The message begins here:

Before we start, I want to mention that, we would like to do the different version of the calculations for

1. before and after price regulation

**input:** data, high or low price tie-breaker choice, pre/after flag, and the **ranking option choice (see part 4)**

1. rolling windows

**input:** data, high or low price tie-breaker choice, window size by number of deals, and step size by number of deals, and the ranking option choice (see part 4)

1. the full sample (which I explain below)

**input:** data, high or low price tie-breaker choice, and the ranking option choice (see part 4)

For output files, try to build in the parameters/model versions we adopt in the file name.

Here goes what we want to do:

**Tasks:**

**Chapter 1: Cosine estimation**

**1. input data:** the unique bidder entity identifier is institution\*bidder, say there are N deals. Prices are normalized already.

**2. calculate participation cosine:**

a) generate a deal participation vector (call it "participation" for example) for each bidding entity:

cell entry = 1 if participated, =0 if not

b) permute through all bidding entities, to calculate the pairwise cosine of the vectors.

output: N\*3 matrix of [bidding entity 1, bidding entity 2, participation cosine calculated in step 2(b)]

**3. calculate the cosine similarity V1: measure based on actual price**

a) prepare the bidding matrix ("Bmatrix") for each bidding entity:

rows of Bmatrix  = number of deals (length of "participation" vector)

cols of Bmatrix = number of bids the bidding entity placed (1-3)

if the bidding entity has only 1 bid, enter price\_normalized in column 1 and the row of the deal.

if the bidding entity has N (N>1) bids, enter price\_normalized in col 1 - N, from high to low

b) prepare the bidding vector for calculation in each bidding entity:

- highest bidding price version: first column of Bmatrix

- lowest bidding price version:  last column of Bmatrix

c) permute through all bidding entities, to calculate the pairwise cosine of the vectors; only when both entities participated in the deal (the cell entry is not zero)

**output**: Cosine\_price = N\*3 matrix of [bidding entity 1, bidding entity 2, highest bid cosine calculated in step 3(c), , lowest bid cosine calculated in step 3(c)]

If it takes long to calculate one loop of cosine, we can set "highest" or "lowest" as a parameter/input, and do two batches of calculation separately.

**4. calculate the cosine similarity V2: measure based on order statistics**

a) assign each bidding price a percentile ranking "price\_order", within all bids on this deal, weighted by the number of bidding share submitted. Or a simple ranking based on bidding price.

Example here:

b) prepare the bidding matrix ("Bmatrix\_order") for each bidding entity:

rows of Bmatrix\_order  = number of deals (length of "participation" vector)

cols of Bmatrix\_order = N

if the bidding entity has only 1 bid, enter "price\_order" in column 1 and the row of the deal.

if the bidding entity N (N>1) bids, enter price\_normalized in col 1 - N, from high to low

c) prepare the bidding vector for calculation in each bidding entity:

- highest bidding price version: first column of Bmatrix\_order

- lowest bidding price version: last column of Bmatrix\_order

d) permute through all bidding entities, to calculate the pairwise cosine of the vectors; only when both entities participated in the deal (the cell entry is not zero).

**output:** Cosine\_order = N\*3 matrix of [bidding entity 1, bidding entity 2, highest bid cosine calculated in step 3(d), lowest bid cosine calculated in step 3(d)]

Again, if it takes long to calculate one loop of cosine, we can set "highest" or "lowest" as a parameter/input, and do two batches of calculation separately.

**Chapter 2: Cluster**

Input: cosine matrices, and the cosine cutoff point

Output: a list of clusters

**4. Construct the collusion index:**

Take Cosine\_price column 3 for example. We want to identify the largest groups existed, that the pairwise cosine within the group is great than a threshold (for example, 0.99).

Output:

a list of the groups

I will leave it to our programming expert about how to identity them as fast as possible.