Function flow

See “overview of functions” spreadsheet for additional details on each function’s input and output, data types, etc

1. Call process\_input (this function was in the zipped UI folder I sent you)
   1. Input: all the design and analysis UI inputs from shiny
   2. Output: a list with three data frames named designs, matches, and analyses
2. Call function study
   1. Input: the simulated population (pop), a single row of the processed input “designs” data frame as a list (design), and a single row of the processed input “matches” data frame as a list (match\_input)
   2. This function should be called as many times as necessary such that each row of designs is run with each row of matches (so if there are 3 rows in the designs data frame and 3 rows in the matches data frame, this function should be run 9 times with each possible combination)
   3. This function calls the sub-functions draw\_sample and match
   4. Output: a list containing: the sampled data that has not been matched even if matching was called (this is for use with bias-corrected matching where matching is performed in the analysis stage)(sample), the sampled/pre-processed data ready for analysis (an\_data), a numeric vector of subclass specifications if the data was subclassified or a vector of NA’s if it was not (subclass), a numeric vector of matching weights (will be a vector of 1’s if no weighting needed) (match\_weights), a numeric vector of sampling weights (will be a vector of 1’s if no weighting needed) (samp\_weights), a single row of matches aka the match input fed into the function (as a list) (match\_input)
3. For each resulting set of output from the study function, do the following (studies loop):
   1. Call function g\_models
      1. Input from the study function: the sampled/pre-processed data ready for analysis from the study function (an\_data), subclass (numeric vector, same obs as an\_data), numeric vectors of match\_weights and samp\_weights
      2. Other input: the full analyses data frame of all analysis specifications
      3. This function calls the sub-function est\_g
      4. Output: g (data frame of all formulated estimates of the probability of tx-- n rows per method)
   2. Call function q\_models
      1. Input from the study function: sample (this is the unmatched sample even if matching was conducted-- will use this for bias-corrected matching analysis in which matching happens in the analysis phase), the sampled/pre-processed data ready for analysis from the study function (an\_data), a numeric vector of subclass specifications if the data was subclassified or a vector of NA’s if it was not (subclass), numeric vectors of match\_weights and samp\_weights
      2. Other input: the full analyses data frame of all analysis specifications
      3. This function calls the sub-function est\_q
      4. Output: q (data fame of all formulated estimates of the outcome model-- n\*3 rows per method-- 1st n obs are predictions for observed data, 2nd n obs are predictions when A\_1=1 for everyone and 3rd n obs are predictions when A\_1=0 for everyone)
   3. For each row of the “analyses” data frame (analyses loop):
      1. Run the function analyze
      2. Input from the study function: sample (this is the unmatched sample even if matching was conducted-- will use this for bias-corrected matching analysis in which matching happens in the analysis phase) , the sampled/pre-processed data ready for analysis from the study function (an\_data), a numeric vector of subclass specifications if the data was subclassified or a vector of NA’s if it was not (subclass), the numeric vectors match\_weights and samp\_weights (these will be vectors of 1’s if no matching is needed), a single row of matches aka the match input fed into the function (as a list) (match\_input)
      3. Input from q\_models and q\_models: g, q
      4. Other input: a single row of the analyses data frame as a list
      5. This function calls the sub-functions analyze\_sub, an\_unadj, an\_gcomp, an\_pweight, an\_tmle, and return\_est as needed
      6. Output: est (the ATE or ATT, a single numeric scalar)
4. Store all estimands