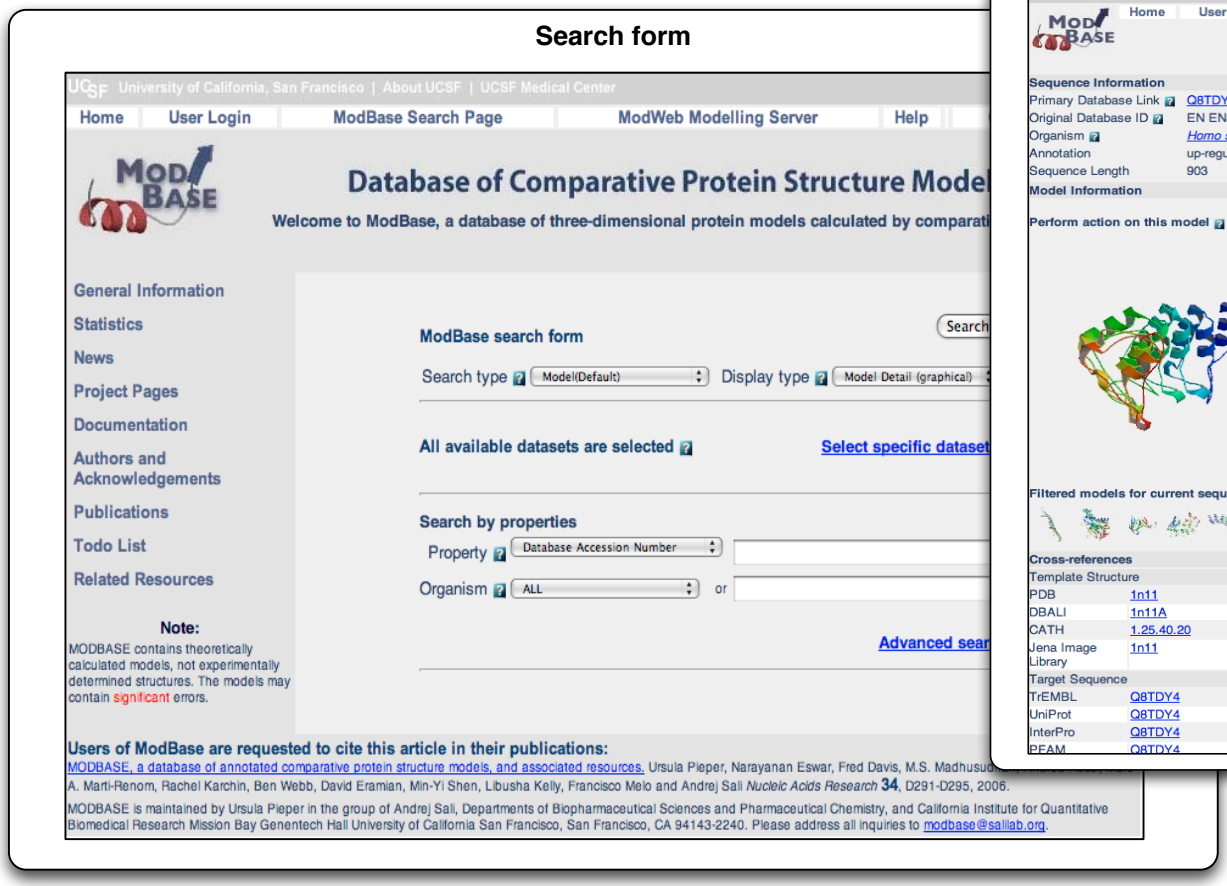
MODBASE

Database of comparative protein structure models

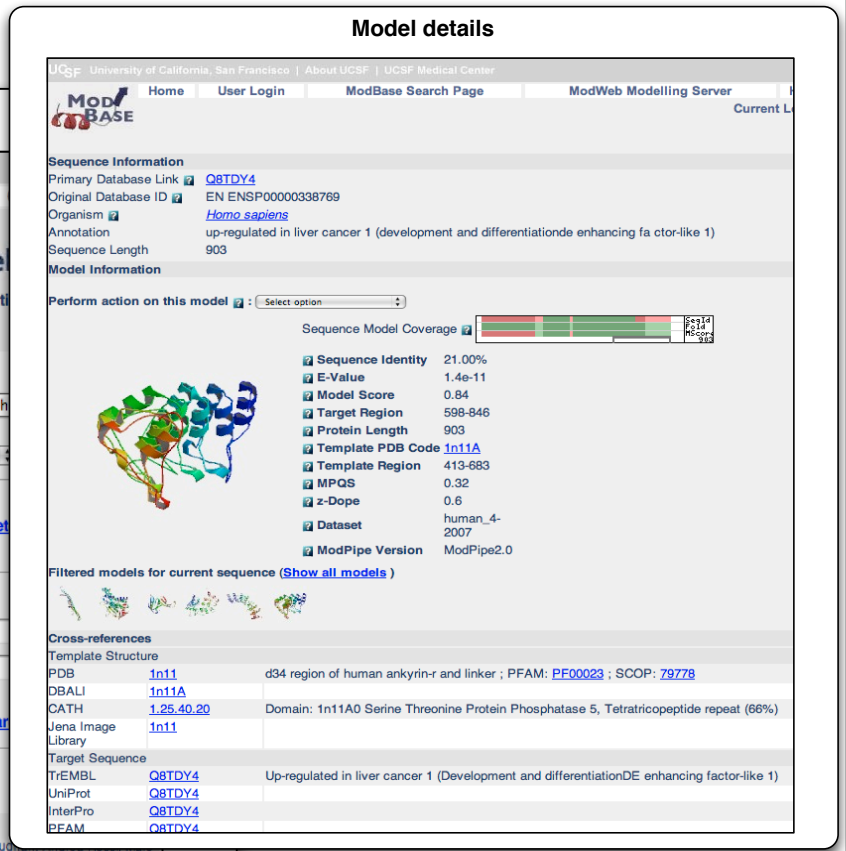
http://salilab.org/modbase

- Current contents: 10,741,814 models from 2,477,564 unique sequences

Search form



Model details



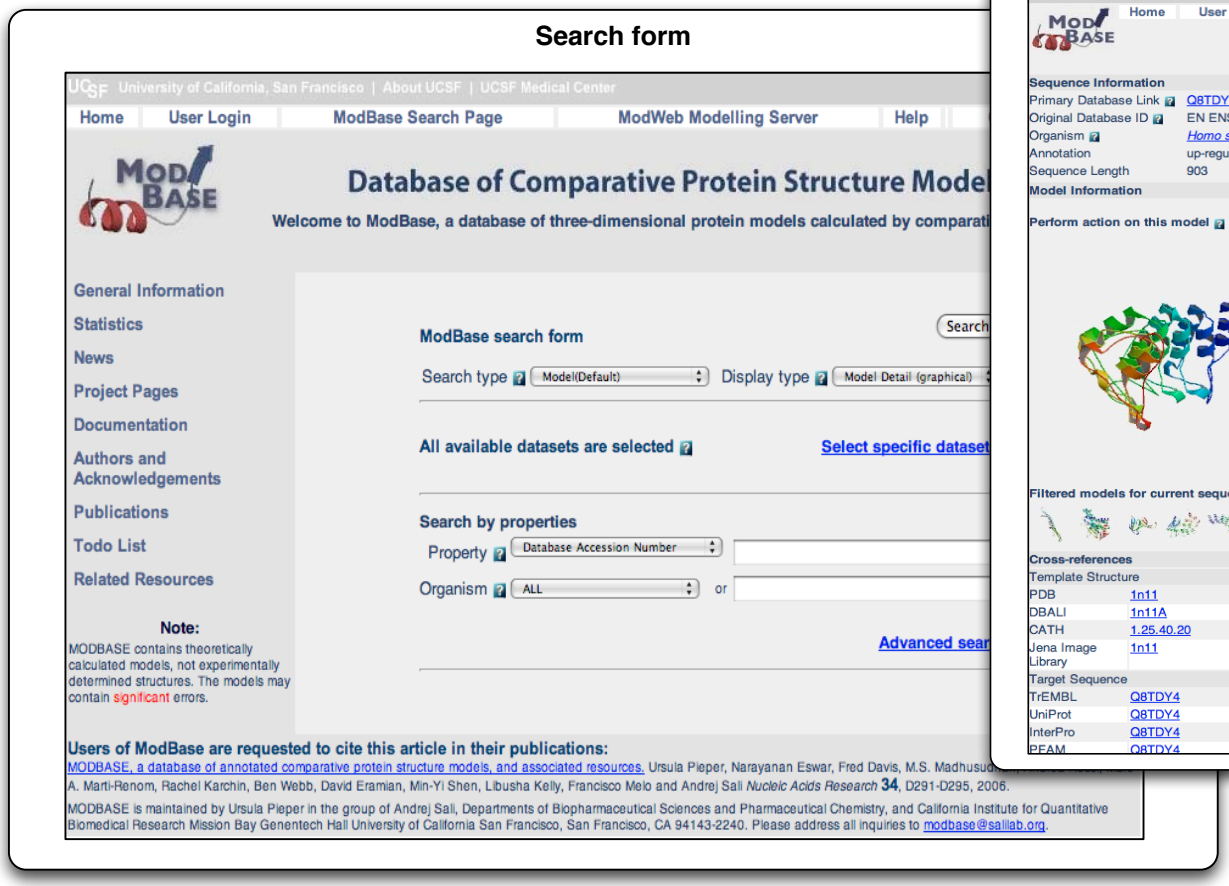
MODBASE

Database of comparative protein structure models

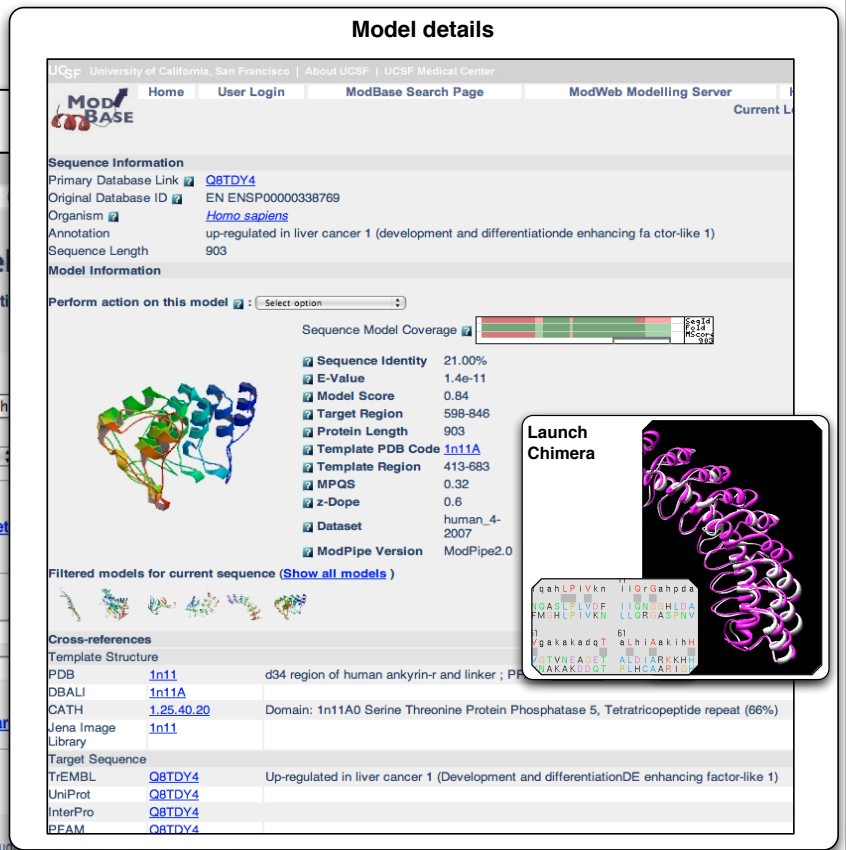
http://salilab.org/modbase

- Current contents: 10,741,814 models from 2,477,564 unique sequences

Search form



Model details

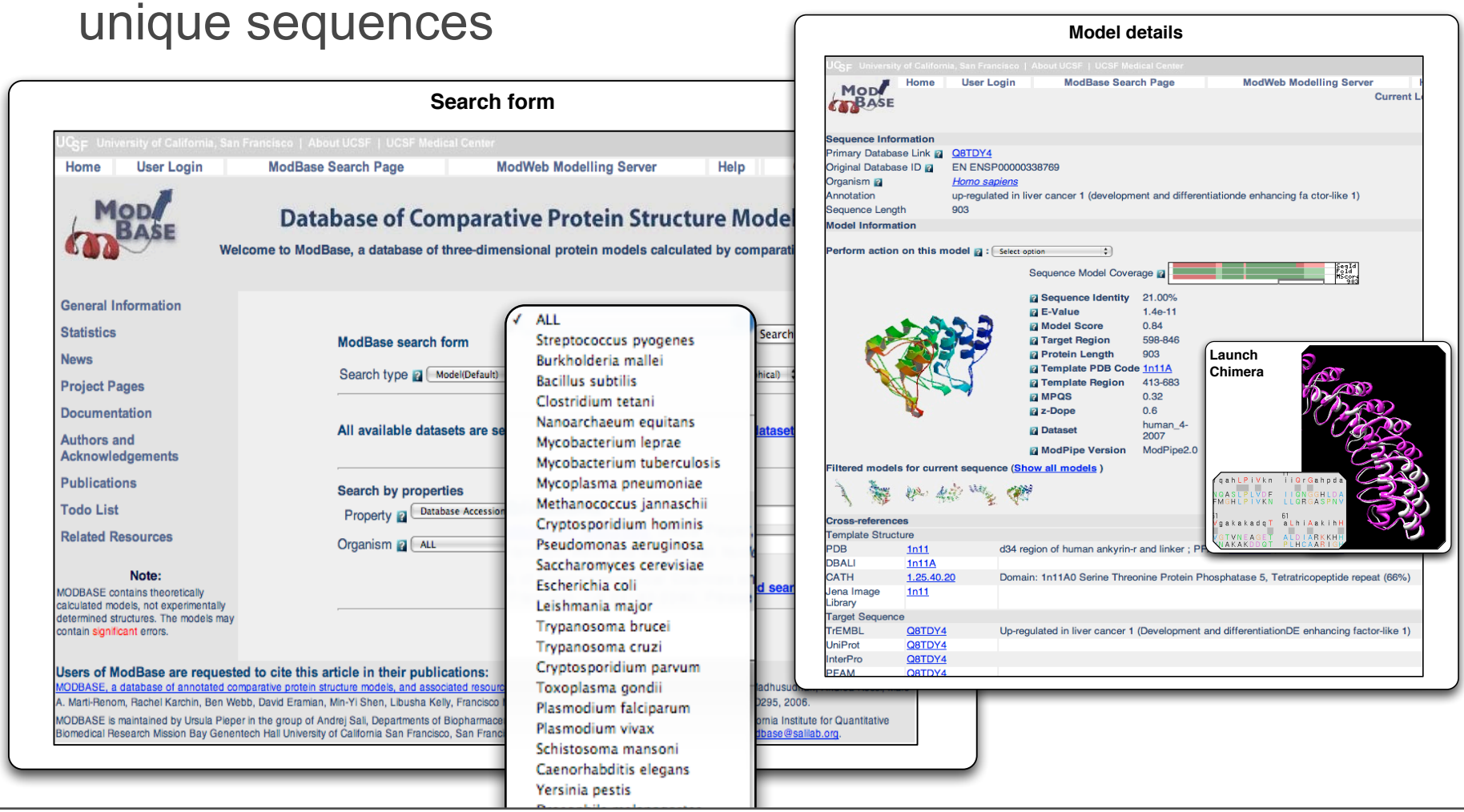


MODBASE

Database of comparative protein structure models

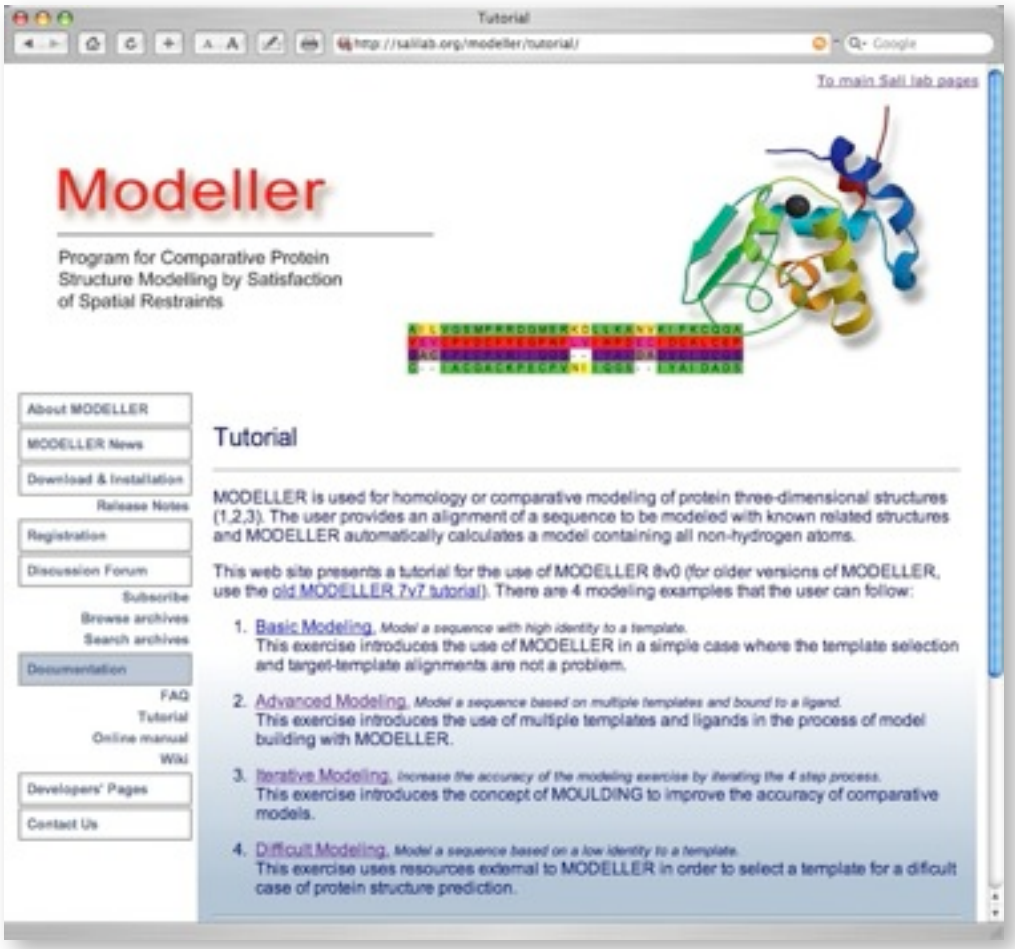
http://salilab.org/modbase

- Current contents: 10,741,814 models from 2,477,564 unique sequences

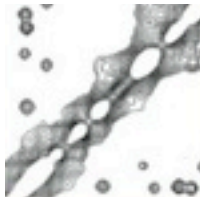


For further examples…

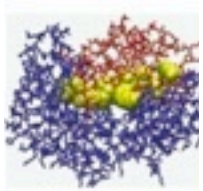
- http://salilab.org/modeller/tutorial/



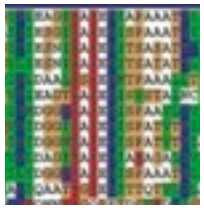
Integrative modeling

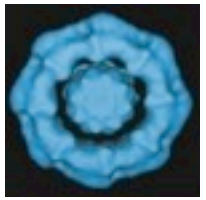

NMR

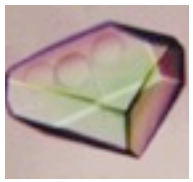
structure prediction

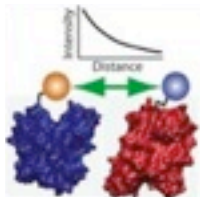

site-directed mutagenesis


affinity purification


bioinformatics

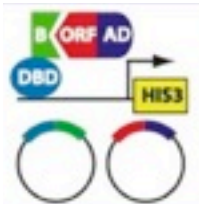

cryo-EM


X-ray structures



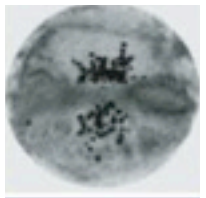
FRET


crosslinking



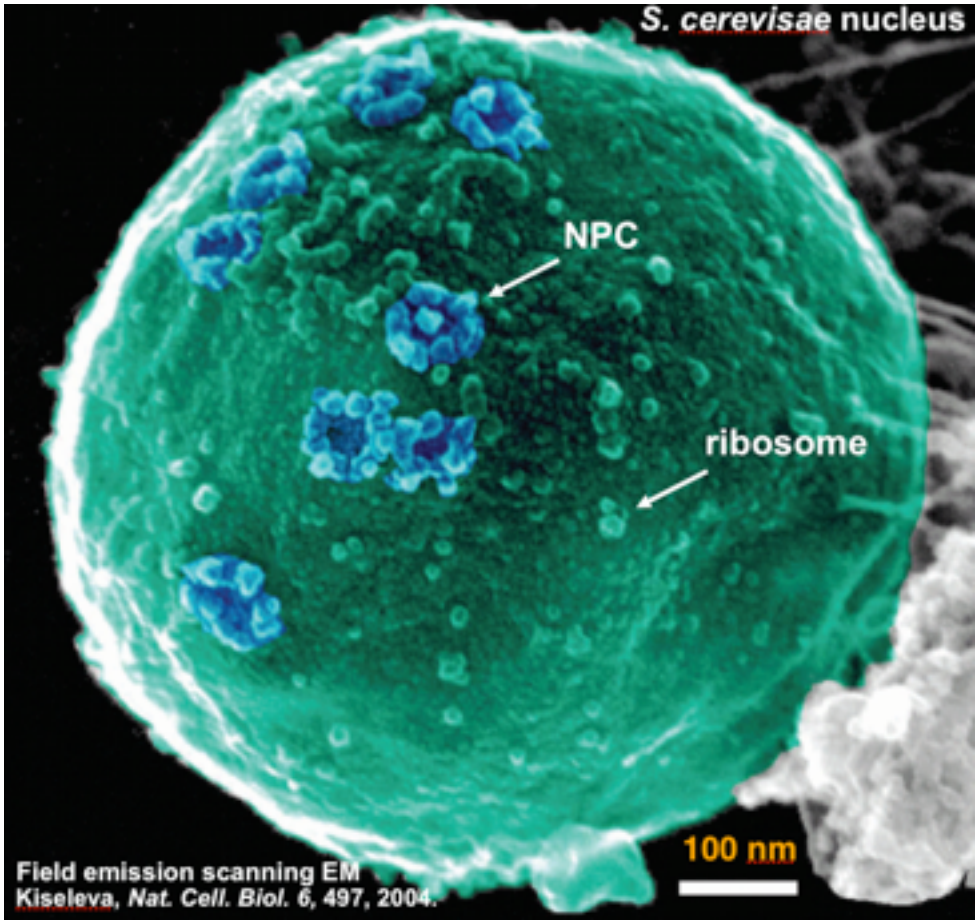
yeast two-hybrid


computational docking


immuno-EM

- Modeller uses only a subset of these data sources
- Limited to building structures of proteins at atomic resolution
- Our IMP package can use all data sources to build models of protein assemblies at a range of resolutions

Nuclear Pore Complex (NPC)



Consists of broadly conserved **nucleoporins** (nups).

50 MDa complex: **~480** proteins of **30** different types.

Mediates all known nuclear **transport**, *via* cognate transport factors.

Mike Rout
Svetlana Dokudovskaya, Liesbeth Veenhoff
Orit Karni-Schmidt, Julia Kipper, Tari Suprapto, Julia Kipper

Brian Chait
Wenzhu Zhang, Rosemary Williams

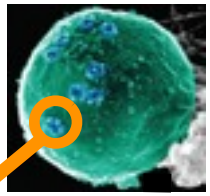
Rockefeller University

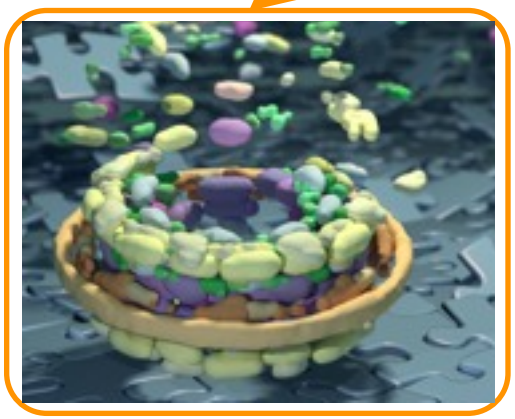
Alber *et al*. Nature 450, 683-694, 2007
Alber *et al*. Nature 450, 695-701, 2007
Devos *et al*. PNAS 14, 2172-2177, 2006
Devos *et al*. PLoS Biology 12, 1-9, 2004

Andrej Sali
Frank Alber, Damien Devos
Narayanan Eswar, Marc Marti-Renom

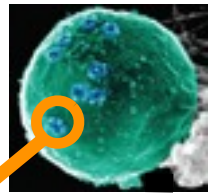
UCSF

Configuration of proteins in NPC?

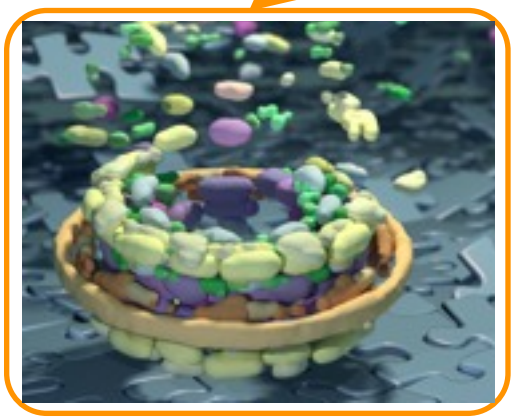




Configuration of proteins in NPC?

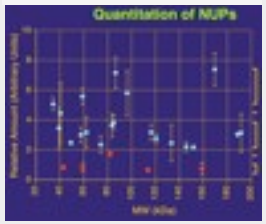


Use all information



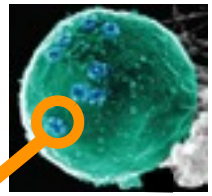
Quantitative Immunoblotting

30 relative abundances

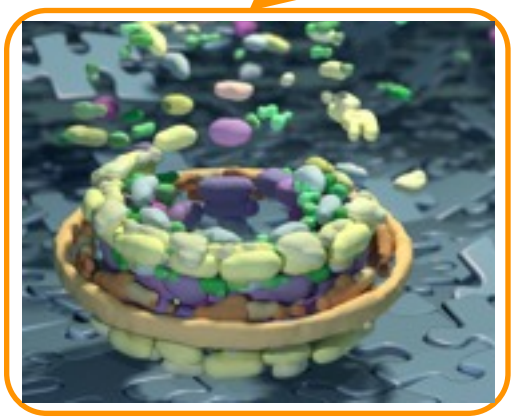


Configuration of proteins in NPC?

Use all information

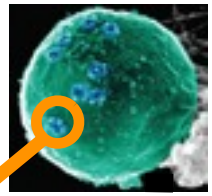


Protein Stoichiometry



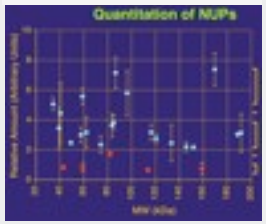
Configuration of proteins in NPC?

Use all information

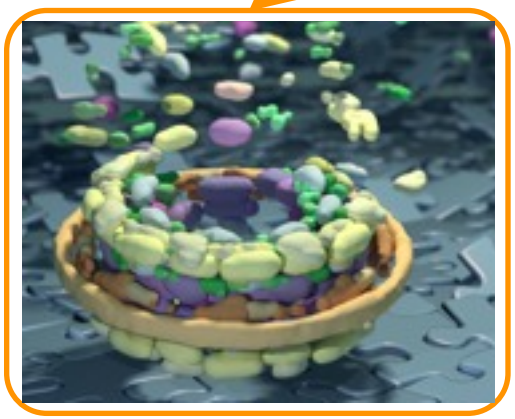


Quantitative Immunoblotting

30 relative abundances



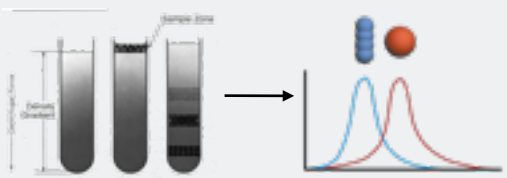
Protein Stoichiometry



Protein Shape

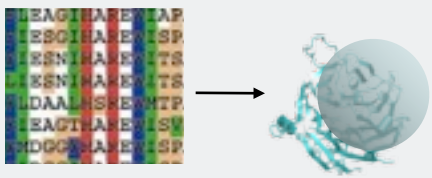
Ultracentrifugation

30 S-values 1 S-value



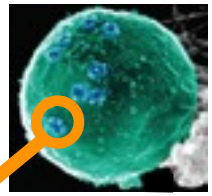
Bioinformatics and Membrane Fractionation

30 protein sequences



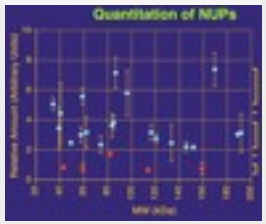
Configuration of proteins in NPC?

Use all information

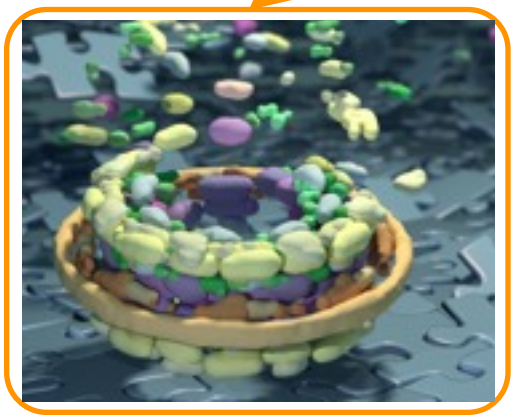


Quantitative Immunoblotting

30 relative abundances



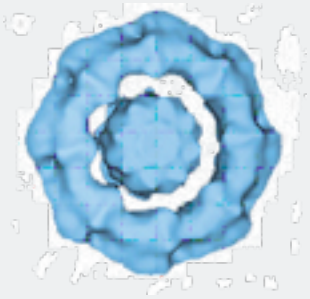
Protein Stoichiometry



Symmetry

Electron Microscopy

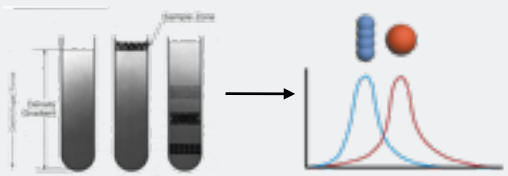
electron microscopy map



Protein Shape

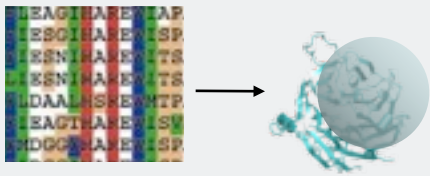
Ultracentrifugation

30 S-values 1 S-value



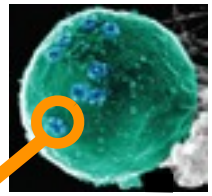
Bioinformatics and Membrane Fractionation

30 protein sequences



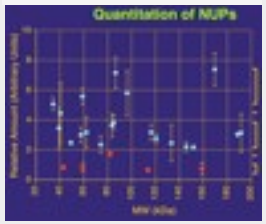
Configuration of proteins in NPC?

Use all information

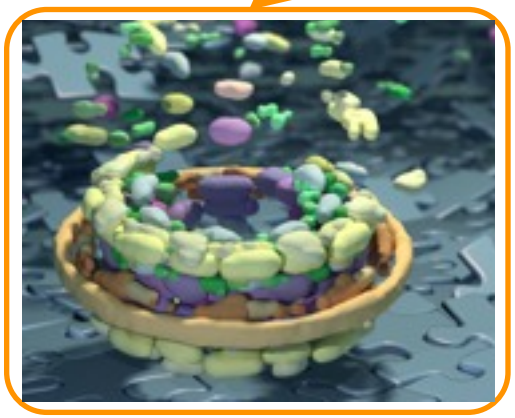


Quantitative Immunoblotting

30 relative abundances



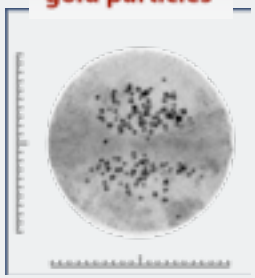
Protein Stoichiometry



Protein Localization

Immuno-Electron Microscopy

10,615 gold particles



Electron Microscopy

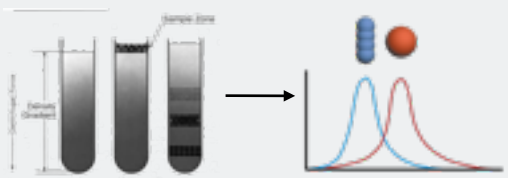
electron microscopy map

Symmetry

Protein Shape

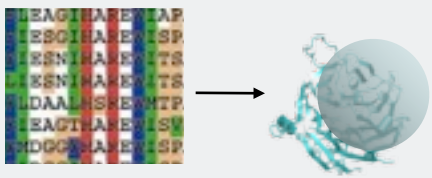
Ultracentrifugation

30 S-values 1 S-value



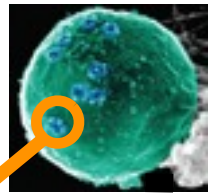
Bioinformatics and Membrane Fractionation

30 protein sequences



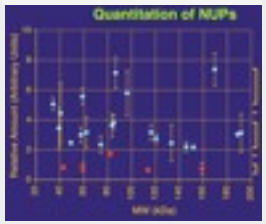
Configuration of proteins in NPC?

Use all information



Quantitative Immunoblotting

30 relative abundances

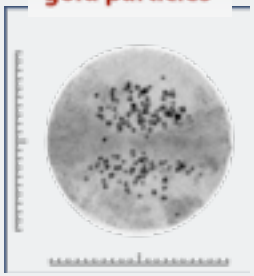


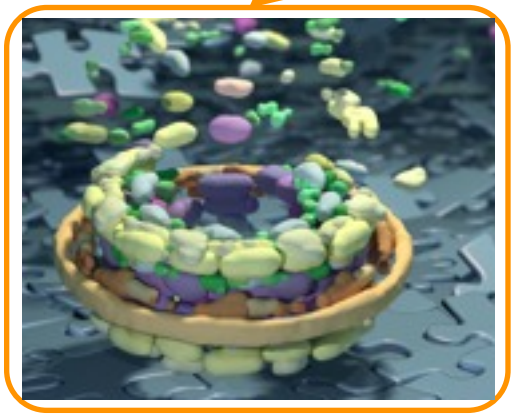
Protein Stoichiometry

Protein Localization

Immuno-Electron Microscopy

10,615 gold particles



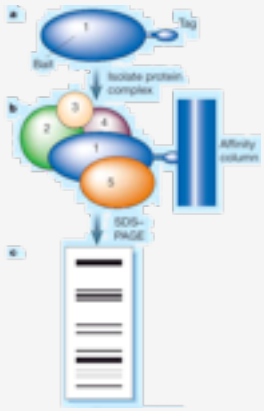


Affinity Purification

Overlay Assay

75 composites

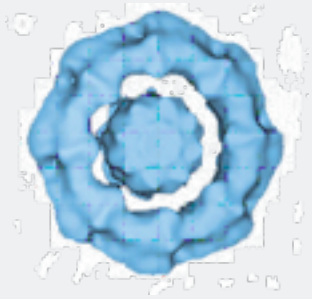
7 contacts



Protein-protein Proximities

Electron Microscopy

electron microscopy map



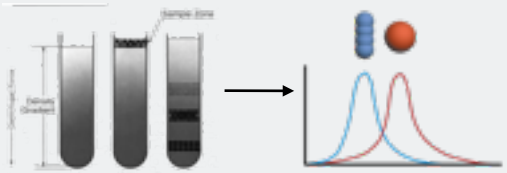
Symmetry

Protein Shape

Ultracentrifugation

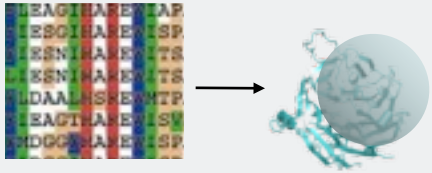
30 S-values

1 S-value

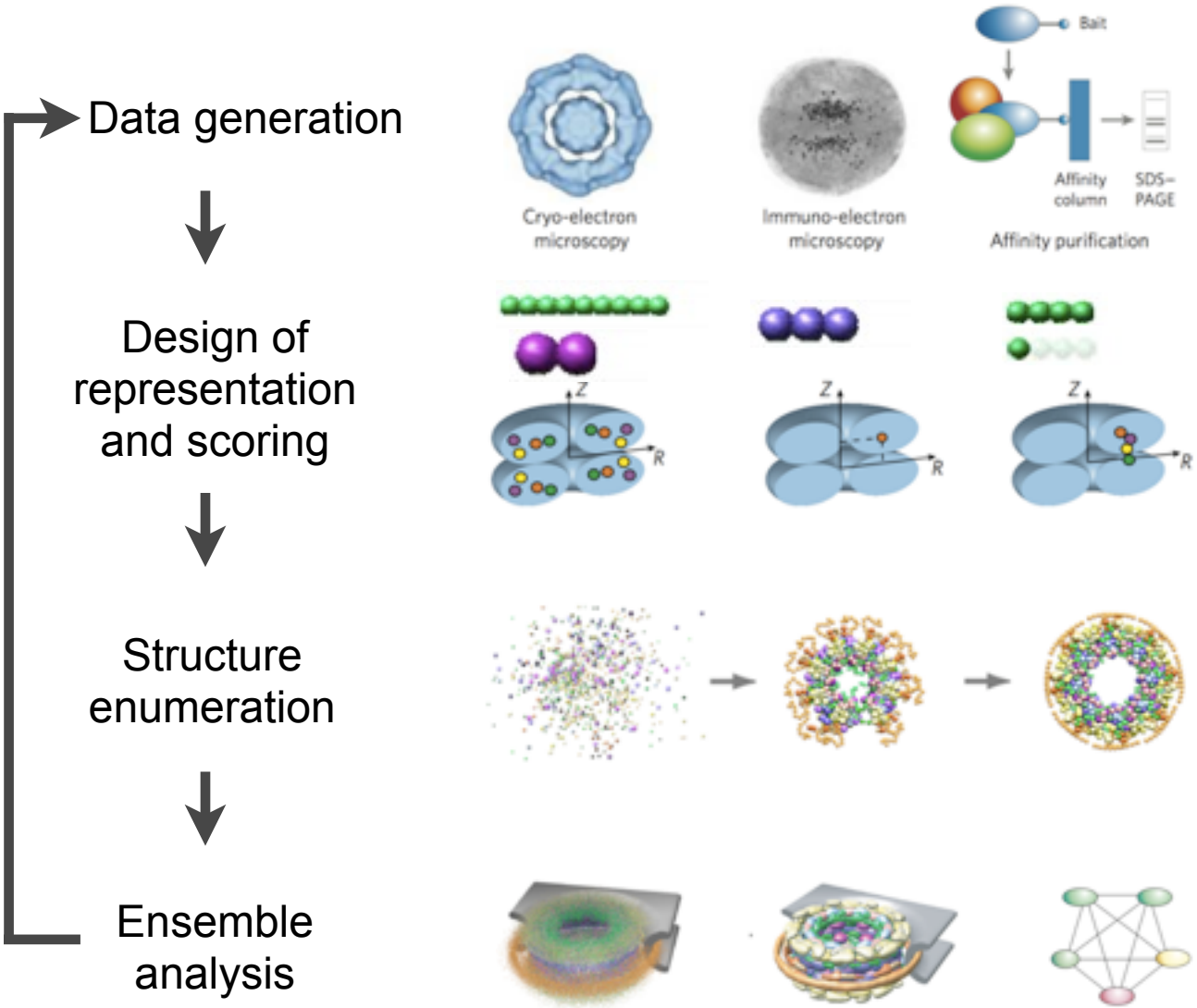


Bioinformatics and Membrane Fractionation

30 protein sequences



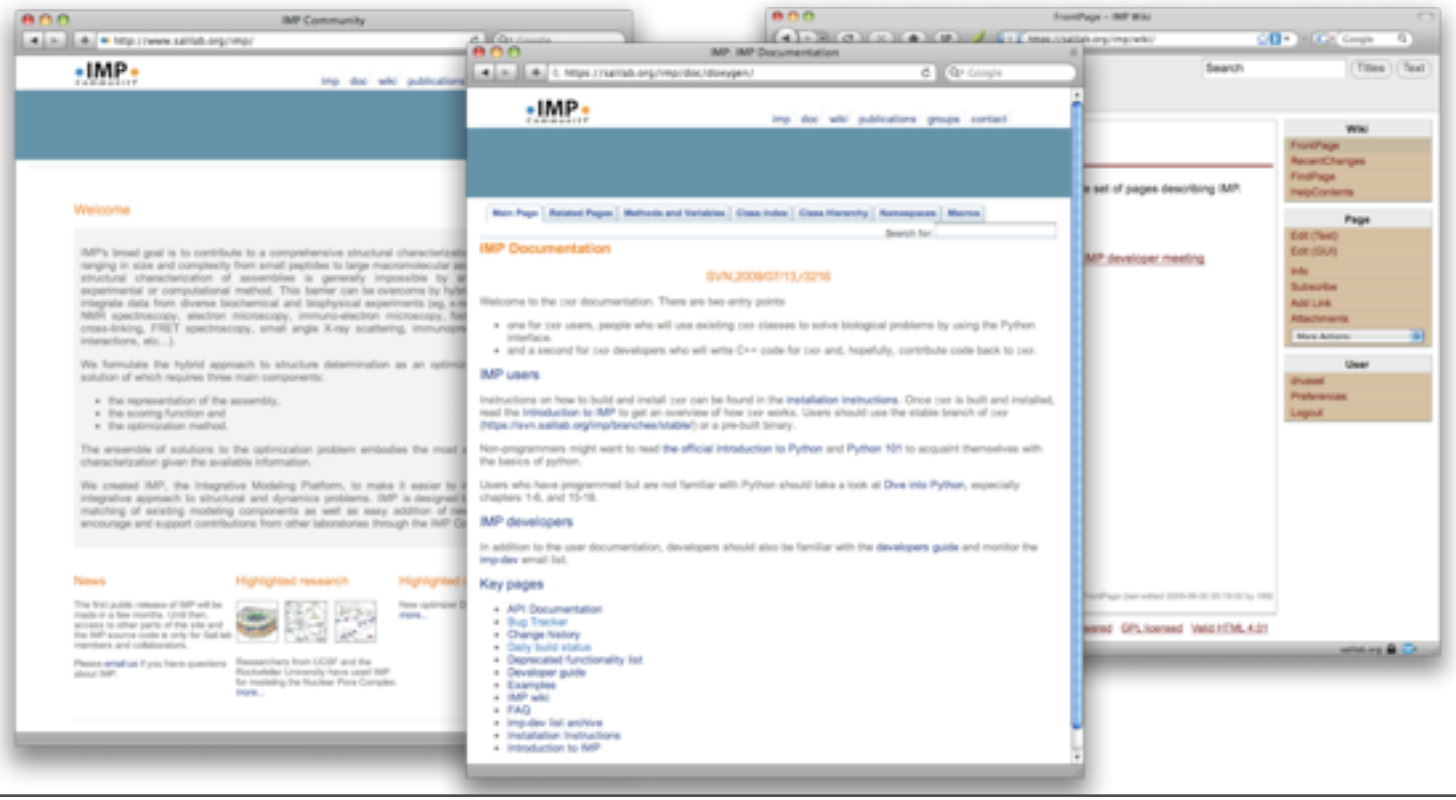
Integrative modeling - IMP



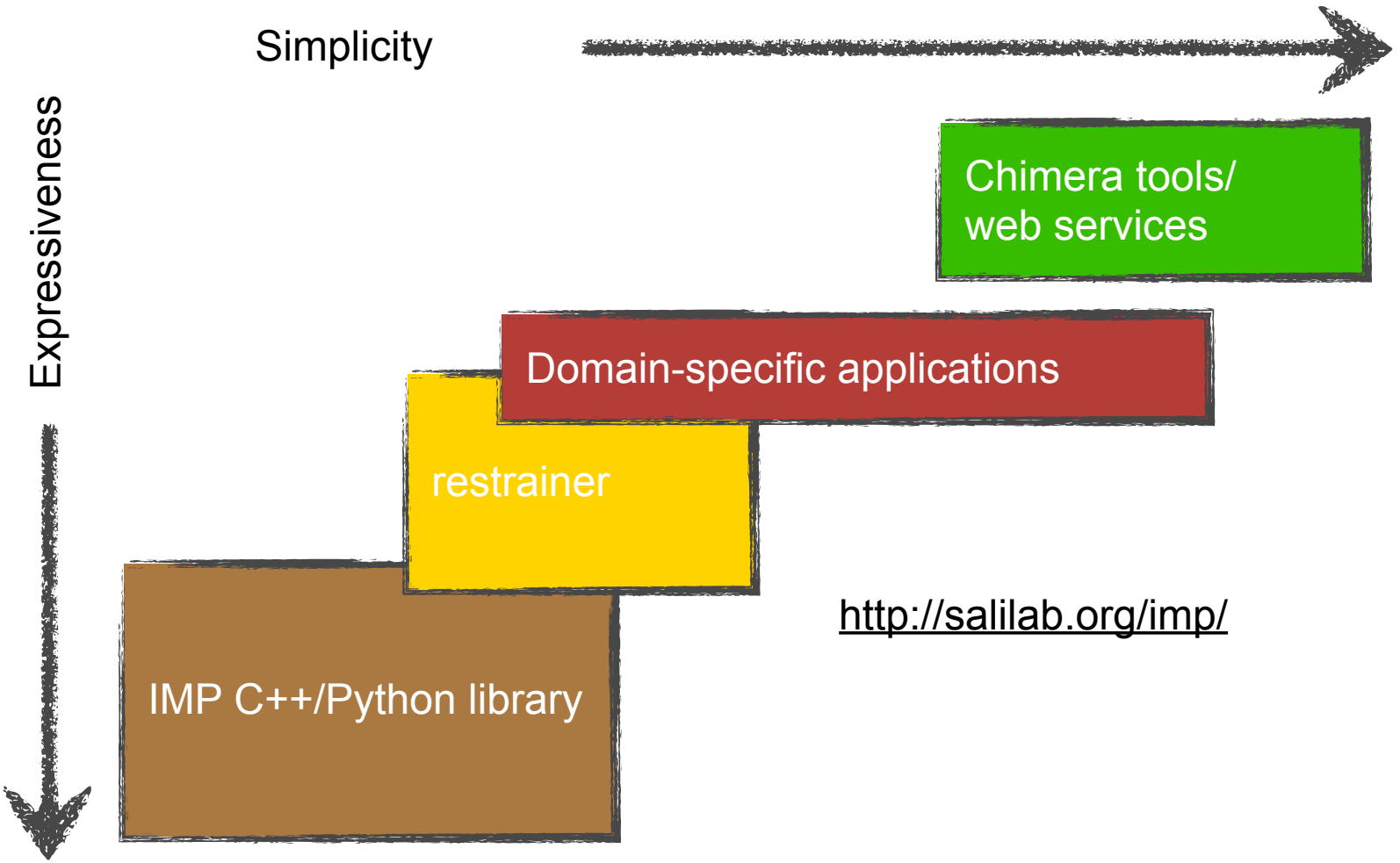
Alber *et al*. *Nature* 2007 • Robinson, Sali, Baumeister. *Nature* 2007 •
Russel, et al. Current Option in Cell Biology, 2009

IMP release

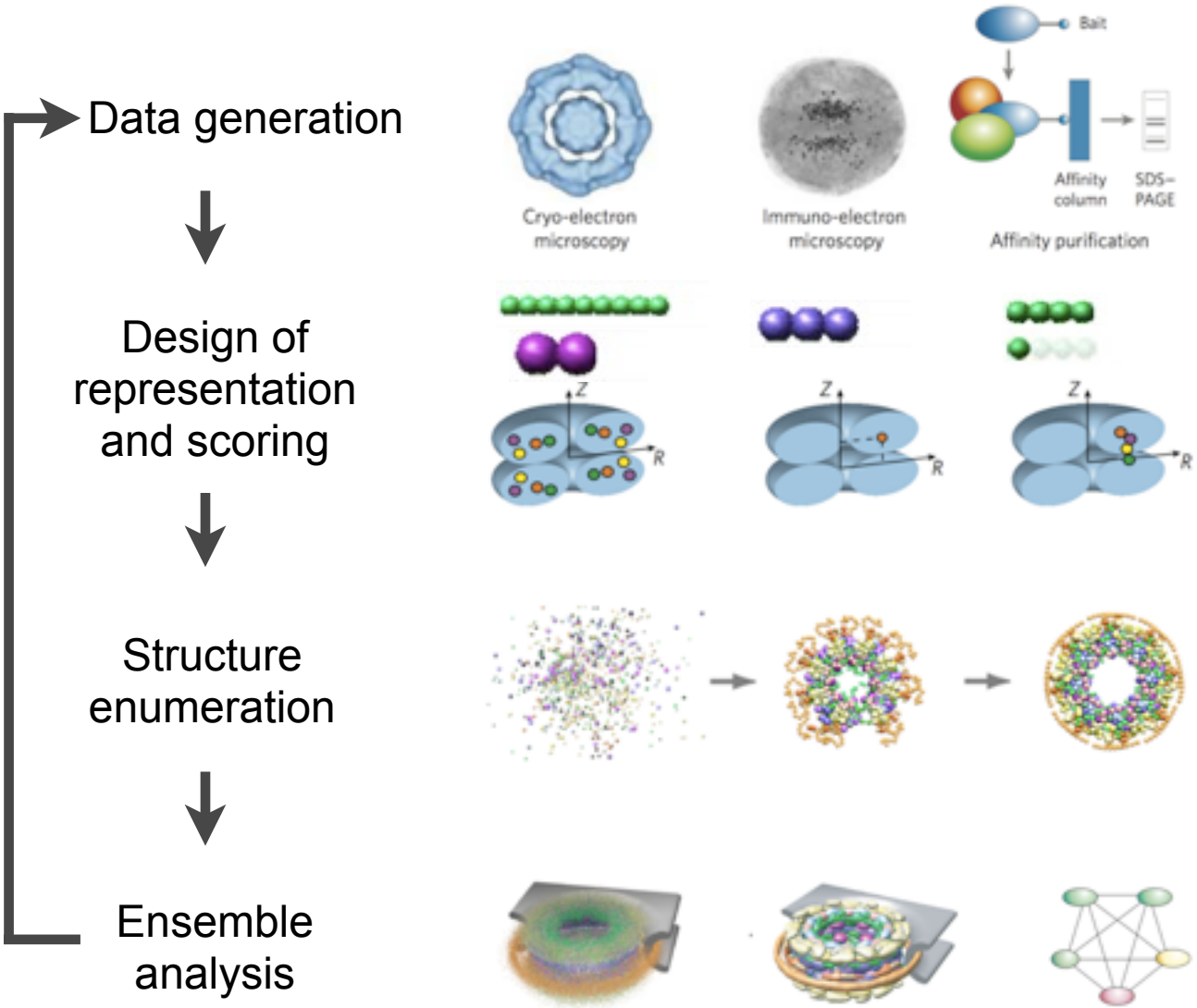
- IMP 1.0 is available as open source (LGPL)
- Binaries (Mac/Windows/Linux), source, SVN, documentation, wiki, examples, mailing lists, unit testing, bug tracking...



Presenting IMP



C++/Python library



Alber *et al*. *Nature* 2007 • Robinson, Sali, Baumeister. *Nature* 2007 • Russel, et al. Current Option in Cell Biology, 2009

C++/Python library

Data generation

↓

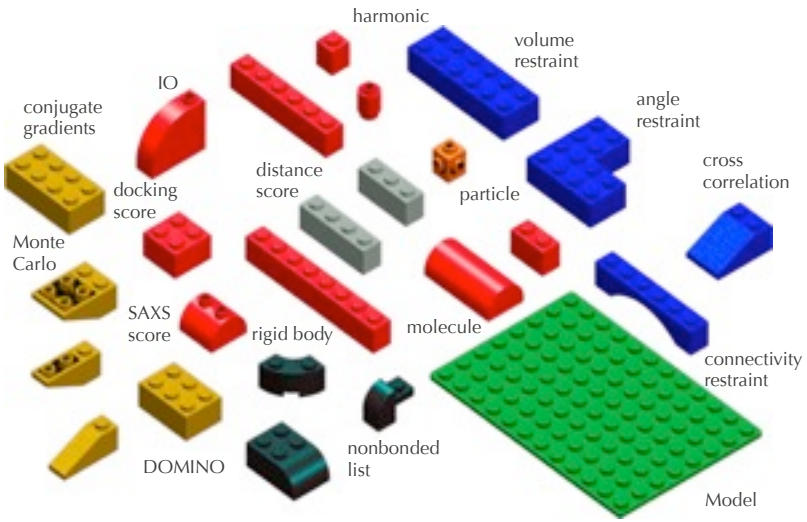
Design of representation and scoring — IMP.Particle, IMP.Restraint

↓

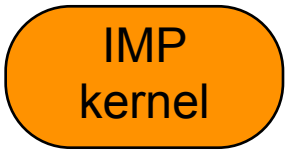
Structure enumeration — IMP.Sampler

↓

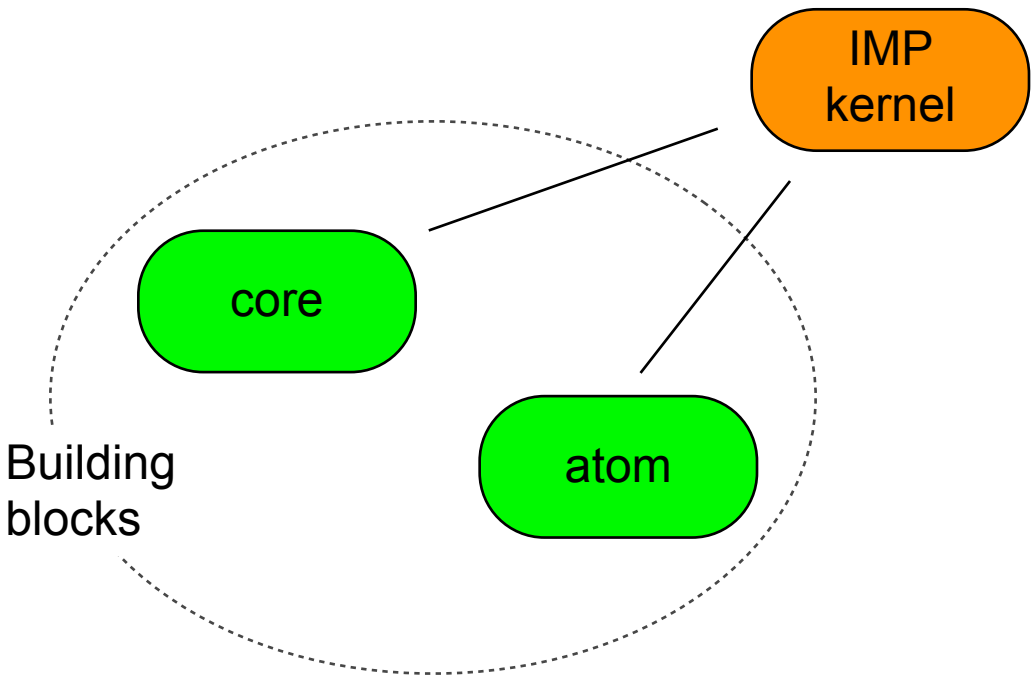
Ensemble analysis — IMP.statistics



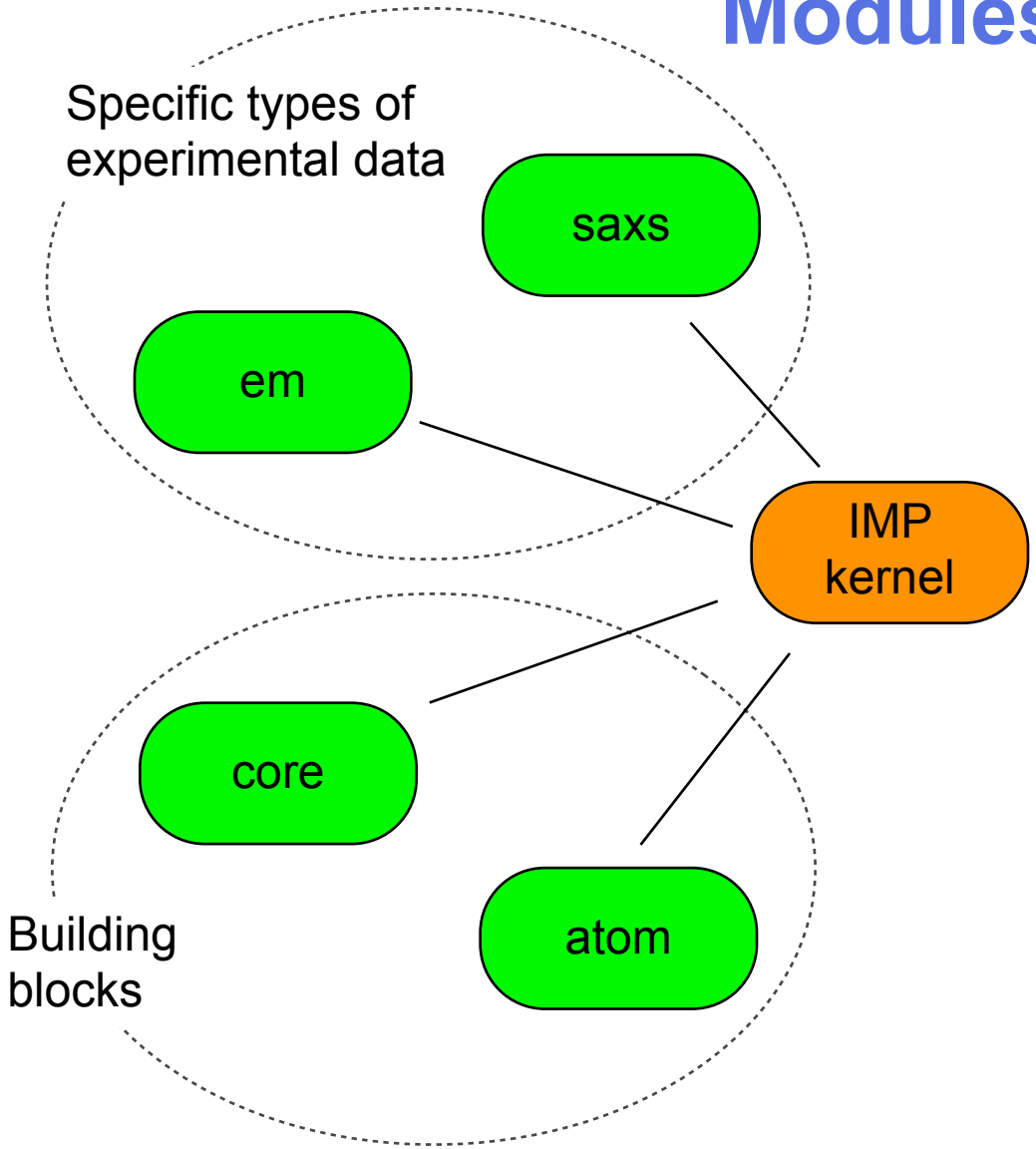
Modules



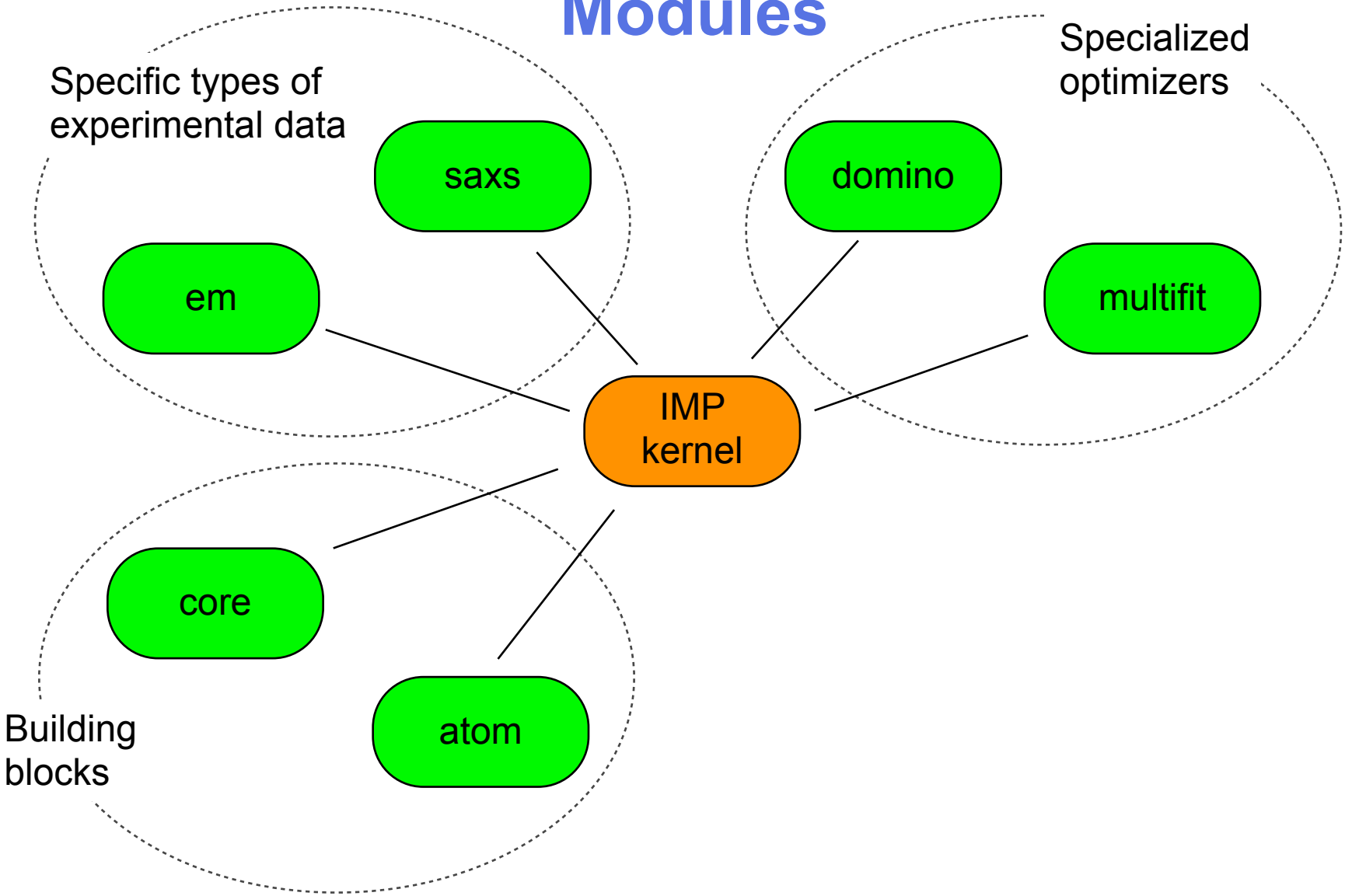
Modules



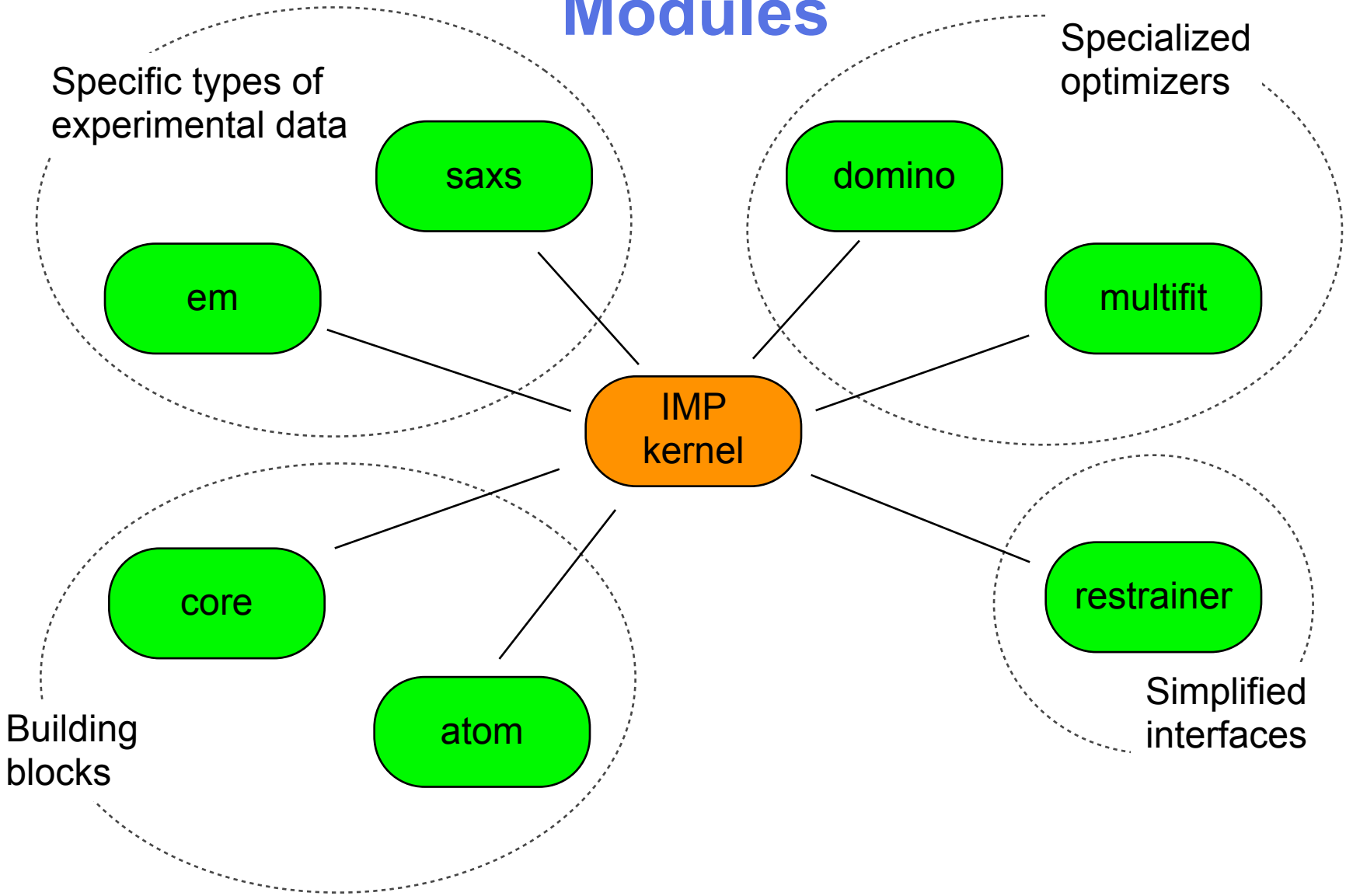
Modules


Specific types of experimental data
saxs
em
IMP kernel
core
atom
Building blocks

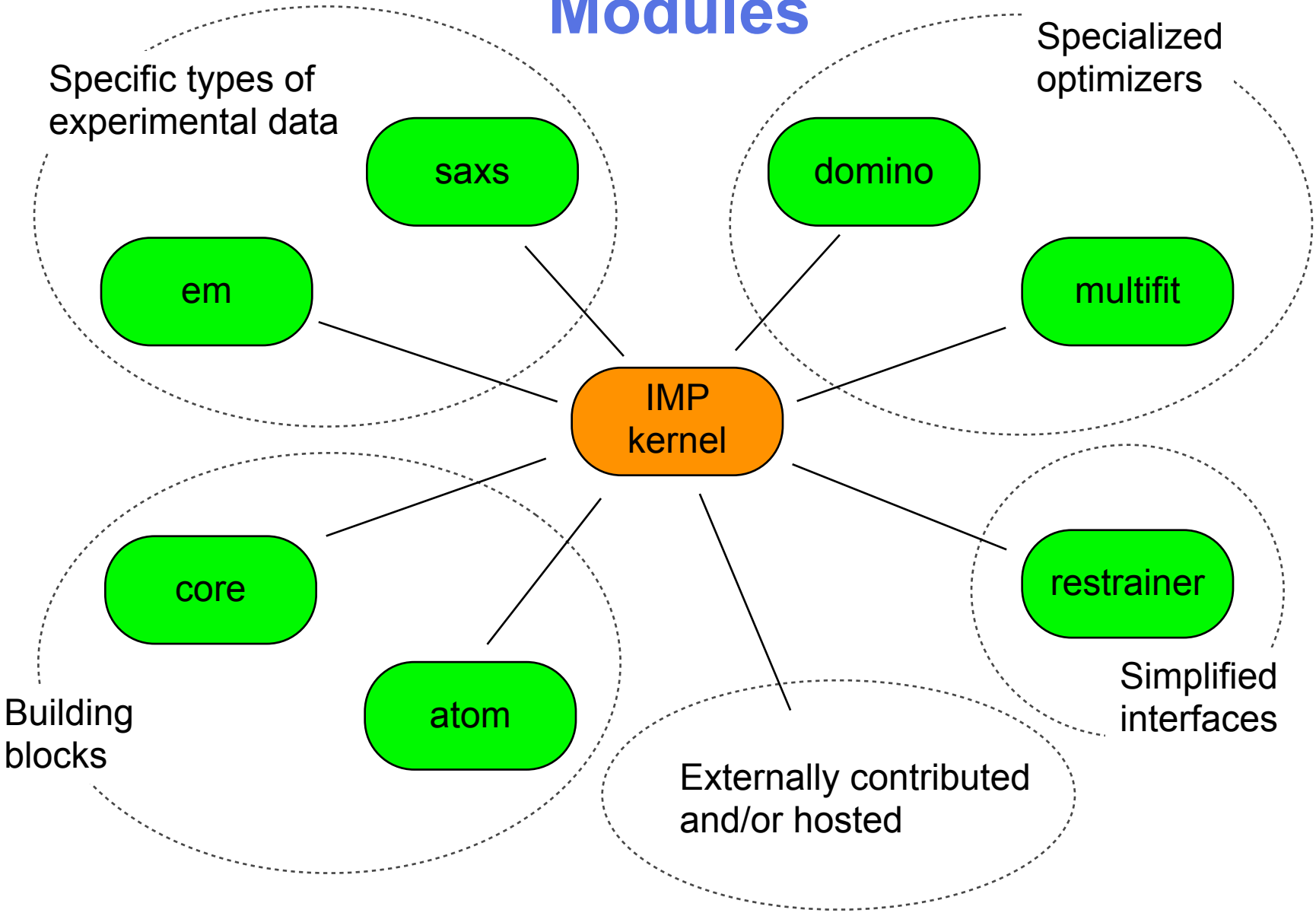
Modules


Specific types of experimental data
saxs
em
Specialized optimizers
domino
multifit
IMP kernel
core
atom
Building blocks

Modules

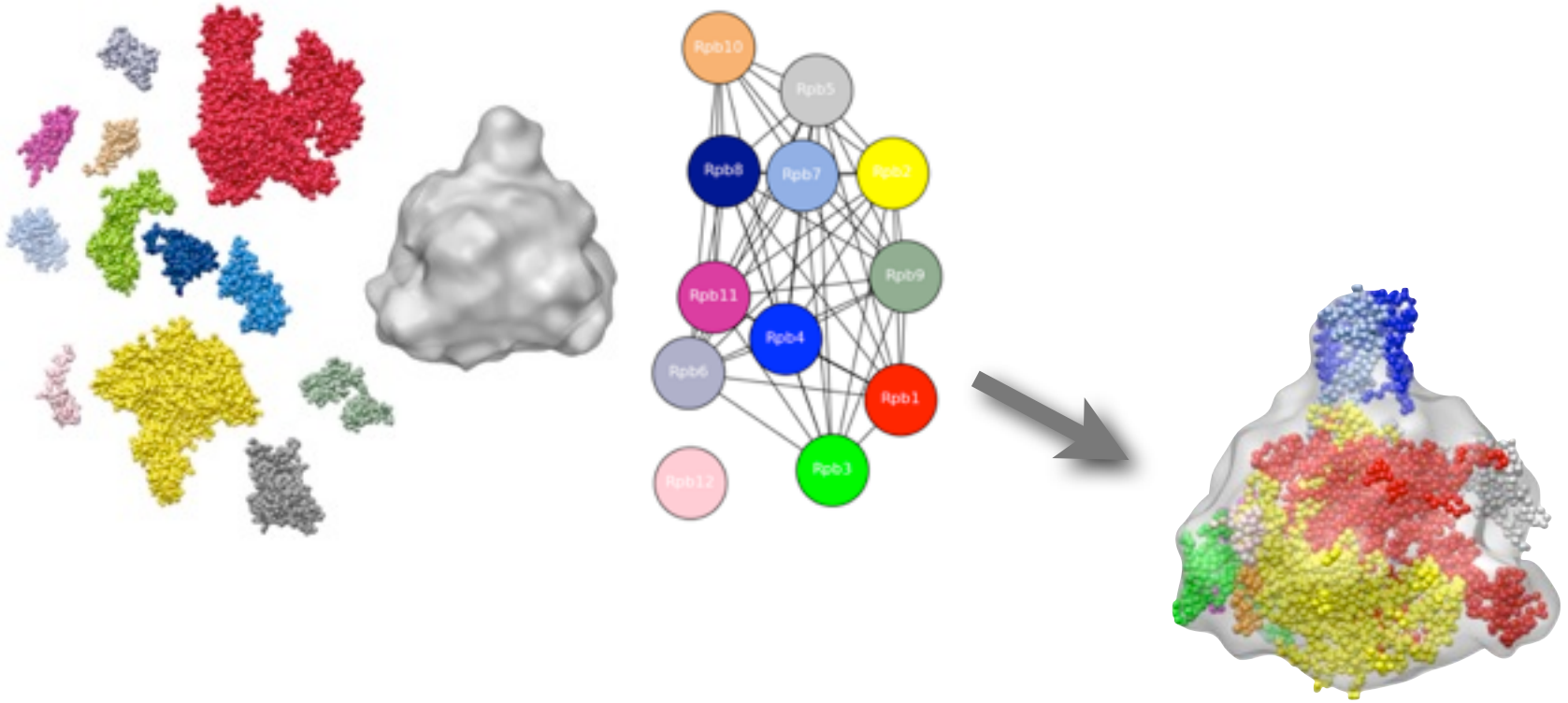

Specific types of experimental data
saxs
em
Specialized optimizers
domino
multifit
IMP kernel
core
atom
Building blocks
restrainer
Simplified interfaces

Modules


Specific types of experimental data
saxs
em
Specialized optimizers
domino
multifit
IMP kernel
core
atom
Building blocks
restrainer
Simplified interfaces
Externally contributed and/or hosted

Example: MultiFit module

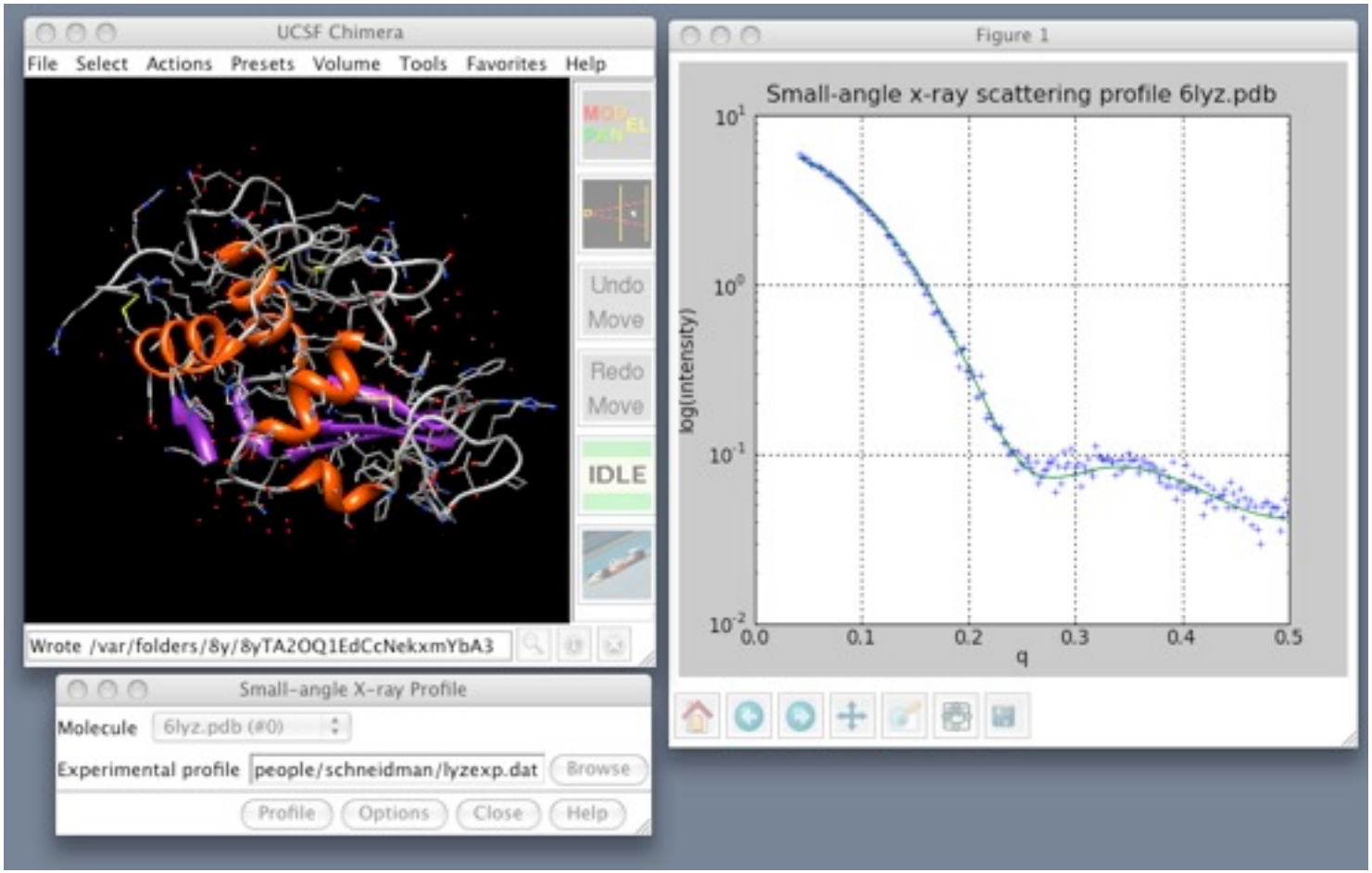
Input: components (atomic or low resolution), cryo-EM map, interaction data (proteomics)



Output: component configuration, to be refined.

Keren Lasker, JMB **388** (2009)

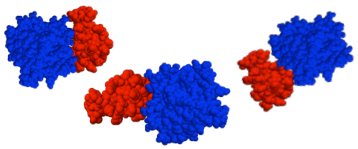
Example: SAXS profiles



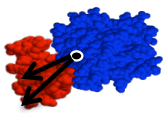
Example: SAXS and docking

Input structures

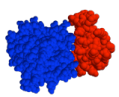
Global Search by Rigid Docking

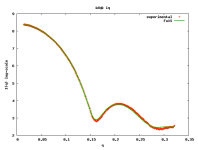


Coarse SAXS Filtering by Radius of Gyration

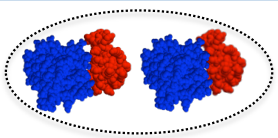


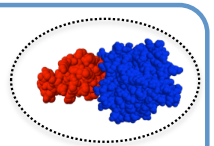
SAXS Scoring by Profile Fitting



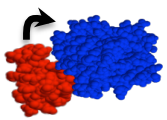


Clustering



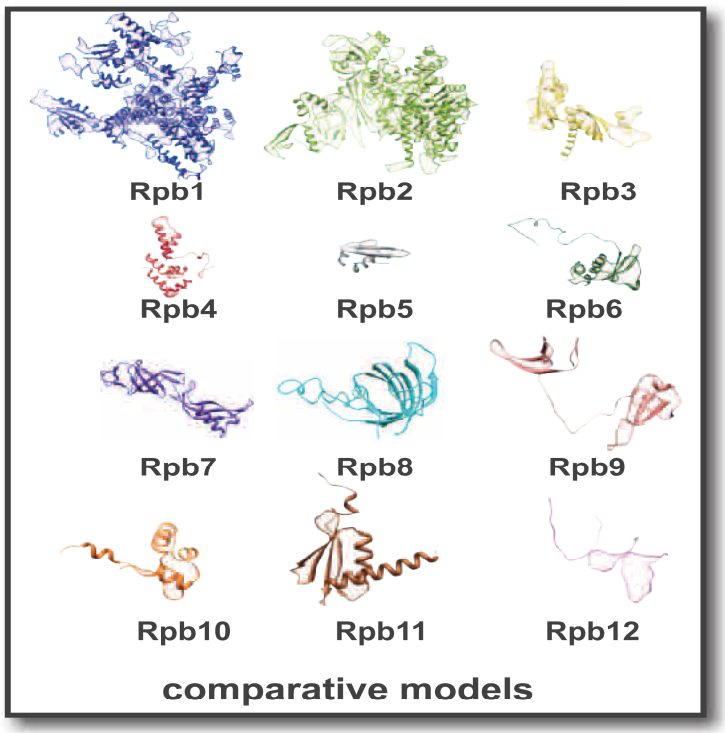


Local Search by Conformational Refinement

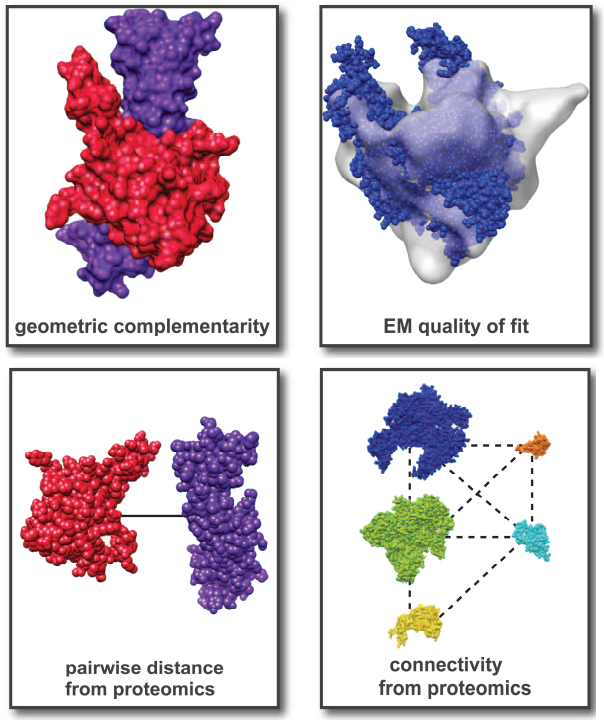


“Restrainer” simplified interface

Representation



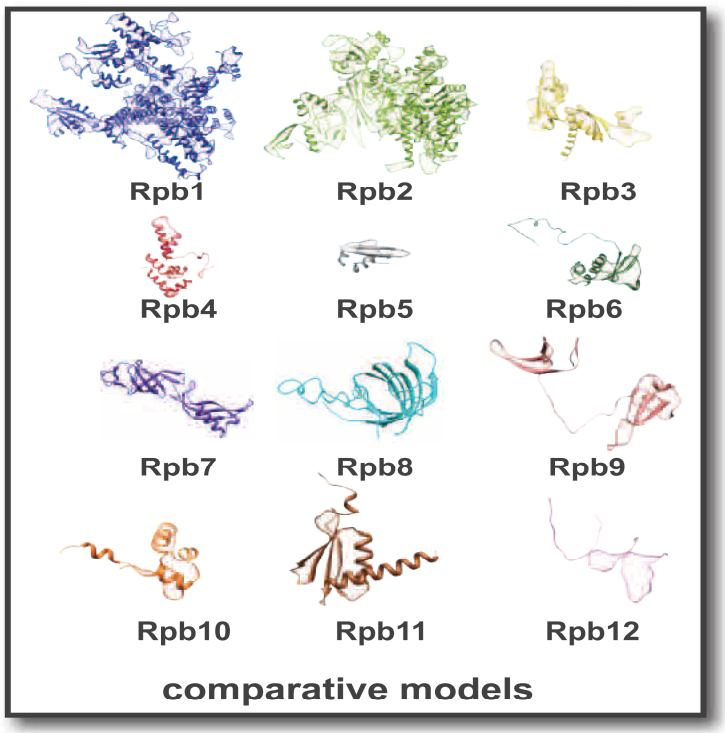
Data Translation into Spatial Restraints



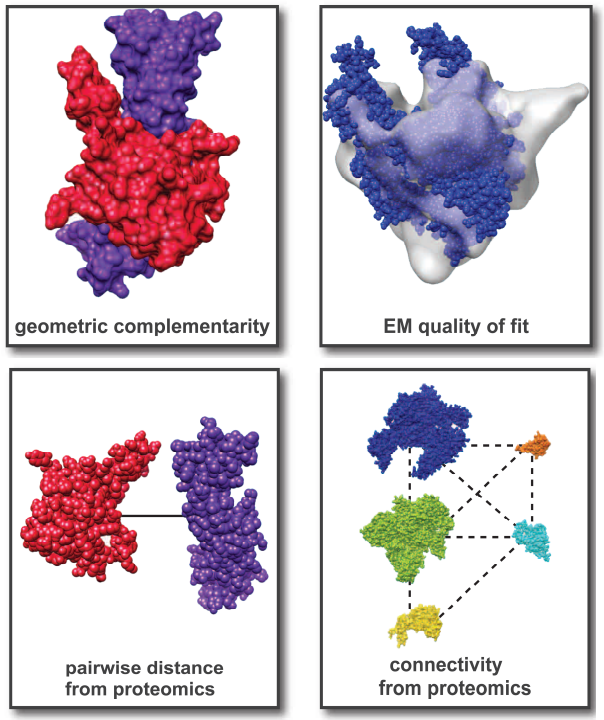
Elina Tjioe, Keren Lasker

“Restrainer” simplified interface

Representation



Data Translation into Spatial Restraints

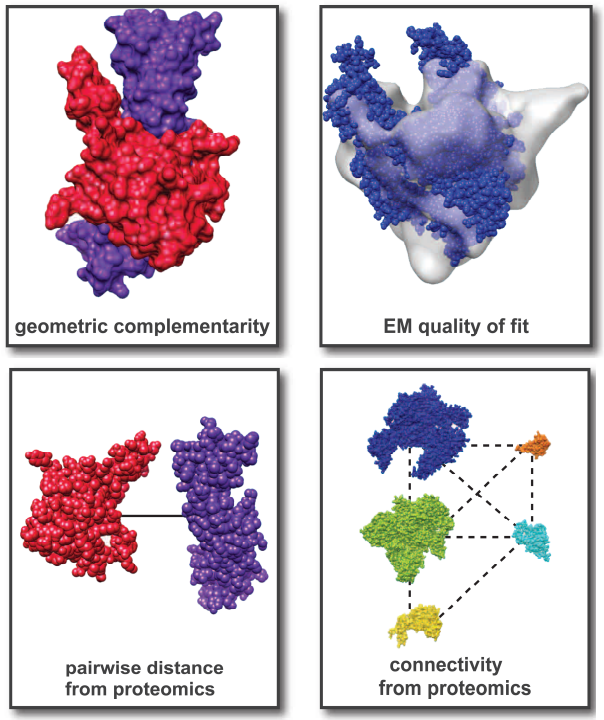



```
<Representation>
    <Protein id="Rpb1"><Chain filename="Rpb1.pdb"/></Protein>
        .
        .
        .
    <Protein id="Rpb12"><Chain filename="Rpb12.pdb"/></Protein>
</Representation>
```

“Restrainer” simplified interface

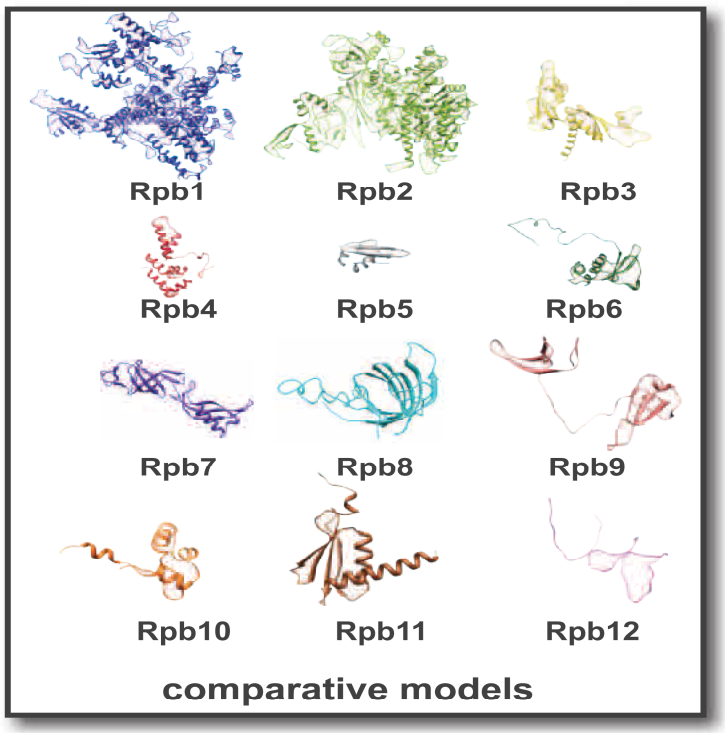
```
<RestraintSet>
    <EM>
        <Restraint density_filename="in.mrc">
            <Particle id="Rpb1"/>
        </Restraint>
    </EM>
    <Distance>
        <Restraint distance="5.0" std_dev="0.1">
            <Particle id="Rpb1"/>
            <Particle id="Rpb4"/>
        </Restraint>
    </Distance>
    <Y2H>
        <Restraint>
            <Particle id="Rpb2"/>
            <Particle id="Rpb3"/>
            <Particle id="Rpb8"/>
        </Restraint>
    </Y2H>
</RestraintSet>
```

Data Translation into Spatial Restraints

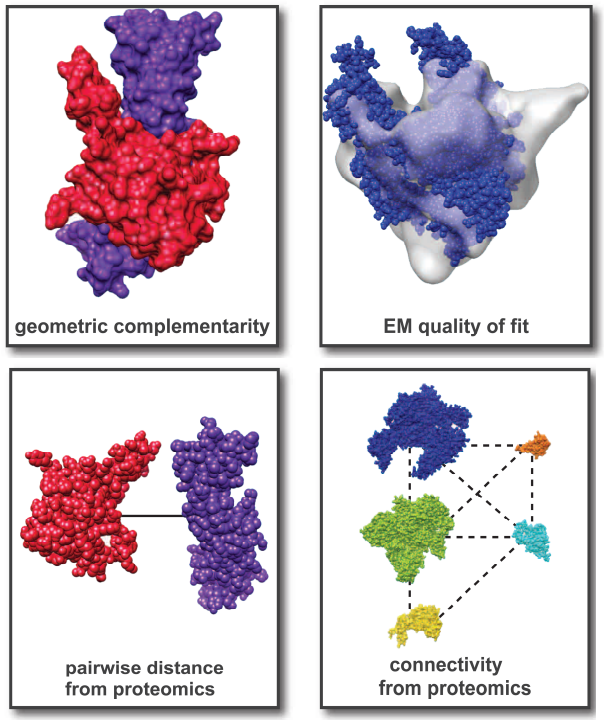


“Restrainer” simplified interface

Representation

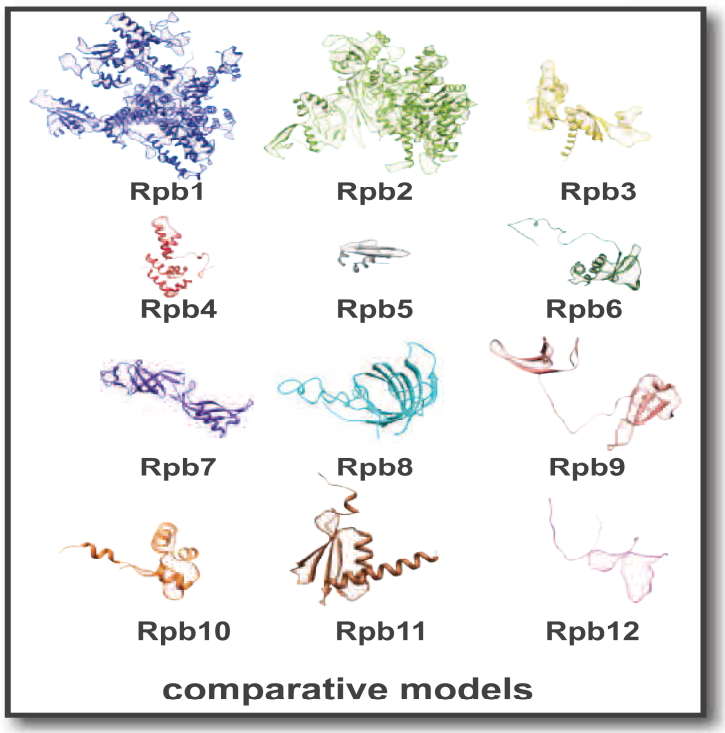


Data Translation into Spatial Restraints

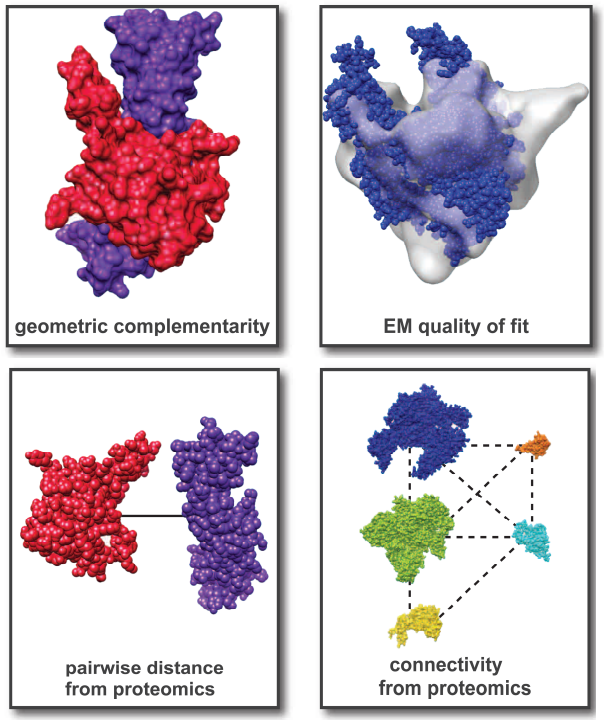


“Restrainer” simplified interface

Representation



Data Translation into Spatial Restraints




```
representation = IMP.restrainer.XMLRepresentation("repr.xml").run()
restraint = IMP.restrainer.XMLRestraint("restraint.xml").run()

model = representation.to_model()
restraint.add_to_representation(representation)

s = IMP.multifit.run_optimization(model)
IMP.statistics.analyze_solution_ensemble(s)
```

Summary

- Both IMP and MODELLER aim to model structures that are not yet determined experimentally
  - MODELLER builds protein models from template structures (comparative modeling)
  - IMP integrates multiple sources of information to build models of protein assemblies at a variety of resolutions
- Web interfaces are available for some specialized tasks
- More information on our website, http://salilab.org/