

Refactoring plan

Refactoring plan

Objective

Want to restructure HDX model code for publication. Need to make it be modular (i.e. reuse models and functions) and readable.

Utility code

- lmfit fitting Model for time-course
- lmfit fitting Model for 2-state melt
- lmfit fitting Model for aggregate fit of kadd
- Arrhenius Linear model for kdeg independent
- global fit to get kdeg

Main nerd

Nucleotide energetics from reactivity data Tools to extract nucleotide energy from chemical probing experiments.

nerd import

1. Sample import - from csv, import into sqlite3 db. Includes sequence info, nucleotide info, etc. input: sample sheet output: sqlite3 db
2. Import previously ran shapemapper folder input: shapemapper directory output: sqlite3 db (add directory)
3. Sample import (NMR) input: sample sheet output: sqlite3 db (nmr_reactions)

nerd fmod_calc

1. Run shapemapper input: sample, shapemapper config output: sqlite3 db (with new entry)
2. Run spats input: XX output: XX
3. Run other tools fmod_calc tools

nerd degradation

1. NMR deg measurements exponential timecourse input: NMR peak integrals (csv) output: exponential fit
2. Global time-course fits for probing data input: timecourses (rg_id, specify sites) output: global kdeg fit
3. Store to db (kinetic params) - need to do if want to do arrhenius

nerd adduction

1. Independent kadd measurements input: NMR peak integrals (csv) output: ODE fit (add values to table)
2. Aggregate kadd from DMS input: kobs values at melted temps output: aggregated kobs values at melted temps (raw data points for arrhenius, add to table)
3. Store to db (kinetic params) - need to do if want to do arrhenius

nerd arrhenius

1. Linear fit input: array of k values, corresponding temperatures output: LinearModel or params with errors
2. Store to db (Arrhenius)

nerd timecourse

1. Independent time-course fits input: reaction group id output: non-linear fit or params
2. Refit with constrained deg input: reaction group id output: non-linear fit or params
3. Store to db (kinetic params, kobs)

nerd calc_energy

1. 2-state melting fit - K linear curve input: array of kobs values from probe_kinetic_rates output: 2-state melt params
2. Single K calc input: kobs, kadd output: K value

New sqlite tables

sqlite table: probe_kinetic_rates

- Columns:
 - nmr_reaction_id
 - k value
 - k error
 - species (DMS, C8, etc.)

sqlite table: nmr_reactions

- Columns:
 - reaction type (deg, add)
 - NMR machine (A600, A400, etc.)
 - temperature
 - scans
 - read time
 - probe
 - probe_conc
 - probe_solvent
 - substrate
 - substrate_conc
 - buffer
 - csv directory

sqlite table: arrhenius_fits

- Columns:
 - Reaction type (degradation, addition, kobs, K)
 - Data source (nmr, probe_global, probe_free)
 - slope
 - slope_err
 - intercept
 - intercept_err
 - rsq
 - dir to LinearModel (contains data)

Example Usage

1. Import NMR deg samples with `nerd import`
2. Run exponential `nerd degradation`
3. Run deg arrhenius `nerd arrhenius`
4. Import DMS samples `nerd import`
5. Run time-course free fits `nerd timecourse`
6. Run global fits for deg `nerd degradation`
7. Run time-course refit with deg `nerd timecourse`

8. Run store kobs values **nerd timecourse store**
9. Run import NMR add samples with **nerd import**
10. Run ODE **nerd adduction**
11. Run add arrhenius **nerd arrhenius**
12. Run aggregate arrhenius on data **nerd arrhenius**
13. Run 2 state melt
14. (optional) Run K calc with given kadd value - HIV or P4P6