**Summary**:

The goal is to use hashing scheme to index high dimensional data and to fast retrieve query results. For the hashing scheme we decided to use Locality Sensitive Hashing (LSH), which is an approximated method. We planned to design an indexing scheme to tolerate a certain amount of error rate.

The idea is, when query q changes to q’, we leverage the indexing scheme to avoid re0compute from scratch to arrive at fast retrieval. Due to the nature of our data input, we decided to use **angular distance (angular hashing)**, the implementation of which is based on E2LSH.

+ First apply Onion to get different layers of Convex Hull structure (using cpp qHull library)

+ Then compute Discounted Cumulative Gain(DCG) to penalize the error rate E. Error rate E is related to each convex hull layer.

+ Other than DCG, we give weight based on the number of elements at each convex hull layer.

+ Then we come to optimization equation to get K and L given a fixed hash function budget B (**using Mathematica**)

+ Then use Angular Hashing method to hash data for indexing and queries (**using implementation provided by E2LSH Github repository**)

* Deep dived into the implementation of Angular Hashing method: its Angular Hashing and authors use L2 or L1 during re-ranking (sort distance on the original space of input data) phase to return top results based on the radius
* The implementation would first self-tuned those parameters such as K and L, but we can use our own parameter sets computed from optimization equation
* Output hashed index as well as hashed functions – so that we can reproduce the hash bucket and re-use the hashed index
* Using Angular Distance, we have tested a given query q and when it changes to q’, the result reported makes sense (test case example)

**To-Do**:

+ We have tested results at a given layers, how to aggregate results from different layers to report the actual Top-K results we expect

+ Right now we are only testing using sample data: it is simple to control and to manually check if the results make sense. The next step would be testing using larger scale synthetic data or real-life data (Movielens?)

+ Compare with Hybrid Index method and/or TA algorithm