ATT Calculation Notes

# Adding Cluster data to g\_data

I altered **Script 15 - Working Data** (made a copy of this script) to output a g\_data with cluster membership included, so that we can group results by cluster

* Ideally need to go back and alter the pipeline properly to include cluster information in the g\_data df

g\_data\_subset\_wide - this doesn't include mergeid

* Wesley was making descriptive stats so I’m guessing didn’t need this column

**Made an assumption that the results data were in the same order as the original df**

Corporatist: 7381 observations (this is after discarding incomplete data)

* initialized the results df by making a unique dataframe of mergeid, cluster, preserving the same order as the training data
* run the whole training + counterfactual pipeline
* Combine the results for each medoid into the results df

# Interpretation of results

What do I expect to see for the ATT relative to ATE?

- If we think conceptually what we're doing here. We're applying the medoid of cluster X and only looking at the treatment effects of cluster X. You would think the obs in cluster X are relatively close to medoid already, at least relative to obs in other clusters

So, I would reasonably expect the ATT to be closer to 0 than the ATE for this cluster

But it's surprising to see in some clusters, that the ATT is larger in magnitude than the ATE

ATT: from outcome perspective within-cluster heterogeneity

# Pipeline function

* Need one more helper function to initialize the results df for each welfare regime
  1. this takes as input the g\_data df after dropping the incomplete data

# Initialize results df with IDs and Cluster

df\_adl\_med\_65\_75\_results = df\_adl\_med\_65\_75.groupby(['mergeid', 'cluster']).size().reset\_index()[['mergeid', 'cluster']]

df\_adl\_cor\_65\_75\_results = df\_adl\_cor\_65\_75.groupby(['mergeid', 'cluster']).size().reset\_index()[['mergeid', 'cluster']]

df\_adl\_scan\_65\_75\_results = df\_adl\_scan\_65\_75.groupby(['mergeid', 'cluster']).size().reset\_index()[['mergeid', 'cluster']]

# Remap cluster name to medoid ID

cluster\_medoid\_map = {

'Cluster 1: FT w Child': 'medoid\_0',

'Cluster 2: FT w/o Child': 'medoid\_1',

'Cluster 3: UE w Child': 'medoid\_2',

'Cluster 4: PT w Child': 'medoid\_3'

}

df\_adl\_med\_65\_75\_results['medoid'] = df\_adl\_med\_65\_75\_results['cluster'].map(cluster\_medoid\_map)

df\_adl\_cor\_65\_75\_results['medoid'] = df\_adl\_cor\_65\_75\_results['cluster'].map(cluster\_medoid\_map)

df\_adl\_scan\_65\_75\_results['medoid'] = df\_adl\_scan\_65\_75\_results['cluster'].map(cluster\_medoid\_map)

* counterfactual\_y\_under\_X\_medoid\_features
  1. Take an additional argument, results\_df (initialized above)
  2. Store observed outcomes + all predicted counterfactual outcomes for each of the 4 medoids in the results\_df
  3. For each medoid:
     1. Groupby medoid (actually we mean cluster here, I used medoid for convenience) and calculate the ATE grouped by cluster
     2. Extract the ATT as the ATE value for the cluster that corresponds to the medoid being applied
        1. E.g. Cluster 1 = medoid\_0, Cluster 2 = medoid\_1, etc
     3. Return a dictionary with ATT values, similar to the other metrics (e.g. raw ATE, std ATE, risk ratio)

See g\_comp\_YD.py for the ***counterfactual\_y\_under\_X\_medoid\_features*** function code

* run\_mc\_bootstrap\_once\_regime\_fixed\_model function needs to also return the ATT value
  1. this is a function to run one iteration of a bootstrap in order to calculate CI
  2. add a few more lines to calculate the bootstrap CI for ATT, similar to the pattern for other metrics
  3. Additionally add a few more lines to parse out the result, following the pattern for other metrics

# Preliminary Result:

1. The ATTS across every cluster are very similar to the ATEs, which shows almost no treatment effect heterogeneities across different cluster groups, in terms of treatment effect size.

