penm development log

30 April 2020

Eliminate need to use `ideal` in enm and lfenm calculations

`ideal` is the ideal protein structure, needed to calculate `v_stress`, `dv_activation`, and `g_entropy_activation`. To eliminate it, I can just avoid calling these functions to set up and mutate `wt`

1. Moved `v_stress`, `dv_activation`, and `g_entropy_activation` to *activation.R*
2. Split old enm_energy into two functions: enm_energy_activation (it calls dv_activation and g_entropy_activation) and enm_energy that calculates just v_min and g_entropy

Eliminate active-site dependent info from enm and lfenm calculations

pdb_active_site indices are used to calculate cmat_activation, kmat_activation, and activation energies. I do not need this in the prot object. If needed, I can add it later.

1. I moved everything related to active site into file activation.R (need to test it, current tests don't use it).
2. Everything else is independent of either "pdb_site_active" or "ideal". Therefore, it should run for tasks that do not depend on defining active sites.
3. Modified tests accordingly (eliminating dummy active site and ideal).
4. Commited changes to git and github

Make a one-button setup of prot object and test

right now setting up the protein object is done by first reading a pdb file into a bio3d pdb object, then calling prot_sc or prot_ca to initialize the prot object, then calling init_prot to cmplete it. Join prot_sc/prot_ca with init_prot into a single function `set_prot(pdb)` Consider adding the enm parameters to prot to be used when passing to other functions(prot)

- Replaced previous "pdb_sc" followed by "init_prot" by a single set_prot(pdb,...) function that sets up the enm, performs nma etc.
- Added enm_param to prot object
- Set up a prot_test.R test in tests/testthat directory, which is run when devtools::test()
- Added objects pdb_2acy_A and prot_2acy_A to data folder for testing Tested and commited all changes.

1 May 2020

Refactor enm

beta and energy terms

beta = 1/(k_boltzman T), depends on temperature. Therefore it is not a property of the protein but, rather, something that depends on the protein's properties and the environment's temperature. For this reason, it would be better to calculate energies when needed, rather than attaching energy terms to the prot object.

Thus, eliminate energy from protein object (and change everything accordingly)

- deleted enm_energy and enm_energy_activation functions
- deleted all calls to enm_energy and recursively...
- deleted query functions get_v_min, get_g_entropy, get_v_min_activaiton, get_g_entropy_activation, get_v_stress
- tested
- commited to git and github

make plot_enm functions good enough to move to package

- added some more plots to test_enm.Rmd
- fixed an issue: graph setting (in set_enm_xyz) missed some i-(i+1) contacts for which dij > d_max (not a problem for CA models, but it's a problem for SC models).
- Changed kij_anm and kij_ming_wall so that they don't set these i-(i+1) kij to 0.
- Tested and commited.

2 May 2020

Deleted `add_site_indexes()`

- added `nsites` and `site` to the result returned by `prot_sc()` and `prot_ca()`
- tested and commited

Deleted `add_enm()`

- renamed `enm_set_xyz` to `enm_from_xyz`
- made `enm_from_xyz` call `nma(kmat)` and return also `mode, evalue, umat, cmat`
- made set_prot call `enm_from_xyz` directly, rather than `add_enm`
- deleted `add_enm`
- tested and commited

Moved non-binary data to `./data_raw`

3 May 2020

merged set_prot.R and add_prot.R into single file

renamed various files in package R directory

4 May 2020

Changed order of `eval` and `umat` columns in `enm_nma()`

Changed all enm plotting functions

Added several functions to *enm_analysis.R*

Changed *test_enm.R*

Now it calculates prot then calls plot functions in the order they are in the *plot_enm.R* file.

5 May 2020

Learned useful tools

- Learnt to use `knitr::purl()` to translate .Rmd into .R
- Learnt to use `mvbutils::foodweb()` to plot dependencies of functions

Refactored prot getters, calculators, and plotters

- Changed function name: `cmat(prot)` to `get_reduced_cmat(prot)`
- Changed function name: `kmat(prot)` to `get_reduced_cmat(prot)`
- Changed function name: `umat2(prot)` to `get_umat2(prot)`
- Changed function name: `umat2_matrix(prot)` to `get_umat2_matrix(prot)`
- Changed definition of all matrix plot functions so that they do not need to transform to tibble before plotting (they call `plot_matrix` instead)
- Deleted `msf_site_mode`
- Renamed `msf_site_mode_matrix` to `get_msf_site_mode`

- Deleted `get_umat2`
- Renamed `get_umat2_matrix` to `get_umat2`
- Renamed `rho_matrix`to `get_rho_matrix` and `plot_rho` to `plot_rho_matrix`
- Moved general functions `matrix_to_tibble`and `plot_matrix` to package `jefuns`
- Moved all plot functions of enm into package directory as `enm_plot.R`
- Tested and comitted to git and github

Other

- Eliminated `d_max` from `get_cn` and related

6 May 2020

Restructured prot object

- Added enm parameters to `prot$enm`
- Wrote getters for prot object
- Changed `set_prot` to a single `set_enm(pdb, ...)` function that sets up the prot object
- Changed structure of `prot` object (and it's class is `prot`)
- Fixed getters according to new structure of prot
- Changed queries by getters in enm module (but not in penm.R and activation.R): e.g. replaced prot*enm*umat by get_umat(prot)

7 May 2020

Rename functions called by `set_enm`

- Extract function `set_enm_param`
- `set_nodes` to `set_enm_nodes`
- `enm_graph_xyz` to `set_enm_graph`
- `eij_edge` to `set_enm_eij`
- `kmat_graph` to `set_enm_kmat`
- `enm_nma` to `set_enm_nma`

Fix calls to prot object in penm.R and related

The prot object of the enm module was restructured. Therefore, I need to change everywhere where prot objects are called in penm.R and related.

- Did all the necessary renaming in penm.R and penm_analysis.R
- Checked that get_penm_mutant works
- Wrote automatic tests test_enm.R test_penm.R for further refactoring
- Tested automatically set_enm() and get_mutant_site(): they work.
- Commited to git and github

WARNING: v_min changes between update_enm = F and update_enm = T

Refactor penm.R

Get enm parameters from prot object rather than pass it independently in all penm.R funcitons

- Eliminate model, d_max, and frustrated from arguments of get_mutant_site and functions called from there

8 May 2020

Refactor set_enm

I refactored set_enm by adding functions that depend as much as possible only on prot objects, so that parameters are passed through prot.

- new set_enm_ functions depend mostly on `prot` rather than explicitly on its components
- added `set_enm_nma()` to file *enm.R* and deleted *enm_nma.R*
- old `set_enm_` are now `calculate_enm_`

Refactor `get_mutant_site()` in *penm.R*

- replaced all `calculate_enm_` functions by `set_enm_(prot)` functions in *penm.R*
- Changed `get_force`
- Changed dlij from lij(mut) - dij(wt) to lij(mut) - lij(wt)
- Made `get_mutant_site` a bit shorter by adding get_dlij
- Removed `wt0` (I wasn't using it, just confusing).
- Tidied up penm.R file a little bit more

9 May 2020

Put update_enm on stand by

- Changed `update_enm` to `mut_model`, that can now be `lfenm` (K doesn't change) or `sclfenm` (the update_enm = T version previous).
- Wrote a note regarding my worries about the sclfenm version
- Separated more clearly the options "lfenm" and "sclfenm" in *penm.R* functions
- Made current "sclfenm" option stop if called because I need to revise it.
- Made "sclefnm" tests skip the test.
- Tested
- Commited

12 May 2020

- Run and revisions of test_mutate_structure.Rmd
- created new perturbation_response_scanning.Rmd that produces slides
- commited everything

Separated data creation and analysis of perturbation response scanning

- Put all creation in "Rmd/prs_data_create.Rmd"
- Put all analysis in "Rmd/prs_data_analyse.Rmd"
- Kept in these files only "prs", not pair comparison mut vs. wt
- Created s pair-comparison file "mut_vs_wt.Rmd", to be completed later

13 May 2020

Complete penm_data_analyse.Rmd

- Created some new notes on PRS
- Completed first full version of penm_data_analyse.Rmd
- Deleted TODO.Rmd (Using Trello and Things... already too much)
- Tested and commited

Clean up

- moved `prs` functions into package: file *R/prs.R*

- created *./to_do* directory to hold planned features
- moved relevant files related to planned features to */to_do*
- run all Rmd reports
- run all tests
- Commit

14 May 2020

Add energy response to prs module

- Added theory doc to *docs*, and notes on energy differences
- Refactored *prs.R*
- Added calculation of data.frame dfej in *prs.R*
- Added calculation of dfej to *prs_create_data.Rmd*
- Added analysis of dfej to *prs_analyse_data.Rmd*
- Tested
- Commited

Add globality of respnse and influence to prs_analyse_data.Rmd

- Added a few slides analysing response and influence globaility in site and mode spaces.
- Test
- Commit

Add v_stress to prs

- Developed some more theory, regarding energies and separating them into site contributions
- Added `enm_v_stress()` and `enm_delta_v_stress` to `penm` module
- Calculated delta_v_stress in *prs.R* (and removed `delta_u` and `delta_a`)
- Recalculated response data using *prs_create_data.Rmd*
- Revised and rerun *prs_analyse_data.Rmd*
- Tested
- Commited

18 May 2020

Superfast calculation of response matrices

- Wrote the key formulae in *docs/notes*
- Wrote *prs_fast.R* that contains functions for fast calcualation of site responses: dr2ij, de2ij, and df2ij
- Wrote *prs_fast.Rmd* that compares cpu times and results between "fast" and "slow" methods
- Tested
- Commited

19 May 2020

Better cpu-time comparison of prs_fast vs. prs

- Improved prs by making the de2_site require kmat_sqrt as input that is calculated outside in the calling function only once for the whole scan.
- Made a more relevant cpu-time comparison
- Verified that simulated prs responses converge to analytical values as number of mutations per site increases
- Tested
- Commited

Added 'fast_delta_structure_mode() to *prs_fast.R*

- Added fast_delta_structure_mode and needed functions called by it.
- Made `delta_structure_mode()` in *prs.R* faster
- Tested cpu-time and convergence in *prs_superfast_mode.Rmd*
- Commited

20 May 2020

Added super-fast energy response

- Developed theory (in *docs/notes*) to calculate dv_min and dv_stress
- Added fucntions to *prs_fast.R*
- Optimized a bit `enm_v_stress()` of *penm_analysis.R*
- Added *prs_superfast_energy.Rmd*
- Tested
- Commited

26 May 2020

Major refactoring of prs.R and prs_fast.R

- Changed everything to make the similarities most obvious
- Used dvm = dvs - de2 by definition so that prs and prs_fast are consistent
- Used similar names (e.g. calculate_dr2ij.fast and calculate_dr2ij.prs, etc.)
- Tested that everything works in prs_superfast.Rmd
- Commited

27 May 2020

Finished version 1 of prs() and prs.fast()

Moving to project superfast_prs

- Moved all prs_*.Rmd files to superfast project, on which I'll work this and next week.