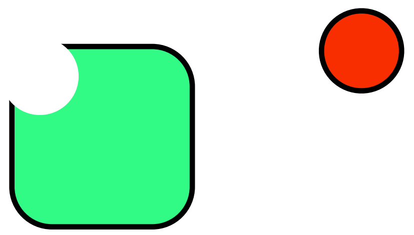
Free energy calculations with *alchemlyb*

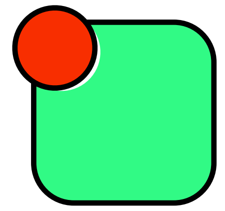
Oliver Beckstein
Arizona State University

SPIDAL Teleconference
2019-02-01

Binding free energy



$$P + X \overset{\Delta F}{\rightleftharpoons} PX$$



- measure of how strong a protein *P* and a "ligand" *X* stick together

- key quantity in quantitative mechanistic explanations of biological processes and in drug discovery

- **free** energy (i.e., averaged over all other degrees of freedom (such as solvent, protein motions, …))

$$\exp[-A(T,V,N)/kT] = \int dp^{3N} dx^{3N} e^{-H(p,x)/kT} \qquad H(p,x) = \sum_{i=1}^{N} \frac{\mathbf{p}_i^2}{2m_i} + U(x)$$

- **free energy difference**

$$\Delta A = A_{\textbf{bound}} - A_{\textbf{unbound}}$$

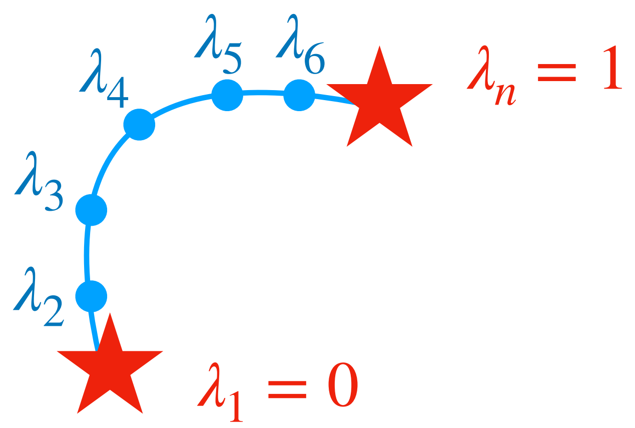
$$\Delta A = -\ln \frac{\int dx^{3N} \exp[-U(x)/kT]\,\chi_{\textbf{bound}}(x)}{\int dx^{3N} \exp[-U(x)/kT]\,\chi_{\textbf{unbound}}(x)}$$

Alchemical free energy calculations

- force fields for *H*, molecular dynamics (MD) simulations for sampling
- free energy is a **state function:** generate non-physical paths between physical end states (bound/unbound)
- use **stratification** ("windows") of path with parameter λ

$$H(\lambda) = (1-\lambda)H_{\textbf{bound}} + \lambda H_{\textbf{unbound}}, \quad 0 \leq \lambda \leq 1$$

$$U_{\textbf{Coulomb}}(\mathbf{x}_1, \mathbf{x}_2; \lambda) = \frac{1}{4\pi\epsilon_0}\frac{(1-\lambda)q_1q_2}{|\mathbf{x}_1 - \mathbf{x}_2|}$$



Methods

- "Free Energy Perturbation" (FEP): Zwanzig FEP, BAR, MBAR (overlaps of distributions)

$$\Delta U(x) = U_{\lambda_{n+1}}(x) - U_{\lambda_n}(x)$$

$$\Delta A = -kT \ln\langle \exp[-\Delta U(x)/kT]\rangle_1, \quad \textbf{with } \Delta U(x) = U_1(x) - U_0(x)$$

FEP

$$\exp(-\Delta A/kT) = \frac{\langle 1 + \exp[(\Delta U - C)/kT]^{-1}\rangle_0}{\langle 1 + \exp[(\Delta U - C)/kT]^{-1}\rangle_1} \exp(-C/kT)$$

BAR

$$\Delta A/kT = C/kT - \ln \frac{n_1}{n_2}$$

(MBAR is more complicated: uses overlaps between all windows)

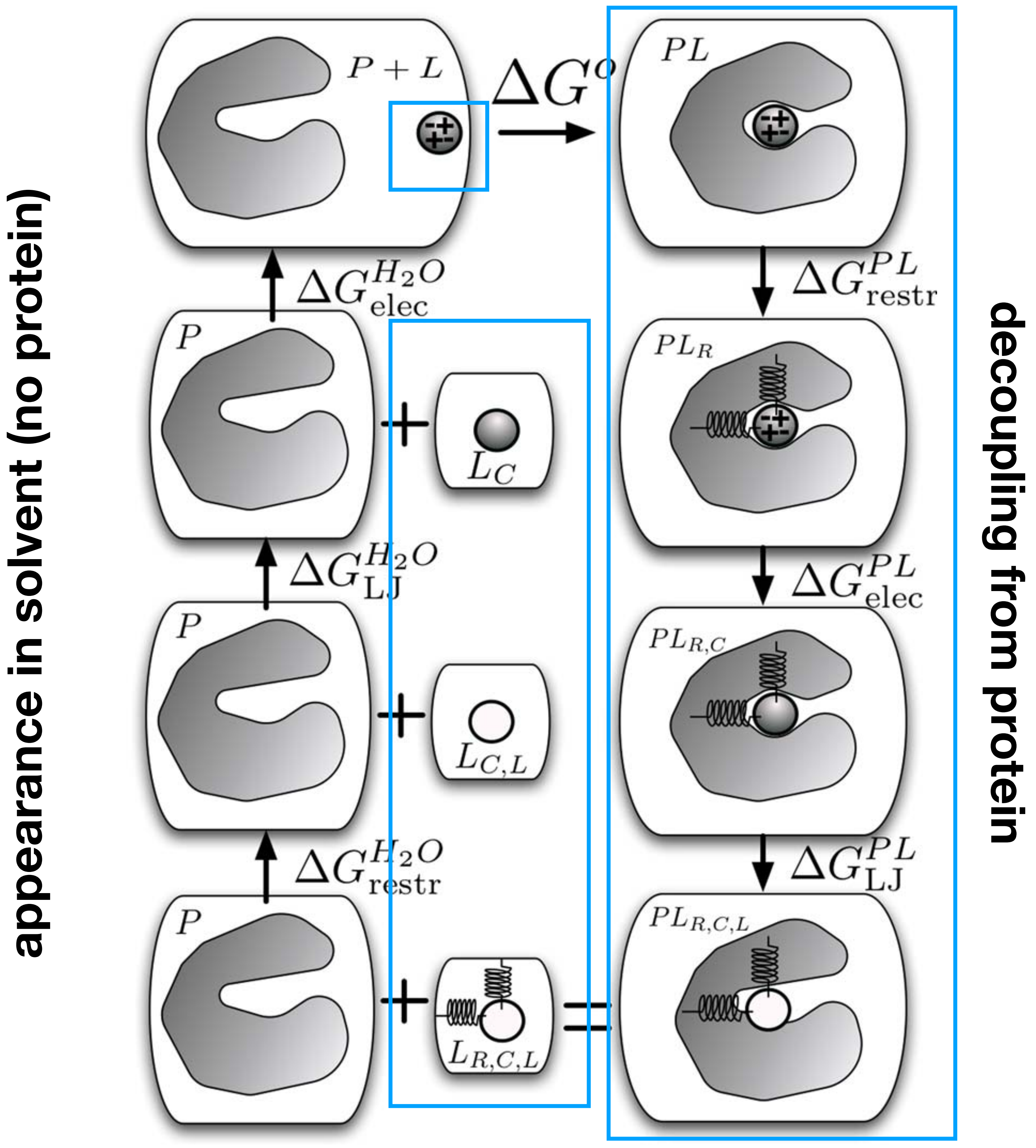
- Thermodynamic Integration (TI): TI

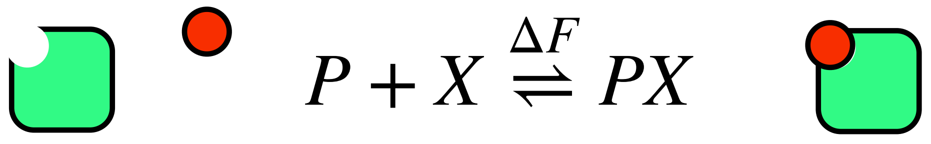
$$\Delta A = \int_0^1 d\lambda \left\langle \frac{\partial H(\lambda)}{\partial \lambda} \right\rangle_\lambda$$

TI

C. Chipot and A. Pohorille, editors. Free energy calculations. Number 86 in Springer Series in Chemical Physics. Springer, Berlin, 2007.

Thermodynamic cycle for absolute binding free energy calculations


appearance in solvent (no protein)
decoupling from protein
$P + L$
ΔG^o
PL
$\Delta G^{H_2O}_{\mathrm{elec}}$
$\Delta G^{PL}_{\mathrm{restr}}$
P
L_C
PL_R
$\Delta G^{H_2O}_{\mathrm{LJ}}$
$\Delta G^{PL}_{\mathrm{elec}}$
$L_{C,L}$
$PL_{R,C}$
$\Delta G^{H_2O}_{\mathrm{restr}}$
$\Delta G^{PL}_{\mathrm{LJ}}$
$L_{R,C,L}$
$PL_{R,C,L}$



- 5 – 50 λ-windows per free energy component
- 10 ns – 250 ns per window

D. L. Mobley, J. D. Chodera, and K. A. Dill. On the use of orientational restraints and symmetry corrections in alchemical free energy calculations. 125:084902, 2006.

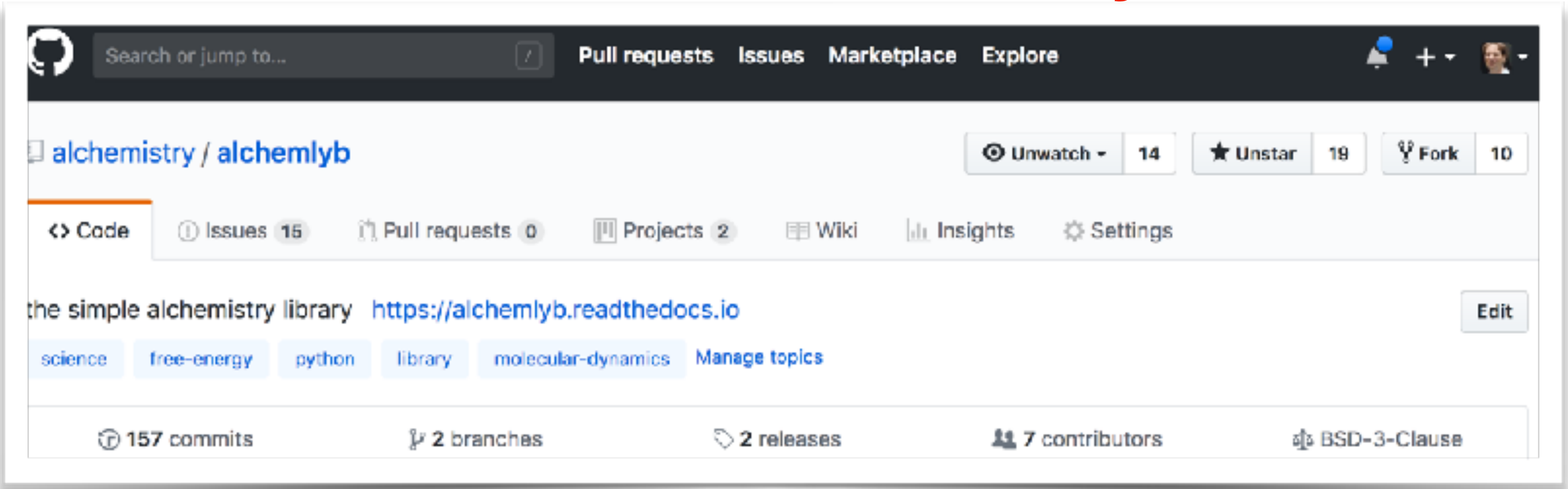
Simulation output

Per timestep

- FEP: Δ*U* $\Delta U_{\lambda_i,\lambda_j} \forall j$ for window *i*
- TI: ∂*U*/∂λ $\left.\frac{\partial U(x;\lambda)}{\partial \lambda}\right|_{\lambda_i}$

But: different file formats for different codes (Gromacs, Amber, NAMD, …)

solution: common interface via ***alchemlyb***



Search or jump to... Pull requests Issues Marketplace Explore

alchemistry / alchemlyb

Unwatch 14 Unstar 19 Fork 10

Code Issues 15 Pull requests 0 Projects 2 Wiki Insights Settings

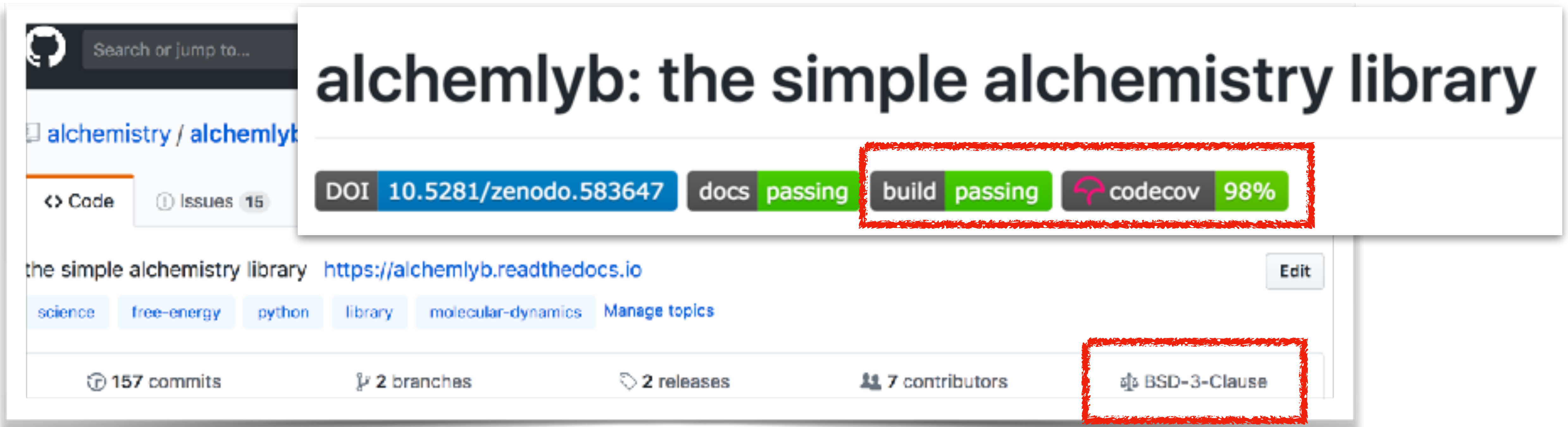
the simple alchemistry library https://alchemlyb.readthedocs.io Edit

science free-energy python library molecular-dynamics Manage topics

157 commits 2 branches 2 releases 7 contributors BSD-3-Clause

alchemlyb

- O Beckstein (ASU), D Mobley (UC Irvine), M Shirts (U Colorado, Boulder)
- **David Dotson** (ASU), Dominik Wille (Freie Univ. Berlin)
- Silicon Therapeutics (STX) (Bryce Allen, Shuai Liu)


alchemlyb: the simple alchemistry library

DOI 10.5281/zenodo.583647 | docs passing | build passing | codecov 98%

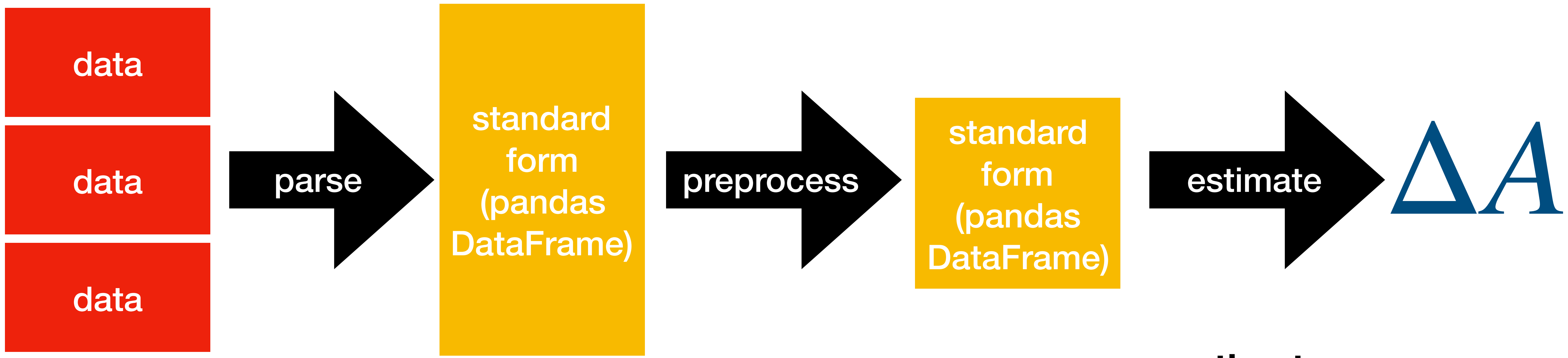
alchemistry / alchemlyb

the simple alchemistry library https://alchemlyb.readthedocs.io

science | free-energy | python | library | molecular-dynamics | Manage topics

157 commits | 2 branches | 2 releases | 7 contributors | BSD-3-Clause

alchemlyb: basic idea


data
data
data
parse
standard form (pandas DataFrame)
preprocess
standard form (pandas DataFrame)
estimate
ΔA

$\Delta U_{\lambda_i,\lambda_j} \forall j$

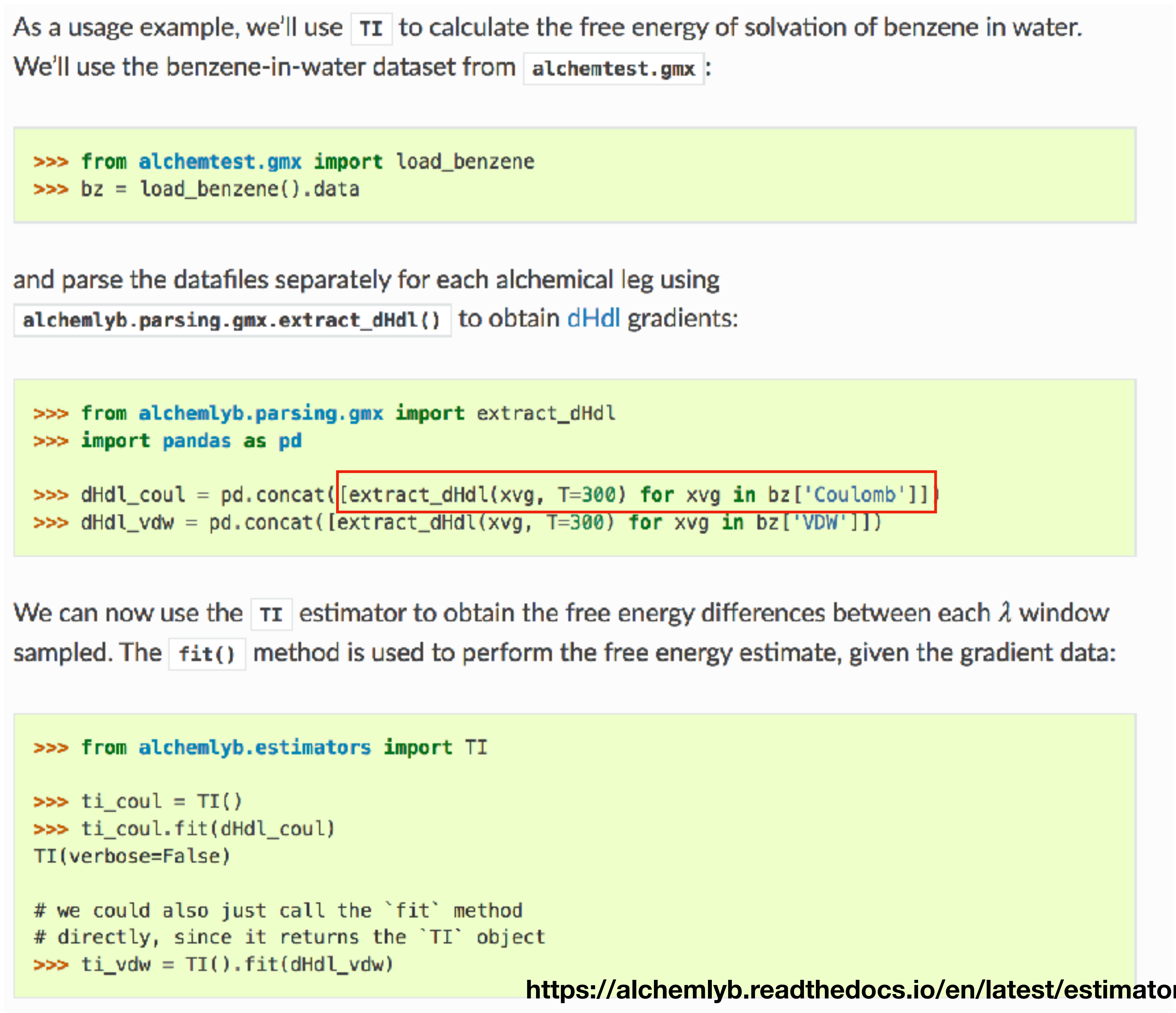
$$\left.\frac{\partial U(x;\lambda)}{\partial \lambda}\right|_{\lambda_i}$$

preprocessors

- **slicing**
- **statistical inefficiency**
- **equilibrium detection**
- **…**

estimators

- **TI**
- **BAR**
- **MBAR**
- **…**



As a usage example, we'll use `TI` to calculate the free energy of solvation of benzene in water. We'll use the benzene-in-water dataset from `alchemtest.gmx`:

```
>>> from alchemtest.gmx import load_benzene
>>> bz = load_benzene().data
```

and parse the datafiles separately for each alchemical leg using `alchemlyb.parsing.gmx.extract_dHdl()` to obtain dHdl gradients:

```
>>> from alchemlyb.parsing.gmx import extract_dHdl
>>> import pandas as pd

>>> dHdl_coul = pd.concat([extract_dHdl(xvg, T=300) for xvg in bz['Coulomb']])
>>> dHdl_vdw = pd.concat([extract_dHdl(xvg, T=300) for xvg in bz['VDW']])
```

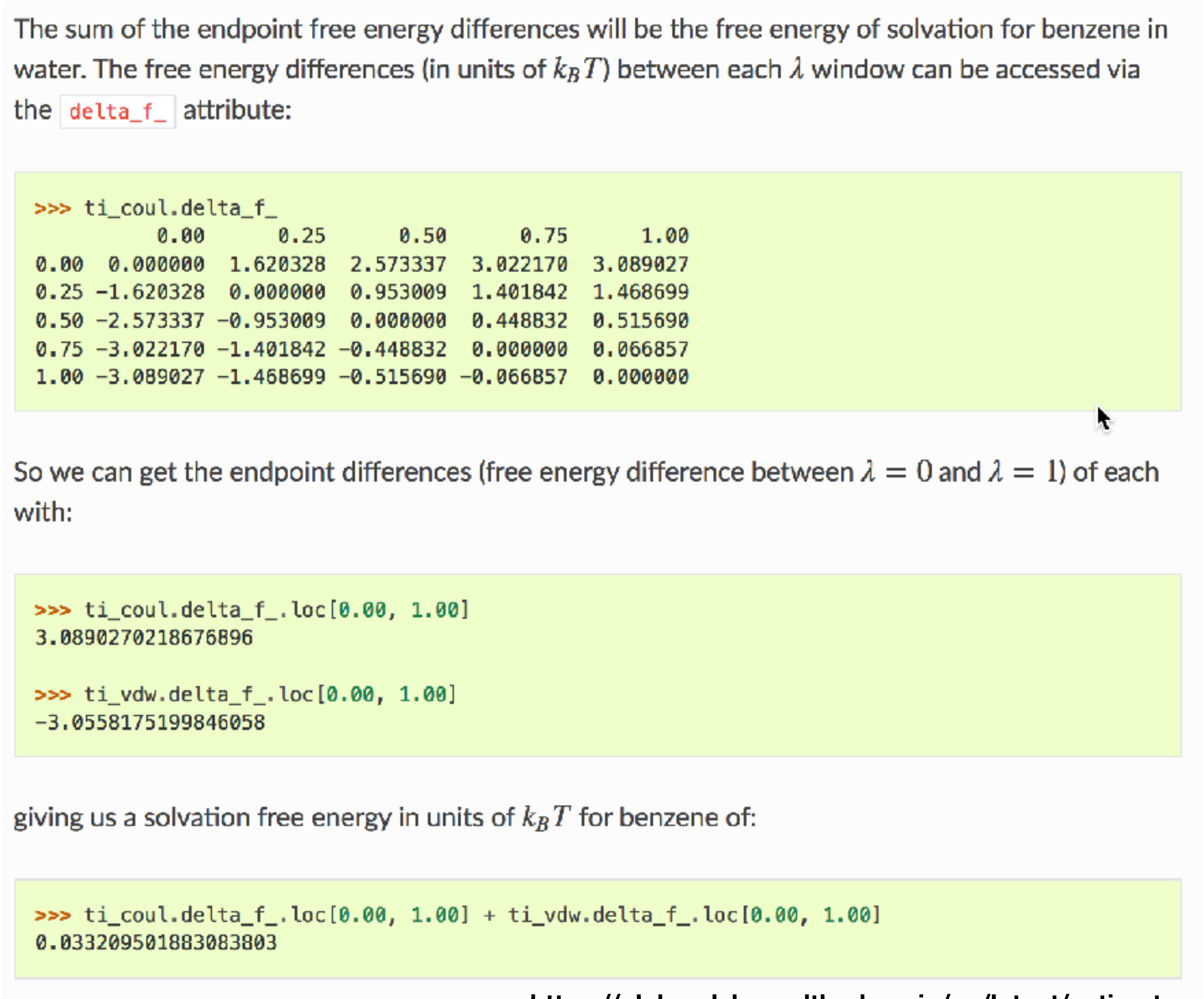
We can now use the `TI` estimator to obtain the free energy differences between each λ window sampled. The `fit()` method is used to perform the free energy estimate, given the gradient data:

```
>>> from alchemlyb.estimators import TI

>>> ti_coul = TI()
>>> ti_coul.fit(dHdl_coul)
TI(verbose=False)

# we could also just call the `fit` method
# directly, since it returns the `TI` object
>>> ti_vdw = TI().fit(dHdl_vdw)
```

https://alchemlyb.readthedocs.io/en/latest/estimators-ti.html



The sum of the endpoint free energy differences will be the free energy of solvation for benzene in water. The free energy differences (in units of $k_B T$) between each λ window can be accessed via the `delta_f_` attribute:

```
>>> ti_coul.delta_f_
          0.00      0.25      0.50      0.75      1.00
0.00  0.000000  1.620328  2.573337  3.022170  3.089027
0.25 -1.620328  0.000000  0.953009  1.401842  1.468699
0.50 -2.573337 -0.953009  0.000000  0.448832  0.515690
0.75 -3.022170 -1.401842 -0.448832  0.000000  0.066857
1.00 -3.089027 -1.468699 -0.515690 -0.066857  0.000000
```

So we can get the endpoint differences (free energy difference between $\lambda = 0$ and $\lambda = 1$) of each with:

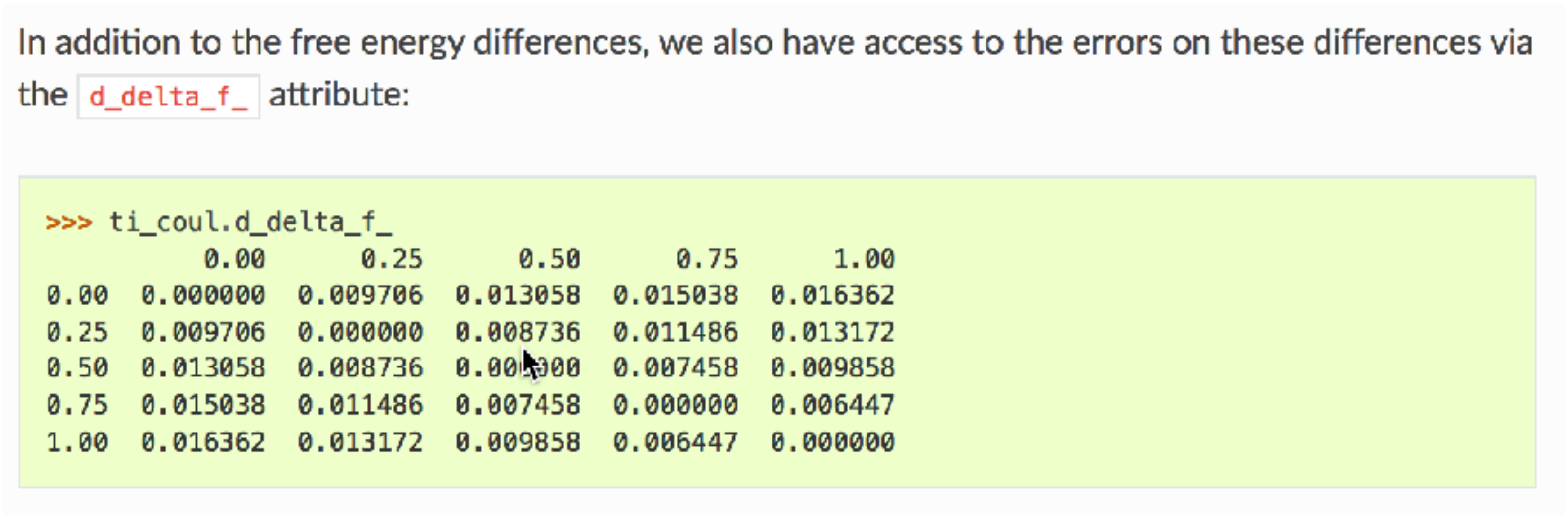
```
>>> ti_coul.delta_f_.loc[0.00, 1.00]
3.0890270218676896

>>> ti_vdw.delta_f_.loc[0.00, 1.00]
-3.0558175199846058
```

giving us a solvation free energy in units of $k_B T$ for benzene of:

```
>>> ti_coul.delta_f_.loc[0.00, 1.00] + ti_vdw.delta_f_.loc[0.00, 1.00]
0.0332095018830838O3
```

https://alchemlyb.readthedocs.io/en/latest/estimators-ti.html

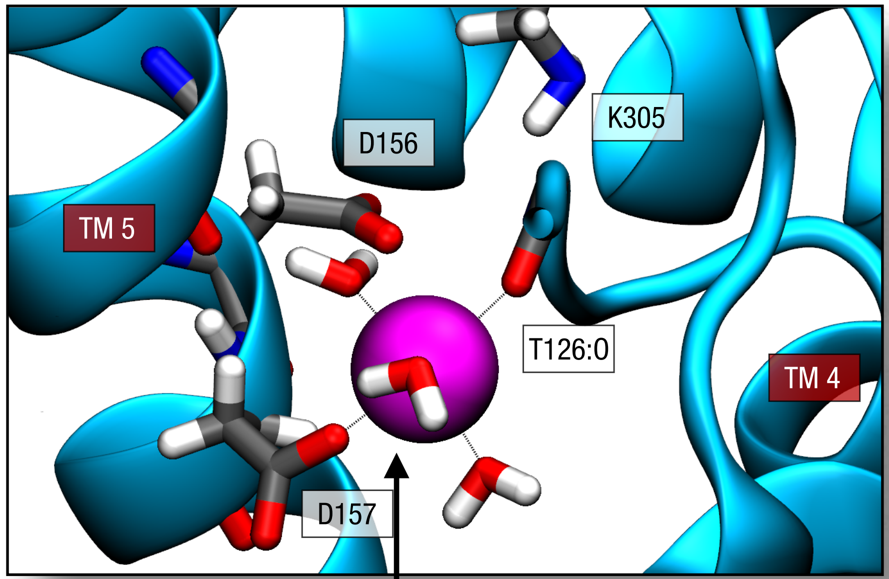


In addition to the free energy differences, we also have access to the errors on these differences via the `d_delta_f_` attribute:

```
>>> ti_coul.d_delta_f_
          0.00      0.25      0.50      0.75      1.00
0.00  0.000000  0.009706  0.013058  0.015038  0.016362
0.25  0.009706  0.000000  0.008736  0.011486  0.013172
0.50  0.013058  0.008736  0.000000  0.007458  0.009858
0.75  0.015038  0.011486  0.007458  0.000000  0.006447
1.00  0.016362  0.013172  0.009858  0.006447  0.000000
```

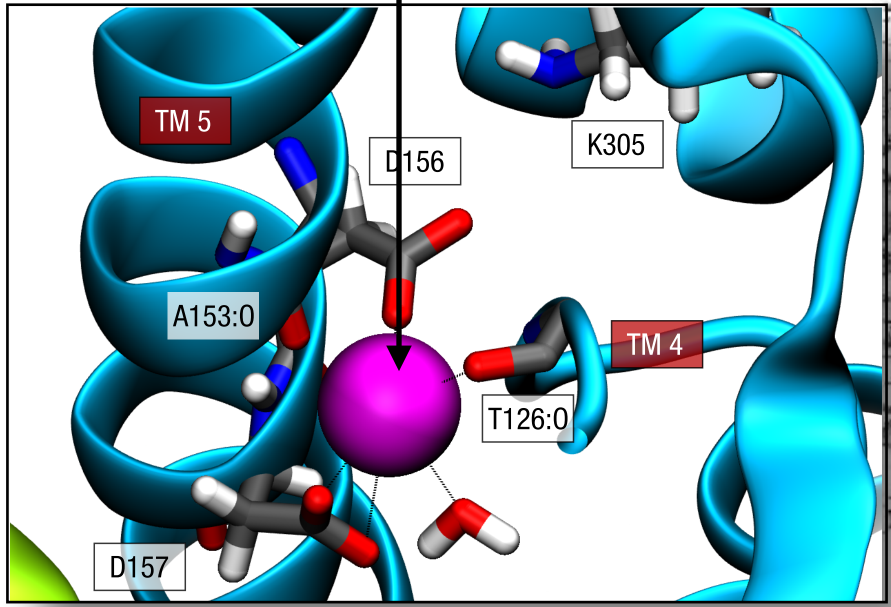
https://alchemlyb.readthedocs.io/en/latest/estimators-ti.html

NapA elevator mechanism

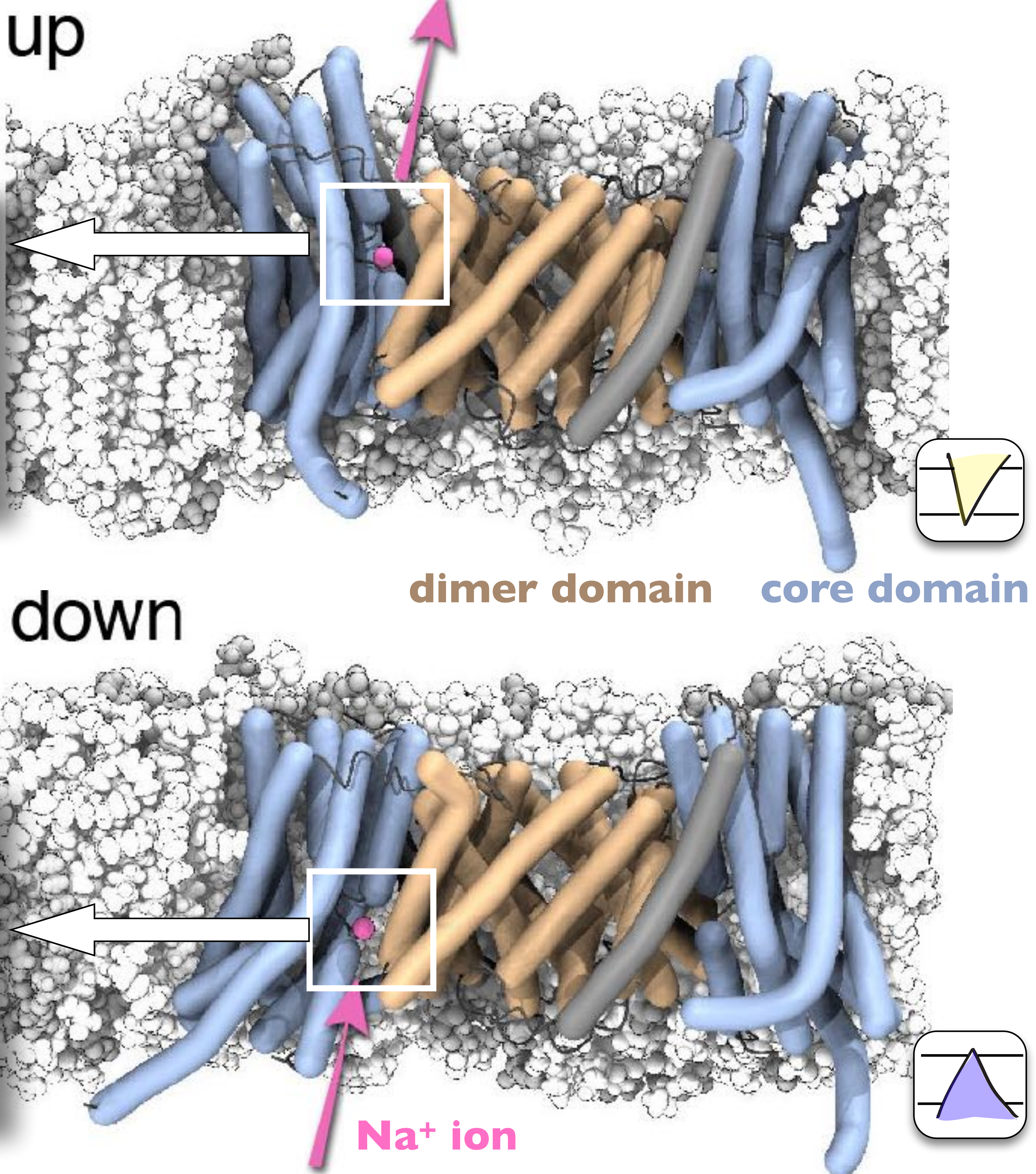


Nature Struct. Mol. Biol., 23 (2016):248–255

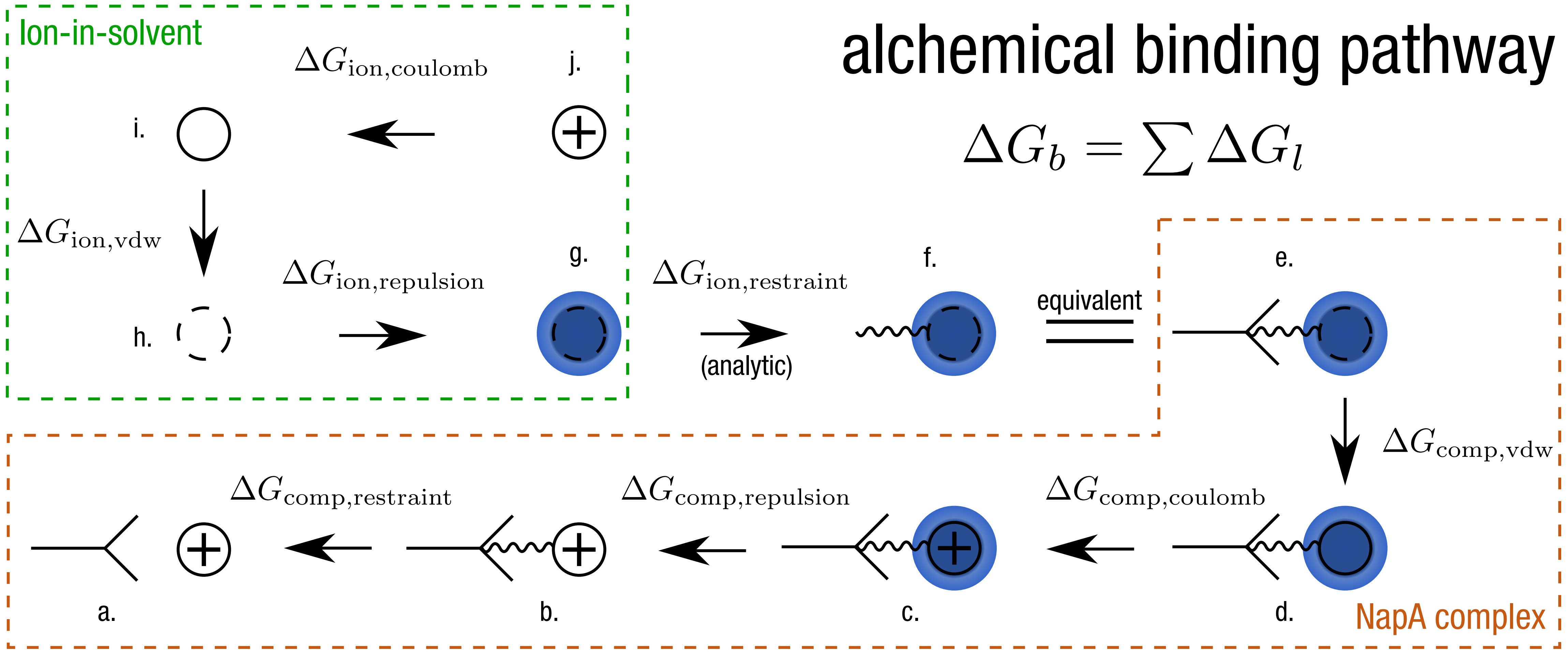
ion binding in different states?



Nature, 501 (2013):573–577.



Absolute binding free energies: alchemistry



Ion-in-solvent
$\Delta G_{\text{ion,coulomb}}$
j.
i.
$\Delta G_{\text{ion,vdw}}$
$\Delta G_{\text{ion,repulsion}}$
g.
h.
$\Delta G_{\text{ion,restraint}}$
(analytic)
f.
equivalent
e.
$\Delta G_{\text{comp,vdw}}$
$\Delta G_{\text{comp,restraint}}$
$\Delta G_{\text{comp,repulsion}}$
$\Delta G_{\text{comp,coulomb}}$
a.
b.
c.
d.
NapA complex

alchemical binding pathway

$$\Delta G_b = \sum \Delta G_l$$

- Windowed alchemical free energy calculations (TI or MBAR)
- 150 ns – 250 ns *per lambda window* (Coulomb/vdW decoupling) for 21+21 windows… ~8 μs (!)
- Position restraints (and analytical removal)
- Additional repulsive ion-ion potential to enforce one-ion occupancy (rigorously removed in calculation)

Amount of raw free energy data

NapA FEP simulations (one free energy)

	windows	time µs	size GB	total time µs	total size (GB)
VDW	21	0.25	3.865	5.25	81.165
Coulomb	21	0.25	3.865	5.25	81.165
repulsion	3	0.01	0.16	0.03	0.48
restraint	11	0.01	0.16	0.11	1.76
				10.64	**164.57**

- conformations: IF and OF (2)
- protonation states: 3
- repeats: x2 (some)
- ~12 sets of simulations: ~2 TB (in ~130,000 files)

Absolute binding free energies: alchemistry

https://github.com/alchemistry/alchemlyb

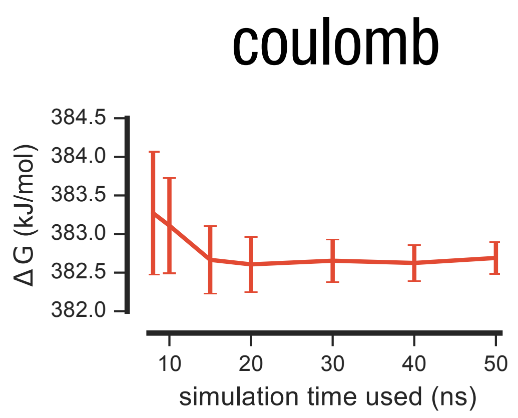
$$\boxed{\Delta G_i^0 = \Delta G^0_{\text{protein+ion},i} - \Delta G^0_{\text{hydration}}}$$

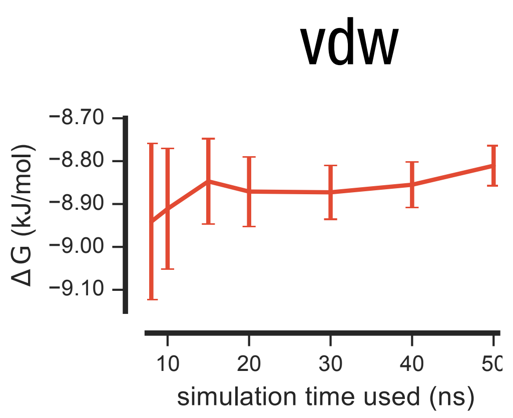
NapA IF

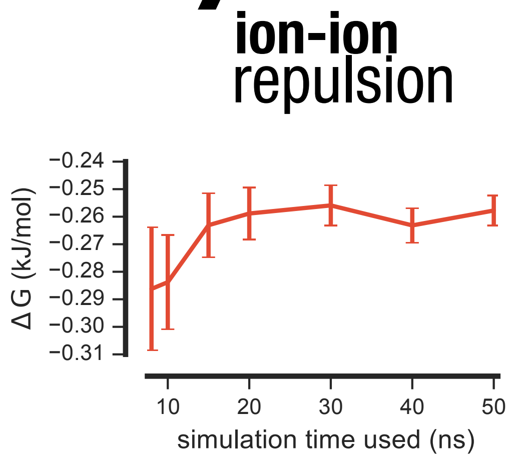
D156	D157	K305	ΔG^0 (kJ/mol)
∅	∅	∅	–103±1
∅	∅	H	–44.6±0.9
∅	H	H	–24.2±13.0

(not binding – ignore state)

Convergence of ion hydration calculation





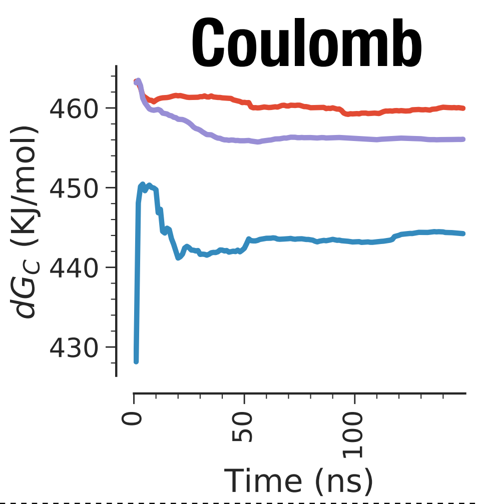


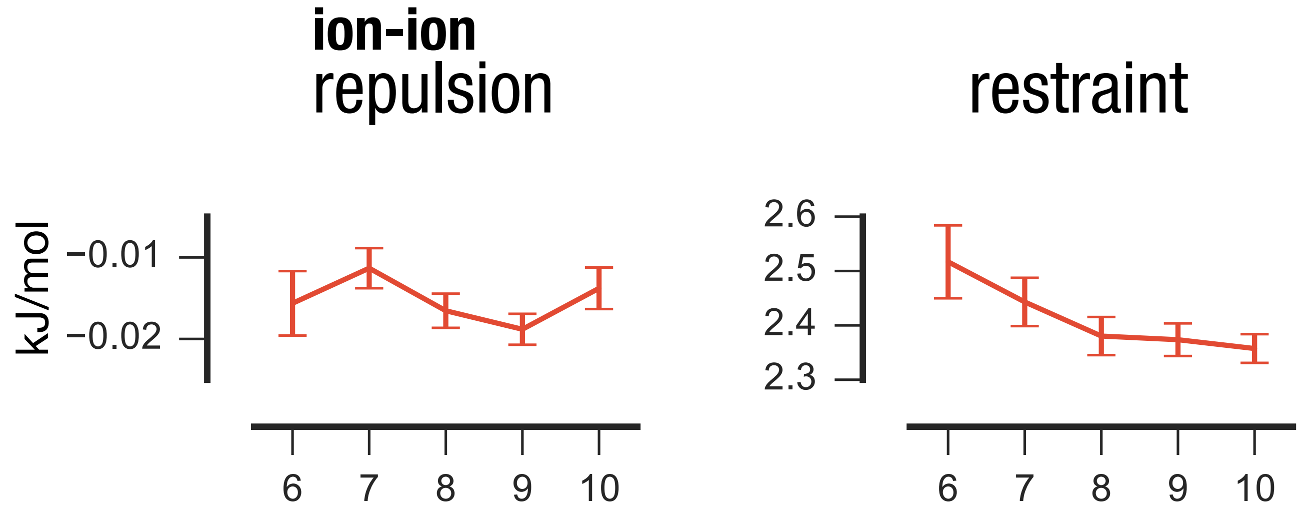
$\Delta G_{\text{hydration}}$ (kJ/mol)	
Coulomb	–382.7±0.2
VdW	8.81±0.05
repulsion	–0.258±0.005
restraint	–17.84
total	**–392.0±0.2**

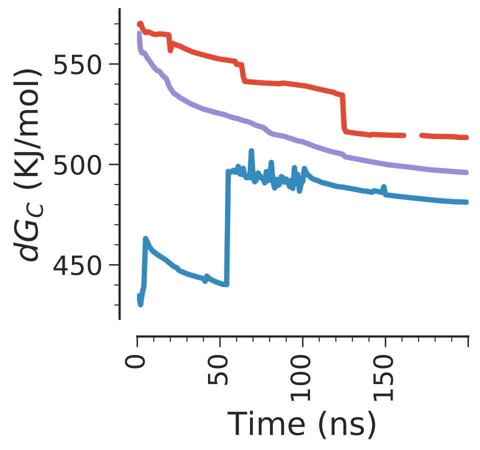
Na^+ ion in CHARMM TIP3P water

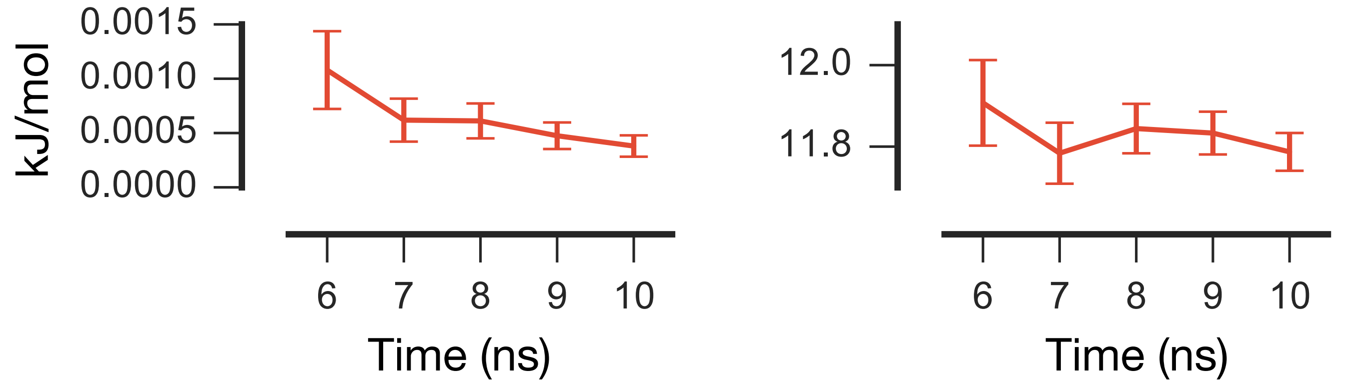
Convergence of protein–ion calculations

D156	D157	K305
∅	∅	H
∅	∅	∅







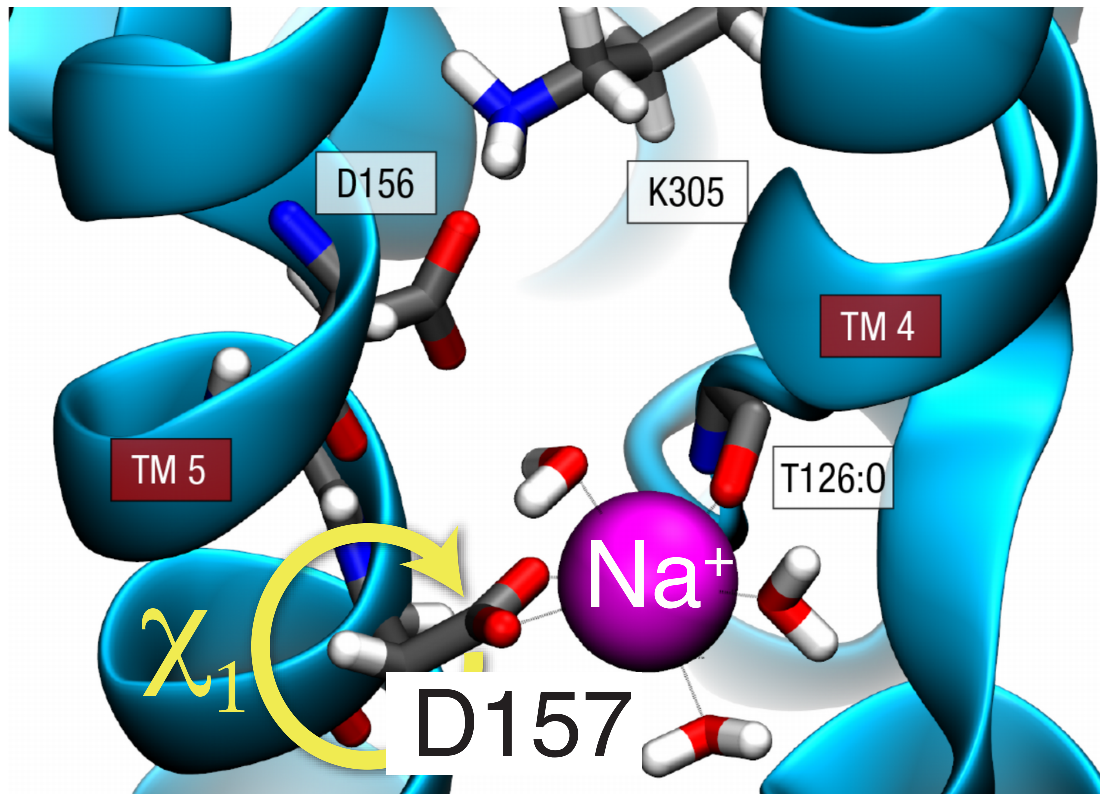


🟥 χ–180°

🟦 χ–90°

🟪 all

- Very slow convergence of Coulomb
- Slow degrees of freedom (e.g., D157 χ_1 dihedral)



Challenge

- data analysis is cumbersome and slow(ish), even with dask on a 6 core workstation (hours)
- on-demand analysis with all current data/while new data is coming in?
- run on HPC system (e.g. XSEDE PSC Bridges)?