**Motivation**

Drug Trials
- Every drug seeking FDA approval must go through Phase II and III clinical trial periods (on human participants) to determine safety and effectiveness [1].
- Drugs compared against other drugs and placebos to determine effectiveness.
- Must compare participants with similar features to eliminate bias due to:
  - Age
  - Gender
  - Ethnicity.
- Consider the following example, with Tylenol and Advil compared with a placebo. Each participant is plotted in terms of weight and age.

**Problem Statement**

- Given a set of points, $K_i$ partitioned into $k$ sets of colors, $K_1, K_2, \ldots, K_k$, with $|K_i| = \frac{n}{k}$.
- Define a match $m = \{p_i, p_{j} \in K_i\}$ where $m = k$
- Each color has one point from each color.
- Find the smallest $n$ matches such that each point is only used once.

**Smallest Match Definition**

- Order-independent for up to 3 colors in $d$ dimensions $O(n)$ to compute.
- Equivalent to Traveling Salesman in 2D as number of colors increases.
- Not well defined in higher dimensions.

**Algorithm**

Create Matches
- Creates the kd-trees.
- Calls the addPutativeMatches subroutine for each first color point.
- Possible matches are added to sorted PriorityQueue.
- Matches are pulled in order.

- If a match is invalid, and the first-color point no longer has matches in the queue, re-call addPutativeMatches for it.

**Algorithm Analysis**

**Worst Case**
- Occurs when first $k-1$ colors are coincident with each other and color $k$ points asymptotically converge to a point within the search area of any match.

$$T_{\text{worst case}} = O((k-1)(\log n + \log(k-1) + n) + \log(n))$$

**Expected Case**
- On average, we assume that:
  - $\exists$ such that $\nu$-sized areas, there are $\nu/n$ point in that region.

In other words, the points are evenly distributed and the number of points in any region is proportional to the size of the region.
- Therefore, we consider the number of points in any small region to be constant.

$$T_{\text{expected case}} = O(k \log n + \log n)$$

**Results**

3 colors in 2 dimensions
- kd-tree algorithm outperforms brute force in expected case.

Brute Force: $O(\nu^2 \log \nu)$ with $O(\nu^2)$ space complexity.

Our Algorithm: $O(\nu^2)$ with $O(\nu)$ space complexity.

**) Arbitrary colors and dimensionality**
- kd-tree algorithm outperforms brute force in expected case.

Brute Force: $O(d\nu^2 \log \nu)$ with $O(\nu^2)$ space complexity.

Our Algorithm: $O(kd\nu)$ with $O(\nu)$ space complexity.