Requirements for w_steadystate_reweight.py

Alex DeGrave

November 9, 2015

- Overall scheme:
  - Parse user inputs
    - Parse command line flags
      - get location of YAML file specifying input locations
      - find which iterations to include in the analysis (first and last iterations)
      - find whether to calculate a single value or an evolution of values
      - get the averaging scheme to use, if any (blocked averaging, cumulative averaging)
    - Parse YAML file
      for each simulation, load the:
      1) file path to the w_postanalysis_matrix output
      2) file path to the w_assign output
      3) bins (or states) that were subject to recycling during the course of the simulation.
        - Load keys as "bins", or "states", which should give lists as values
        - Ultimately, we need a list of bin indices. If the user specifies state indices, we need to get the corresponding bin indices from the w_assign output files.
          >>> where in the w_assign output is this located? 'state_map'?
  - For each input simulation, load flux matrix.
    - "flux matrix": A square, nonnegative, N*N matrix F where each element represents probability flow between two "bins". In this context, each element $F_{i,j}$ (where i is the column index, and j is the row index) represents flux calculated from bin i, into bin j.
      >>>DOUBLE CHECK THIS

    - Load flux matrix from output of w_postanalysis_matrix.py
      - Flux matrices are stored for each iteration in
        flux_matrix_h5['iterations/iter_%08d/']
      in a sparse matrix format. Output files from w_postanalysis_matrix store data in a manner based on the coordinate matrix format from the Scipy library. Each iteration group contains four data sets:
      1) cols: a data set storing a vector of column indices
      2) rows: a data set storing a vector of row indices
      3) flux: a data set storing a vector of flux values
      4) obs: a data set storing a vector of transition counts
      These data sets may be combined to form either the flux matrix (described above), or a "count matrix"
      - Count matrix: A square, integer-valued, nonnegative, N*N matrix C where each element represents the COUNT of transition events between two bins. In this context, each element $C_{i,j}$ represents the number of transitions oberved from bin i, into bin j, during a given iteration.
      For example, given some integer index k, with $0 \leq k < len(cols)$, we have:
          flux_matrix[rows[k], cols[k]] = flux[k]
          count_matrix[rows[k], cols[k]] = obs[k]

- For each input, build a transition matrix
  - "transition matrix": For this purpose, a transition matrix is a square, nonnegative, N*N matrix T where each row sums to one. More formally, it is a RIGHT STOCHASTIC matrix. Here, each element $T_{i,j}$ represents the probability that a trajectory walker in bin i will next transition to bin j during the lag time τ.
    >>>WHAT IS THE DEFAULT LAG TIME?

  - In the case of a simulation without recycling conditions, a transition matrix may be constructed from a flux matrix by row-normalizing. Such a transition matrix would depend only upon the events observed within a single iteration. The user may desire that multiple flux matrices are represented in a single transition matrix. Such average should tend to stabilize the estimate of the transition probabilites, as observations from a single iteration may be insufficient to obtain reasonable statistics. One may imagine multiple methods for averaging.
    1) Average the flux matrices. Given a collection of $N * N$ flux matrices
    $\{F^{(1)},\ F^{(2)},\ ...,\ F^{(n)}\}$
    the following is an estimator for the flux per τ:
    >>>> WHAT TAU DOES w_postanalysis_matrix OUTPUT?

$$< F >= \frac{\sum_{k=1}^{n} F^{(k)}}{n}$$

    NOTE: It probably make more sense to restrict this sum for each row to only those simulations for which observations were made, ie:

$$< F_{i,j} >= \frac{\sum_{k \in D_{i,j}} F_{i,j}^{(k)}}{D_{i,j}^{\#}}$$

    In the most direct sense, a flux matrix is calculated by summing probability flow between some time t and some discrete lag-time later, $t + \tau$. Given the $N * 1$ vector $P^{(k)}$ where each $P(k)_i$ represents the probability in bin i of simulation k at the first timestep t in building the flux matrix, the transition matrix may then be esimated by
    $< T_{i,j} > = \ \frac{<F>_{i,j}}{P_i}$

$$< T_{i,j} >= \frac{< F_{i,j} >}{\sum_{k=1}^{n} P_i^{(k)}}$$

    >>> If w_postanalysis_matrix calculates the flux on a sub-WE-iteration basis, does it also store probability on a sub-WE-iteration basis?
    2) Average the transition matrices. Given a collection of $N * N$ flux matrices
    $\{F^{(1)},\ F^{(2)},\ ...,\ F^{(n)}\}$
    we first normalize each row F_i by the total probability in bin i at the first timestep t in building the flux matrix, we first row-normalize each flux matrix as
    $T_{i,j}^{(k)} \ = \ \frac{F_{i,j}^{(k)}}{P_i}$

$$T_{i,j}^{(k)} = \begin{cases} \frac{F_{i,j}^{(k)}}{P_i} & P_i \neq 0 \\ NAN & P_i = 0 \end{cases}$$

for k=1,2,...,n. We may then obtain an estimate of the transition matrix $< T >$ as:

$$< T_{i,j} >= \frac{\sum_{k \in D_{i,j}} T^{(k)}}{D^{\#}_{i,j}}$$

where $D_{i,j} = \{k \in \{1,2,\ldots,n\} : T^{(k)}_{i,j} \neq NAN\}$, and $D_{i,j}$ represents the number of elements in D. Intuitively, we include

Either formulation may be valid. Method (1) places more "emphasis" on high-weight transitions, in that large flux values contribute more to the estimate <T> than do low-weight transitions. Method (2) places equal emphasis on each transition observation, regardless of the associated weight.

- In the case of a simulation with recycling conditions, construction of a transition matrix is not so straightforward. Consider a set of n simulations indxed by $k = 1, 2, \ldots, n$. Let $F^{(k)}$ be a raw, $N * N$ flux matrix calculated by observing probability flux between bins during the period betweenn time t and time $t + \tau$. We may first obtain a transition matrix for each simulation $k = 1, 2, \ldots, n$ as follows:

  $$T^{(k)}_{i,j} = \frac{F^{(k)}_{i,j}}{P_i}$$

  Let $B^{(k)} \subset \mathbb{N}$, $|B| < N$ be a collection of bin indices corresponding to those bins from which trajectory walkers are recycled during the course of the simulation. Because trajectory walkers are continually removed from bins $i \in B^{(k)}$, it is unlikely to achieve sufficient sampling of transition events out of bins $i \in B^{(k)}$ to reasonably estimate any transition rate $T^{(k)}_{i,j}$ where $i \in B^{(k)}$, $j = 1, 2, \ldots, N$. Thus, such transition rates are not allowed to contribute to the estimate.
  An estimate for the transition matrix is then:

  $$< T_{i,j} > = \frac{\sum_{k \in I_i} T^{(k)}_{i,j}}{I^{\#}_i}$$

  where:

  $$I_i := \{k \in \{1, \ldots, n\} : B^{(k)} \ni i\}$$

  and $I^{\#}_i$ denotes the number of elements in I_i.
- Apply averaging. Averaging could, in theory, take place at the transition matrix level (i.e., find the mean of all transition matrices), or could occur at the level of populations (average the stationary distributions calculated from a sequence of transition matrices; more on this below), or for rates. Since this script will include transition matrices in its output, I think it makes sense to average at the transition matrix level. This is also consistent with w_postanalysis_reweight.