* Goal of (bio)CASP: infer from an ensemble of "pathway-level" reaction data the elements of reactant structure that are necessary & sufficient for the reaction mechanism(s)
  + Encode these in a form that can be applied in silico (e.g., operators)
* Our groups’ approach so far is a hybrid physical + data-driven approach
* Key considerations
  + Is our reaction featurization (or / and similarity metric) sensitive to structural elements important for mechanisms?
  + Are we catching important elements of varying (relative) size, including those that are small relative to the overall structure?
  + Are all observed structural elements equally informative?
    - (I think no, and yet we elevate all to the same level of important by generating operators for every size of cluster, e.g.)
  + Did we coarse-grain too much (too little)?
  + BIGGIE: how can we evaluate our method, including especially false positive rate?
* Challenges particular to our current intermediate operators create a catch-22
  + We don’t know: Do our similarity metrics & coarse-graining criteria infer a good approximation of mechanistically important structures?
  + We cannot ask the above question because we cannot interpret the operators’ proposed mechanistically important structures (we can’t read the rules)
* To illustrate the above considerations and challenges, take a look at this example, a real situation from an expansion on the BOTTLE project
  + RCMCS appears sensitive to dicarboxylic acid pattern, not sensitive to which functional groups present near reaction center
    - Figure: decarboxylation case study
  + Hard to understand the proposed mechanism of the rules
    - Figure: colored rule text
* Note how RCMCS considers these 5 reactants to be equivalent and is sensitive most to their C4 dicarboxylic backbone which is not important for the mechanism
  + Figure: 5x5 with RCMCS values
  + Note: there may be a way to fix up RCMCS by customizing atom and bond compare classes
* Now that we’ve discussed problems with the existing method and considerations for developing new methods, let’s look at a few new methods
  + First: assigning equivalence as matching structures within a radius around the reaction center of R is a more physical, less data-driven approach
    - Pros
      * Simple
      * Interpretable
      * Practical
    - Cons
      * Coarse
      * Which R?
      * Doesn’t really give you similarity
      * Assumes reaction center hypothesis is very right
      * Published already multiple times (but good for internal use)
      * Can lose sense of how clusters are related, which share elements of higher / lower likelihood
    - To demonstrate what this looks like, lets look at the unique peri-rc structures for several values of R
      * Figure: substructures, number of reactions in that category for R=1, 2, 3
    - Number of operators produced is maybe fine?
      * Figure: # unique substructure vs R
  + Second: hashing method (ecfp) with location-sensitive features (location relative to the reaction center)
    - Pros
      * Can capture structures of multiple scales
      * Can get sense of “likelihood” of structures
    - Cons
      * Bit collisions (maybe fixable)
      * A bit hard to interpret
      * Do we even care that some substructures occur more often?
      * How fix redundancy between structures of different scales?
    - Review the idea
      * Figure: typical ecfp with DAI
      * Figure: ecfp with constant features
    - You get an n x d feature matrix with bits indicating unique structures that you can recover
    - So you can, just like with rcmcs, compute similarities, cluster, create operators
    - You can also look at what structures are probable, informative, etc. but I haven’t managed to squeeze anything useful out of this
* To get a sense for the differences between these three methods: RCMCS (as is), radius-around-reaction-center, location-sensitive-ecfp, look at what each of them fetch as similar reactions to the 5 we looked at from the start
  + Figures: either ranked lists for each of the 5 or a 5x5…