CSDID-Python Parity Report

Aligning the Python ATT(g,t) Pipeline with R/Stata Reference

Difference-in-Differences Implementation Validation

Gabriel Saco

Prior to the fixes documented in this report, the Python package produced estimates that diverged significantly from the R/Stata benchmarks. Most notably, the weighted IPW and DR estimators showed dramatic discrepancies:

Estimator	**Python (Before)**	**R/Stata Reference**
Weighted IPW	≈ 0.18	-3.84
Weighted DR	≈ 0.49	-3.76

After applying all fixes, all six estimator configurations match the R/Stata reference within acceptable numerical precision ($< 10^{-6}$ for point estimates, < 0.12 for bootstrap standard errors).

Issues Identified

Five critical bugs were identified and resolved:

Bug 1. Weight Propagation Failure: User-specified weights were silently ignored in the estimation pipeline.

Bug 2. Panel Data Misrouting: Unbalanced panel data was incorrectly routed to repeated cross-section (RCS) estimators.

Bug 3. Universal Base Period Logic Error: Time index was compared against year values, causing influence function shape mismatches.

Bug 4. Missing Propensity Score Trimming: DRDID-default trimming (`trim_level=0.995`) was not applied to IPW/DR estimators.

Bug 5. Scalar Aggregation Crash: Single-group aggregations failed due to scalar vs. array handling.

1 Ground Truth: R/Stata Reference

1.1 Data Source

The validation uses the **Medicaid expansion mortality dataset** from the JEL Difference-in-Differences reference materials:

Dataset Information

- **File**: `county_mortality_data.csv` (6.0 MB)
- **Source**: Callaway & Sant'Anna (2021) replication materials
- **Coverage**: U.S. county-level mortality rates, 2009–2019
- **Outcome**: Crude mortality rate for ages 20–64 per 100,000

1.2 Sample Selection Criteria

The following filters were applied to the raw data:

1. Exclude states: DC, DE, MA, NY, VT (early adopters or special cases)
2. Keep counties where `yaca == 2014`, `yaca` is missing, or `yaca > 2019`
3. Require complete covariate data for 2013–2014
4. Require complete outcome history for all 11 years (2009–2019)

1.3 Reference Results from R

The following table presents the expected ATT and standard error values from the R implementation:

Table 1: Reference Results: Medicaid Expansion Effect on Mortality (R)

	Unweighted			**Weighted**		
	REG	IPW	DR	REG	IPW	DR
ATT	−1.6154	−0.8586	−1.2256	−3.4592	−3.8417	−3.7561
SE	(4.610)	(4.642)	(4.872)	(2.394)	(3.337)	(3.203)

Note: ATT = Average Treatment Effect on the Treated; SE = Bootstrap Standard Error; REG = Outcome Regression; IPW = Inverse Probability Weighting; DR = Doubly Robust.

2 Diagnosis of Bugs

This section provides a detailed technical diagnosis of each bug discovered in the `csdid-python` codebase.

2.1 Bug 1: Weight Propagation Failure

Bug Identified

Location: `csdid/att_gt.py`, method `ATTgt.__init__`
Symptom: Weighted analyses produced results identical to unweighted analyses, despite user-specified weights.
Root Cause: The constructor passed hardcoded `None` values for `weights_name` and `clustervar` to the `pre_process_did` function, ignoring user-supplied arguments.

Original Code:

```
dp = pre_process_did(
    yname=yname, tname=tname, idname=idname, gname=gname,
    data=data, control_group=control_group, anticipation=anticipation,
    xformla=xformla, panel=panel, allow_unbalanced_panel=allow_unbalanced_panel
    ,
    cband=cband, clustervar=None, weights_name=None  # BUG: Hardcoded None
```

```
)
```

2.2 Bug 2: Panel Data Misrouting

Bug Identified

Location: `csdid/attgt_fnc/preprocess_did.py`
Symptom: Panel data with `allow_unbalanced_panel=True` was processed using repeated cross-section (RCS) estimators instead of the correct panel estimators.
Root Cause: The preprocessing logic incorrectly set `panel=False` whenever `allow_unbalanced_panel=True`. Additionally, the sample size n was computed as the row count instead of the unique unit count.

Original Logic:

```
if panel:
    if allow_unbalanced_panel:
        panel = False  # BUG: Forces RCS path incorrectly
        true_rep_cross_section = False
```

A secondary issue involved truncating the `control_group` parameter when passed as a list:

```
control_group = control_group[0]  # BUG: Fails if already a string
```

2.3 Bug 3: Universal Base Period Logic Error

Bug Identified

Location: `csdid/attgt_fnc/compute_att_gt.py`
Symptom: Universal base period estimation produced incorrect influence function sizes and sporadic `panel2cs2` failures.
Root Cause: The code compared a *time index* (`pret`) against a *year value* (`tn`), and used `len(data)` (row count) for influence function padding instead of n (unique unit count).

Original Code:

```
# BUG: pret is an index, tn is a year value
if base_period == 'universal' and pret == tn:
    add_att_data(att=0, pst=0, inf_f=np.zeros(len(data)))  # BUG: len(data) !=
    n
    continue
```

The same index vs. year confusion affected `ypre`/`ypost` assignment:

```
ypre = disdat.y0 if tn > pret else disdat.y1   # BUG: pret is index, not year
ypost = disdat.y0 if tn < pret else disdat.y1
```

2.4 Bug 4: Missing Propensity Score Trimming

Bug Identified

Location: `csdid/attgt_fnc/compute_att_gt.py`
Symptom: Weighted IPW and DR estimates diverged dramatically from R/Stata benchmarks (Python $\approx 0.18/0.49$ vs. R $\approx -3.84/-3.76$).
Root Cause: The Python implementation called untrimmed versions of `drdid.ipwd_did` and `drdid.drdid`, while the R DRDID package applies a default `trim_level=0.995` to cap extreme propensity scores.

Without trimming, extreme propensity scores near 1.0 in the control group caused IPW weights to explode:

$$w_{\text{control}} = \frac{\hat{p}(X)}{1-\hat{p}(X)} \xrightarrow{\hat{p}\to 1} \infty$$

This inflated weight on a few observations severely biased the weighted estimators.

2.5 Bug 5: Scalar Aggregation Crash

Bug Identified

Location: `csdid/aggte_fnc/utils.py`, function `AGGTEobj`
Symptom: Single-group aggregations crashed with indexing errors.
Root Cause: The significance flag `overall_sig` was treated as an array, but became a scalar for single-group outputs. The expression `overall_sig[np.isnan(...)]` failed on scalars.

3 Patches and Fixes

This section documents the specific code changes applied to resolve each bug.

3.1 Fix 1: Weight Propagation

Fix Applied

Forward user-specified `weights_name` and `clustervar` parameters to the preprocessing function.

File: `csdid/att_gt.py`

```
# BEFORE
dp = pre_process_did(
    ..., clustervar=None, weights_name=None
)

# AFTER
dp = pre_process_did(
    ..., clustervar=clustervar, weights_name=weights_name
)
```

3.2 Fix 2: Panel Handling

Fix Applied

Preserve `panel=True` for unbalanced panels; compute n as the number of unique units; handle both string and list inputs for `control_group`.

File: `csdid/attgt_fnc/preprocess_did.py`

```
# Control group: handle string or list input
if isinstance(control_group, (list, tuple)):
    control_group = control_group[0]

# Panel handling: preserve panel=True, compute unique unit count
if panel:
    if allow_unbalanced_panel:
        try:
            n = data[idname].nunique()
        except Exception:
            n = len(pd.unique(data[idname]))
        # panel remains True (do NOT set to False)
```

3.3 Fix 3: Universal Base Period Logic

Fix Applied

Convert the time index to the actual year value before comparison; use n (unique unit count) for influence function array sizing.

File: `csdid/attgt_fnc/compute_att_gt.py`

```
# Convert index to year value
pret_year = tlist[pret]
post_treat = 1 * (g <= tn)

# Correct comparison using year values
if base_period == 'universal' and pret_year == tn:
    add_att_data(att=0, pst=post_treat, inf_f=np.zeros(n))  # n, not len(data)
    continue

# Correct ypre/ypost assignment using year values
ypre = disdat.y0 if tn > pret_year else disdat.y1
ypost = disdat.y0 if tn < pret_year else disdat.y1
```

3.4 Fix 4: Propensity Score Trimming

Fix Applied

Implement DRDID-compatible propensity score trimming with default `trim_level=0.995`.

New File: `csdid/attgt_fnc/drdid_trim.py`

This module provides trimmed versions of the IPW and DR estimators that match the R DRDID package behavior:

```
def _trim_pscore(ps_fit, D, trim_level):
    """Trim propensity scores following DRDID convention."""
    ps_fit = np.minimum(ps_fit, 1 - 1e-6)  # Cap at 1 - epsilon
```

```
    trim_ps = ps_fit < 1.01  # Initialize all as valid
    controls = D == 0
    trim_ps[controls] = ps_fit[controls] < trim_level  # Trim controls with
    high ps
    return ps_fit, trim_ps
```

The module exports four functions:

- `std_ipw_did_panel`: IPW estimator for panel data with trimming
- `drdid_panel`: Doubly robust estimator for panel data with trimming
- `std_ipw_did_rc`: IPW estimator for repeated cross-sections with trimming
- `drdid_rc`: Doubly robust estimator for repeated cross-sections with trimming

Wiring in compute_att_gt.py:

```
# Import trimmed estimators instead of untrimmed versions
from csdid.attgt_fnc import drdid_trim

# In the estimation block:
if est_method == "ipw":
    est_att_f = drdid_trim.std_ipw_did_panel  # was: ipwd_did.std_ipw_did_panel
elif est_method == "dr":
    est_att_f = drdid_trim.drdid_panel        # was: drdid.drdid_panel
```

3.5 Fix 5: Scalar Aggregation Handling

Fix Applied

Convert scalar significance flags to arrays before indexing operations.

File: `csdid/aggte_fnc/utils.py`

```
# BEFORE
overall_sig[np.isnan(overall_sig)] = False
overall_sig_text = np.where(overall_sig, "*", "")

# AFTER
overall_sig = np.asarray(overall_sig)  # Ensure array type
overall_sig = np.where(np.isnan(overall_sig), False, overall_sig)
overall_sig_text = np.atleast_1d(np.where(overall_sig, "*", ""))
```

4 Validation Results

4.1 Python vs. R Comparison

After applying all fixes, the Python implementation produces results that closely match the R reference:

Table 2: Comparison: Python Results vs. R Reference

	Unweighted			**Weighted**		
	REG	IPW	DR	REG	IPW	DR
Python (After Fixes)						
ATT	−1.6154	−0.8586	−1.2256	−3.4592	−3.8417	−3.7561
SE	(4.69)	(4.65)	(4.98)	(2.40)	(3.41)	(3.25)
R Reference						
ATT	−1.6154	−0.8586	−1.2256	−3.4592	−3.8417	−3.7561
SE	(4.61)	(4.64)	(4.87)	(2.39)	(3.34)	(3.20)
Difference (Python − R)						
ΔATT	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$
ΔSE	0.08	0.01	0.11	0.01	0.07	0.05