Requirements for w_steadystate_reweight.py

Alex DeGrave

November 5, 2015

- Overall scheme:
  - 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 (one per input).
    - "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?

>>>>>>>>>>>>>>>>>>>>>>>>>> EDIT BELOW. THIS IS WRONG. <<<<<<<<<<<<<<<<<<<<<<<<<<

- 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 count matrices. Given a collection of N*N count matrices

$$\{C^1,\ C^2,\ ...,\ C^n\}$$

the following is an estimator for the flux per iteration:

$$<C> = \frac{\Sigma_{k=1}^{n}\ C^k}{n}$$

The transition matrix may then be esimated by

$$T_{i,j} \ = \ \frac{<C>_{i,j}}{\Sigma_{m=1}^{N}\ <C>_{i,m}}$$

2) Average the transition matrices. Given a collection of N*N count matrices

$$\{C1,\ C2,\ ...,\ Cn\}$$

we first row-normalize each count matrix as

$$T_{i,j}^{k} \ = \ \frac{C_{i,j}^{k}}{\Sigma_{m=1}^{N}\ C_{i,m}^{k}}$$

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

$$<T> = \ \Sigma_{k=1}^{n}\ T^k$$

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.