TROP Reference Implementation vs. Paper Analysis

Documenting Methodological Differences

Reference Paper:
Athey, S., Imbens, G. W., Qu, Z., & Viviano, D. (2025).
Triply Robust Panel Estimation with Factor Imputation.
arXiv:2508.21536v2

January 2026

SUMMARY

The reference implementation (TROP_TWFE_average) differs from the paper's methodology in several ways. This document catalogs these differences to clarify the relationship between the paper and implementation.

1. Executive Summary

This document analyzes the reference implementation (TROP_TWFE_average) provided alongside the "Triply Robust Panel Estimation" paper by Athey et al. (2025). The analysis identifies four areas where the implementation differs from the methodology described in the paper.

Note: The function name includes "average" which may indicate this is an intentional simplification or variant of the full algorithm described in the paper.

Aspect	Paper	Implementation	Status
Optimization scope	Control obs only	All observations	Different
tau estimation	Imputation post-fit	Joint parameter	Different
Time distance	dist(s, t) = |t - s|	dist(s) = |s - midpoint|	Different
Unit distance	Pairwise d(j, i)	From treated average	Different
Nuclear norm	Yes	Yes	Same
Exp. weights	Yes	Yes	Same

2. Difference #1: Optimization Scope

The paper and implementation differ in which observations are included in the loss function being minimized.

Paper (Equation 2, Page 7)

Equation 2:

$$(\hat{\alpha}, \hat{\beta}, \hat{L}) = \underset{\alpha, \beta, L}{\operatorname{argmin}} \sum_{j=1}^{N} \sum_{s=1}^{T} \theta_s^{i,t} \omega_j^{i,t} (1 - W_{js}) [Y_{js} - \alpha_j - \beta_s - L_{js}]^2 + \lambda_{nn} \|L\|_*$$

The term (1 - W_js) is a key element: it equals 0 for treated observations (W_js = 1) and 1 for control observations (W_js = 0). This restricts the optimization to fit only on CONTROL observations.

Implementation (methods.py, lines 100-103)

Python implementation:

```
if lambda_nn == np.inf:
    objective = cp.sum_squares(cp.multiply(Y-mu-unit_factor-time_factor-L-W*tau,delta))
else:
    objective = cp.sum_squares(cp.multiply(Y-mu-unit_factor-time_factor-L-W*tau,delta)) \
                + lambda_nn*cp.norm(L, "nuc")
```

The implementation does not include the (1-W) factor. The objective function sums over ALL observations weighted by delta (the distance-based weights).

Comparison

Paper approach:
- Parameters (alpha, beta, L) are estimated using ONLY control observations
- Treatment effect tau is computed separately by imputation

Implementation approach:
- All parameters including tau are jointly optimized over all observations
- The model fits to both treated and control observations simultaneously

3. Difference #2: Treatment Effect Estimation

Related to the first difference, the method for estimating the treatment effect tau also differs between paper and implementation.

Paper (Page 7, after Equation 2)

"...followed by estimating the treatment effect as:"

$$\hat{\tau}_{it} = Y_{it} - \hat{\alpha}_i - \hat{\beta}_t - \hat{L}_{it}$$

The paper describes a two-step procedure:
1. Estimate (alpha, beta, L) by minimizing loss on control observations
2. Compute tau by IMPUTATION: subtract fitted counterfactual from observed outcome

Implementation (methods.py, lines 96, 111)

Variable declaration (line 96):

```
tau = cp.Variable()
```

Return statement (line 111):

```
return tau.value
```

In the implementation, tau is declared as a cvxpy Variable and estimated jointly with all other parameters (mu, unit_effects, time_effects, L). It is NOT computed via imputation after fitting the factor model.

Comparison

The key difference:
- Paper: Two-stage approach (fit factors, then impute treatment effect)
- Implementation: Single-stage joint optimization of all parameters

4. Difference #3: Time Distance Formula

Paper (Equation 3, Page 7)

Equation 3 (time distance):

$$\text{dist}^{\text{time}}(s,t) = |t-s|$$

The paper defines time distance as the absolute difference between period s and the TARGET period t. This means weights adapt to each specific treated period being analyzed.

Implementation (dist_time.R and methods.py line 75)

R implementation (dist_time.R):

```
dist_time <- function(s, T, T_treat) {
  abs(s - (T - T_treat / 2))
}
```

Python implementation (methods.py line 75):

```
dist_time = np.absolute(np.arange(T)-(T-treated_periods/2))
```

The implementation uses a FIXED reference point: (T - T_treat/2), which is the midpoint of the treatment period. This is the same for all treated observations.

Example Comparison (T=20, T_treat=10)

Paper formula for different target periods:
  Target t=11: weights based on |11-s| (minimum at s=11)
  Target t=15: weights based on |15-s| (minimum at s=15)
  Target t=19: weights based on |19-s| (minimum at s=19)

Implementation formula (midpoint = 20 - 10/2 = 15):
  ALL targets: weights based on |s-15| (minimum always at s=15)

The implementation uses the same time weights for all treated periods, while the paper suggests target-specific weights.

5. Difference #4: Unit Distance Formula

Paper (Equation 3, Page 7)

Equation 3 (unit distance):

$$\text{dist}_{-t}^{\text{unit}}(j,i) = \sqrt{\frac{\sum_u 1\{u \neq t\}(1-W_{iu})(1-W_{ju})(Y_{iu}-Y_{ju})^2}{\sum_u 1\{u \neq t\}(1-W_{iu})(1-W_{ju})}}$$

The paper defines unit distance as a PAIRWISE distance between units j and i, computed as the root mean squared difference in outcomes during periods where BOTH units are untreated.

Implementation (dist_unit.R and methods.py lines 78-84)

R implementation (dist_unit.R):

```
Y_treated_avg <- mean(Y[(N - N_treat + 1):N, untreated_time])
squared_diffs <- sum((Y[j, untreated_time] - Y_treated_avg)^2)
sqrt(squared_diffs / (T - T_treat))
```

Python implementation:

```
average_treated = np.mean(Y[treated_units,:],axis=0)

mask = np.ones((N, T))
mask[:,-treated_periods:] = 0
A = np.sum(np.multiply(np.square(average_treated-Y),mask),axis=1)
B = np.sum(mask,axis=1)
dist_unit = np.sqrt(A/B)
```

The implementation computes distance from each unit to the AVERAGE of all treated units, rather than pairwise distances between individual units.

The "average" in TROP_TWFE_average

The function name "TROP_TWFE_average" may refer to this averaging approach:
- Uses average of treated units instead of pairwise comparisons
- This is computationally simpler (O(N) vs O(N^2) comparisons)
- May be an intentional simplification for the reference implementation

6. Implementation Verification

Both Python and R implementations show the same patterns, confirming these are intentional design choices rather than translation errors.

R Implementation (DWCP_TWFE_average.R, lines 44-48)

R objective function:

```
if (lambda_nn == Inf) {
  objective <- sum_squares((Y - mu - unit_factor - time_factor - L - W * tau) * delta)
} else {
  objective <- sum_squares((Y - mu - unit_factor - time_factor - L - W * tau) * delta) +
               lambda_nn * cvxr_norm(L, "nuc")
}
```

The R implementation shows:
- No (1-W) factor in the objective
- tau as a jointly estimated Variable
- Same time and unit distance formulas as Python

7. Summary

This analysis documents four differences between the paper's methodology and the reference implementation. These appear to be intentional design choices in the implementation, possibly representing a simplified or variant algorithm.

Summary of Differences

1. Optimization scope: Implementation optimizes over all observations; paper restricts to control observations via (1-W) term.

2. Treatment effect: Implementation estimates tau jointly; paper uses imputation after fitting factors.

3. Time distance: Implementation uses fixed midpoint reference; paper uses target-specific distances.

4. Unit distance: Implementation uses average of treated units; paper specifies pairwise unit distances.

Interpretation

The "average" suffix in TROP_TWFE_average suggests this may be a simplified variant. The implementation choices (averaging, joint estimation, fixed reference points) are computationally simpler than the paper's specification.