pyxtdpthresh user guide

Version 0.0.1 (alpha)

04 November 2026

Contents

1. Before you start **1**

2. Calling `xtdpthresh` **2**

3. Options of `xtdpthresh` **3**
Model . . . 3
Instruments . . . 3
Transformation and threshold search . . . 3
Inference . . . 4
Reporting . . . 5
Combinations the commands refuse . . . 5
Retired options . . . 5
The options in use . . . 5

4. Choosing your settings **6**
Instruments: the choice that matters most . . . 6
First differences or forward orthogonal deviations . . . 7
Jump or kink . . . 7
Threshold search . . . 7
Bootstrap . . . 7
Variance . . . 7
Before you report . . . 8

5. Calling `xthenreg` **8**

6. Reading the output **9**
The printed result . . . 9
Numbers as objects . . . 10
`res.scalars` . . . 10
A reading order that works . . . 11
`diagnose()` and `settings_log()` . . . 11
Two diagnoses, read . . . 11

7. Working with a result **12**

8. Reproducibility, speed and memory **14**

9. Simulation and Monte Carlo **14**

10. Errors and warnings **16**

11. What differs from Stata **16**

A Methodological note **18**
A.1 Notation . . . 18
A.2 Model and moment conditions . . . 18

A.3 Estimation . . . 19
A.4 Cumulative sums: the criterion on a whole grid in one pass . . . 20
A.5 Variances and specification tests . . . 21
A.6 Bootstrap inference . . . 21
A.7 Gong and Seo's Algorithm 1 and what the package does . . . 23
A.8 `xthenreg` . . . 24
A.9 Conventions of the Stata commands that are reproduced . . . 24
A.10 Validation journal . . . 25
A.11 Where every Stata function went . . . 26

B Validation against Stata 17 **30**
B.1 Rule and method . . . 30
B.2 Summary by block . . . 30
B.3 The examples of the help file . . . 31
B.4 The stress test . . . 31
B.5 Largest deviation by object . . . 32
B.6 Largest deviation by case . . . 35
B.7 Comparisons under a looser tolerance . . . 43
B.8 What this report does not cover . . . 43
B.9 Environment . . . 44

This guide explains every argument of the two estimators, how each one corresponds to the Stata option it ports, and how to read what comes back. It assumes you know what a dynamic panel threshold model is; the equations and the algorithms are in the methodological note, Appendix A of the PDF edition of this guide (`user_guide.pdf`, built from `docs/methodological_note.tex`); the comparison with Stata 17, number by number, is Appendix B. The Stata help files (`xtdpthresh.sthlp`, `xthenreg.sthlp`) remain the reference for the commands; this guide says where the Python package differs from them.

The code blocks of this guide run top to bottom as one script (`tests/integration/test_examples.py` runs them), so a block may use the names defined in the blocks before it. A block whose first line is `# not run by the tests` is shown for illustration only.

---

1. Before you start

Data. Both estimators take a pandas `DataFrame` in long form, one row per (unit, period). Name the unit and time columns with `id=` and `time=`, or give the frame a two-level `MultiIndex` (unit, time) and leave both out. `pyx.from_stata("file.dta")` reads a `.dta` file and uses the `xtset` variables Stata recorded in it; a file saved without `xtset` is returned as it is and you pass `id=` and `time=` yourself (the names are never guessed).

`xtdpthresh` accepts unbalanced panels. `xthenreg` needs a strongly balanced panel (every unit observed in every period) and stops with an error otherwise, as the Stata command does.

Names and missing values. Column names may contain accents and other Unicode letters (`"investissé"`, `"dette_é"`), as in Stata 14 and later; a name with a space is refused. A `NaN`, a missing nullable value and an infinity (`±inf`, for example after a `log(0)`) are all missing, as Stata has no infinity; an infinity gives a warning that names the column and the count.

Time-series operators. Variable names in every role may carry Stata's operators as text: `"L.x"`, `"L2.x"`, `"D.x"`, `"S12.x"`, `"L(1/2).x"`. They are computed on the calendar, as in Stata: `L.x` is the value of the *period* before, so a gap in a unit's dates gives a missing value. The dependent variable may not carry an operator. The rules of the help file apply: each variable sits in only one role, a lag of the dependent variable is declared `predetermined`, and differences and leads of the dependent variable are refused.

The example used below. Hansen's (1999) firm investment panel, 565 firms observed from 1973 to 1987, with the five extreme firms removed as in Seo and Shin (2016).

```
import pyxtdpthresh as pyx

df = pyx.load_dataset("invest")
df = df[~df["firm"].isin([137, 391, 407, 488, 538])]

# The model of Seo and Shin (2016, section 7): all four slopes change at a leverage threshold,
```

```
# cash flow, Tobin's q and leverage are endogenous, instruments are lags 3 to 8, collapsed.
MODEL = dict(
    q="debt",
    endogenous=["cashflow", "tobin_q", "debt"],
    maxlag=(3, 8),
    collapse=True,
    id="firm",
    time="year",
)
```

Other ways in. The unit and time columns can be the two levels of the index; `pyx.list_datasets()` names the bundled data (`"invest"` today); `pyx.from_stata` reads a `.dta` file.

```
indexed = df.set_index(["firm", "year"])
no_names = {k: v for k, v in MODEL.items() if k not in ("id", "time")}
same = pyx.xtdpthresh(indexed, "invest", bootstrap=False, **no_names)
print(round(same.gamma, 4))

print(pyx.list_datasets())  # name -> description

# A .dta saved without xtset gives a plain frame: the names are never guessed.
df[["firm", "year", "invest", "debt"]].to_stata("panel.dta", write_index=False)
raw = pyx.from_stata("panel.dta")
print(raw.attrs["id"], raw.attrs["time"])  # None None: pass id= and time= yourself
```

A file saved after `xtset` (or `xtset …, delta()`) gives a frame whose index levels are the panel and time variables Stata recorded, with their names in `attrs["id"]`, `attrs["time"]` and the time unit in `attrs["delta"]`; `id=` and `time=` can then be left out. Stata value labels are not applied.

---

2. Calling `xtdpthresh`

```
res = pyx.xtdpthresh(data, y, x=(), *, q, id=None, time=None, where=None, grid_values=None, **options)
```

Argument	Stata	Meaning
`data`	the data in memory	long `DataFrame`
`y`	`depvar`	dependent variable (a plain column name)
`x`	`indepvars`	strictly exogenous regressors, a name or a list
`q`	`qx(varname)`	threshold variable, **required**
`endogenous`	`endogenous(varlist)`	regressors correlated with the current error
`predetermined`	`predetermined(varlist)`	regressors uncorrelated with current and later errors
`exogenous`	`exogenous(varlist)`	more strictly exogenous regressors, treated like `x`
`iv`	`iv(varlist)`	external instruments, entered with their period-t values
`id`, `time`	`xtset id time`	unit and time columns (or index level names)
`where`	`if`	condition selecting the equations (see below)
`grid_values`	(none)	candidate thresholds to use instead of the grid built from `grid`
`**options`	the other options	Section 3

`endogenous`, `predetermined`, `exogenous` and `iv` are keyword arguments given as a name or a list of names. Anything else the call receives is an option of Section 3; an unknown name raises a `TypeError` (`… got an unexpected keyword argument 'gridd'`).

The model is `y = x'`β `+ (1, x')`δ `·` `1(q >` γ`)` `+` η `+` ε. In the dynamic model (the default) the lag of `y` is added to `x` automatically; `static=True` removes it. The slopes are β below the threshold and β `+` δ_2 above it; δ_1 is the shift of the intercept.

How the roles are instrumented. With `method="fd"`, an endogenous regressor is instrumented by its levels from t-2, a predetermined one from t-1, a strictly exogenous one by its own transformed value; with `method="fod"` the dates are t-1 and t. `maxlag` and `collapse` shape these sets, `iv` adds external ones.

`where` and `in`. `where` is a string evaluated with `DataFrame.eval` (`"year >= 1980"`, `"firm != 12"`) or a boolean array of the length of the frame. Two differences from Stata's `if`: a missing value in the condition counts as *false* (Stata counts it as true; the rows with a missing variable are removed afterwards anyway), and there is no `in`; select rows of the frame instead. By default (`history="panel"`) the condition restricts the *equations*, and observations outside it remain available as instruments.

```
# Point estimates only: the bootstrap is the slow part, bootstrap=False skips it.
fast = pyx.xtdpthresh(df, "invest", bootstrap=False, **MODEL)
print(round(fast.gamma, 4), fast.n_obs, fast.n_units, fast.n_instruments)

# Equations from 1980 on; the earlier observations still serve as instruments.
late = pyx.xtdpthresh(df, "invest", where="year >= 1980", bootstrap=False, **MODEL)
print(late.n_obs)

# Your own candidate thresholds, instead of the grid built from `grid`.
import numpy as np

own = pyx.xtdpthresh(
    df, "invest", grid_values=np.linspace(0.05, 0.55, 11), bootstrap=False, **MODEL
)
print(round(own.gamma, 4))
```

A complete call, with inference, a seed, and the test of the threshold estimated by Hansen (1999), 0.0157 (this is the second example of the Stata help file):

```
res = pyx.xtdpthresh(df, "invest", gamma_test=0.0157, seed=12345, **MODEL)
print(res)
```

External instruments. `iv` takes names (with operators) of variables that are not regressors:

```
df["peer_q"] = df.groupby("year")["tobin_q"].transform("mean")  # yearly mean of Tobin's q
withiv = pyx.xtdpthresh(df, "invest", iv=["L2.peer_q"], bootstrap=False, **MODEL)
print(withiv.n_instruments)
```

A variable that takes the same value for every unit in a period (such as `peer_q`) is identified through the period constants: without `collapse`, its own columns repeat the constants and are dropped, and `res.scalars["N_iv_common"]` counts them.

---

3. Options of `xtdpthresh`

Python names are in `snake_case`; the Stata name is given for each. Defaults are those of the Stata command. Every rule of the Stata command is checked in the same order and raises `SpecificationError` (Section 10) carrying Stata's return code.

Model

Python	Stata	Default	Notes
`kink`	`kink`	`False`	Continuous effect at the threshold instead of a jump. Put `q` in `x`, `endogenous` or `predetermined` so that it has a slope below γ; otherwise a warning says the model is a one-sided hinge.
`static`	`static`	`False`	No automatic lag of `y`. Accepted by the Stata command; the help file of version 0.9.38 does not describe it.
`time_effects`	`td`	`False`	Partial common time effects out of the equations (dummies are not entered by hand).
`history`	`history(panel|sample)`	`"panel"`	With `"sample"`, `where` also bounds the lag history; then no operated term (`L.x`) may be used. Build the lags as columns first.

Instruments

Python	Stata	Default	Notes
`maxlag`	`maxlag(# [#])`	all lags	An integer `L` uses lags 1 to `L` for predetermined variables and 2 to `L` for endogenous variables and `y`; a pair `(a, b)` uses `a` to `b` with the same minimums. `None` means no limit. Without it the number of instruments grows with the square of T.
`collapse`	`collapse`	`False`	One instrument column per lag instead of one per lag and period (Roodman 2009). Fewer instruments, possibly weaker identification of γ.
`iv_collapse`	`iv(…, collapse)`	`False`	Collapse only the external instruments. `collapse=True` implies it.

Transformation and threshold search

Python	Stata	Default	Notes
`method`	`method(fd|fod)`	`"fd"`	First differences, or forward orthogonal deviations. FOD discards fewer equations when the panel has interior gaps (an interior gap removes three FD equations and two FOD equations).
`grid`	`grid(#)`	`100`	Points of the threshold search, at least 10. Uniform grids of 100, 199 and 397 points are nested, so a finer grid checks the sensitivity of $\hat{\gamma}$.
`grid_type`	`gridtype(uniform|quantile)`	`"uniform"`	Equally spaced in `q`, or at its empirical quantiles; used for the estimation grid and the confidence-set grid.
`trim`	`trim(#)`	`0.10`	Total trimming rate of `q`, split between the tails; 0.10 restricts the grid to the 5th–95th percentiles. Between 0.01 and 0.45. (`xthenreg` uses 0.4.)
`grid_sample`	`gridsample(effective|observed)`	`"effective"`	Which values of `q` define the trimming bounds and the regime counts of `min_regime`: those that enter the transformed equations, or those in the retained rows.
`min_regime`	`minregime(#)`	`0`	Raise the minimum number of observations in each regime. The default, and the least allowed, is the size of the smaller trimmed tail.
`refine`	`refine(#)`	`0`	Rounds (0–20) of local refinement after the grid search: the observed values of `q` next to $\hat{\gamma}$ become candidates. The confidence set and tests use the initial grid only. Not allowed with `kink`, whose $\hat{\gamma}$ is refined continuously.
`grid_values`	(none)	`None`	An argument of the call, not an option: an explicit array of candidates.

Inference

Python	Stata	Default	Notes
`bootstrap`	`noboot` (negated)	`True`	`False` skips the confidence set and the linearity test. Point estimates, Hansen and AR tests remain.
`n_boot`	`boot(#)`	`299`	Bootstrap replications, at least 10. With B draws the smallest possible p-value is 1/(B + 1).
`grid_ci`	`gridci(#)`	`100`	Candidate thresholds of the confidence-set grid, at least 10.
`boot_type`	`boottype(wild|unit)`	`"wild"`	Scheme of the confidence set and of `gamma_test`. `"unit"` resamples whole units, follows Algorithm 1 of Gong and Seo (2026), needs `method="fd"` without `kink` or `time_effects`, and is much slower. The linearity test always uses the wild scheme.
`seed`	`rseed(#)`	drawn	Integer in 0 … 2^{31}-1. When the bootstrap is on and `seed` is omitted, a seed is drawn and recorded: `res.seed` replays the run. Separate seeds are derived for the confidence set, the linearity test and `gamma_test`, so changing one leaves the draws of the others unchanged.
`linearity_test`	`notest` (negated)	`True`	`False` skips the bootstrap linearity test.
`gamma_test`	`citest(#)`	`None`	Also run the bootstrap test of H_0: $\gamma = \#$, with a value of `q`. Needs the bootstrap. It is evaluated after the other components with its own seed, so asking for it changes nothing else. The result is in `res.scalars["citest_p"]` and `["citest_accept"]`.
`continuity_test`	`conttest`	`False`	Bootstrap test of continuity (is the kink model enough?), result in `res.scalars["pval_cont"]`. Needs the bootstrap and the linearity test, no `kink`, and `q` as a contemporaneous regressor. Accepted by the command; not described in the 0.9.38 help file.
`vce`	`vce(robust|windmeijer)`	`"robust"`	Cluster-robust two-step sandwich, or a Windmeijer-type correction. The variance includes the estimation error of $\hat{\gamma}$; the one conditional on $\hat{\gamma}$ is `res.fit.variance.cond` (`e(V_cond)`).
`bw_scale`	`bwscale(#)`	`1.5`	Scale of the kernel bandwidth `h = # · 1.06 · s_q · n^(-1/5)` of the jump model's joint variance. Only the reported variance and the AR statistics that include $\hat{\gamma}$ depend on it.
`center`	`nocenter` (negated)	`True`	`False` uses the uncentered moment covariance, as `xtabond2` does.
`level`	`level(#)`	`95`	Confidence level, between 10 and 99.99.

Python	Stata	Default	Notes
`coef_boot`	`coefboot(twostep|onestep|none)`	`"none"`	Legacy coefficient bootstrap that searches the threshold again in each draw. A note says so; the intervals are in `res.fit.bootstrap.coefficient.interval` (`e(b_bootci)`) and the printed tables keep the analytic ones. `"twostep"` is refused with `time_effects`.
`coef_ci_type`	`coefcitype(symmetric|percentile)`	`"symmetric"`	Kind of interval of that bootstrap.

Reporting

Python	Stata	Default	Notes
`warn`	`nowarn` (negated)	`True`	`False` silences nonfatal warnings and most notes. Notes on dropped instrument columns and a missing Hansen J, and the warnings for an incomplete or empty confidence set, are always shown.
`verbose`	`verbose`	`False`	Accepted, **no effect** in Python. The per-candidate table of the confidence set is `res.ci_table()`.
`export_gmm`	`exportgmm`	`False`	Accepted, **no effect** in Python.

Combinations the commands refuse

- `n_boot < 10` or `grid_ci < 10` with the bootstrap on; `grid < 10`.
- `gamma_test` without the bootstrap; `continuity_test` without the bootstrap and the linearity test.
- `refine > 0` with `kink`.
- `boot_type="unit"` with `method="fod"` or with `kink` (checked when the bootstrap is on), and with `time_effects`.
- `coef_boot="twostep"` with `time_effects`.
- A `maxlag` pair with the low end above the high end.
- `history="sample"` with a lagged or differenced term.

Retired options

These existed in earlier versions of the Stata command and are refused with an explanation: `method="system"`, `boot_type="exact"`, `coef_boot="gs"`, and the arguments `search_mode`, `search_max`, `search_tol`, `lev_maxlag` and `iv_maxlag`.

The options in use

Each option below is used once, on the model of Section 1. Point estimates first (`bootstrap=False` is instant), then the options of the bootstrap with a light one (99 draws, 40 candidates).

```
def fit(**changes):
    """Point estimates of the model of Section 1, with some options changed."""
    return pyx.xtdpthresh(df, "invest", **{**MODEL, "bootstrap": False, **changes})


def show(label, r):
    print(
        f"{label:26s} gamma={r.gamma:.4f}  equations={r.n_obs}  instruments={r.n_instruments}"
        f"  Hansen p={r.scalars['hansen_p']:.3f}"
    )


# Model
show("static (lags 2-8)", fit(static=True, maxlag=(2, 8)))  # no automatic L.invest
show("time effects (td)", fit(time_effects=True))
show("kink", fit(kink=True))
show("history='sample', 1976+", fit(history="sample", where="year >= 1976"))

# Instruments
show("maxlag (3, 5)", fit(maxlag=(3, 5)))
show("no collapse, lags 3-5", fit(collapse=False, maxlag=(3, 5)))
show(
    "external, iv_collapse",
    fit(iv=["L2.cashflow"], iv_collapse=True, collapse=False, maxlag=(3, 5)),
```

```
)

# Transformation, search, variance
show("fod", fit(method="fod"))
show("trim 0.20", fit(trim=0.20))
show("quantile grid", fit(grid_type="quantile"))
show("grid_sample='observed'", fit(grid_sample="observed"))
show("min_regime 500", fit(min_regime=500))
show("refine 3", fit(refine=3))
show("windmeijer", fit(vce="windmeijer"))
show("uncentered covariance", fit(center=False))
show("bw_scale 3", fit(bw_scale=3))
show("warn=False", fit(warn=False))

infer = dict(n_boot=99, grid_ci=40, seed=1)  # a light bootstrap, for illustration


def full(**changes):
    return pyx.xtdpthresh(df, "invest", **{**MODEL, **infer, **changes})


print(full().gamma_ci, full(level=90).gamma_ci)  # 95 % and 90 % sets (hulls)
print(full(linearity_test=False).scalars["pval_lin"])  # nan: the test was skipped
print(full(continuity_test=True).scalars["pval_cont"])  # bootstrap test of continuity

# Unit resampling (Algorithm 1 of Gong and Seo); slower, so fewer draws here.
unit = pyx.xtdpthresh(
    df, "invest", **{**MODEL, "n_boot": 49, "grid_ci": 20, "seed": 1, "boot_type": "unit"}
)
print(unit.gamma_ci, unit.scalars["ci_nseg"])

# Legacy coefficient bootstrap: a note is issued; the intervals are kept on the fit.
legacy = full(coef_boot="onestep", coef_ci_type="percentile")
print(legacy.fit.bootstrap.coefficient.interval.shape)  # (2, 9): lower and upper, e(b_bootci)
```

`verbose` and `export_gmm` are accepted and do nothing (Section 11).

---

4. Choosing your settings

No setting is right in general: the data and the question decide. This section says what each choice does according to the help file of the command, and shows on the Hansen panel how far the answer can move. The numbers are one panel and one model. They illustrate sensitivity; they are not rules.

Instruments: the choice that matters most

Without `collapse` and `maxlag`, the number of instruments grows with the square of the number of periods. Many instruments relative to the number of units makes the Hansen test unreliable (Roodman 2009) and, with the default centered covariance, tends to make it over-reject (help file). `collapse` and `maxlag` shrink the set, at the price of weaker identification of γ.

```
import pandas as pd

settings = {
    "collapsed, lags 3-8": {},
    "collapsed, lags 2-10": {"maxlag": (2, 10)},
    "collapsed, lags 3-5": {"maxlag": (3, 5)},
    "not collapsed, lags 3-5": {"collapse": False, "maxlag": (3, 5)},
    "not collapsed, all lags": {"collapse": False, "maxlag": None},
}
rows = {}
for label, changes in settings.items():
    r = fit(**changes)
    rows[label] = {
        "gamma": r.gamma,
        "instruments": r.n_instruments,
        "hansen_p": r.scalars["hansen_p"],
        "ar2_p": r.scalars["ar2_p"],
    }
print(pd.DataFrame(rows).T.round(4))
```

On this panel (560 units) the table reads:

Setting	$\hat{\gamma}$	Instruments	Hansen p	AR(2) p
collapsed, lags 3-8	0.4194	25	0.382	0.738
collapsed, lags 2-10	0.0121	37	0.626	0.013
collapsed, lags 3-5	0.5895	13	0.972	0.476
not collapsed, lags 3-5	0.1216	144	0.000	0.939
not collapsed, all lags	0.1092	377	0.000	0.922

The threshold estimate moves from 0.01 to 0.59 with the instrument set. Two readings are grounded in the help file. The lag-2 window rejects AR(2) (p = 0.013), which "casts doubt on the lag-2 instruments or on the specification" (third example of the help file). The uncollapsed sets have 144 and 377 instruments for 560 units and a Hansen p-value of 0.000, which the help file warns is unreliable in that situation: it is not by itself evidence against the model, but the instability of $\hat{\gamma}$ is a fact to report. `pyx.robustness()` (Section 7) runs this kind of variation for you.

A workable order: start from a collapsed set with a lag window justified by the timing of the variables; check AR(2) and Hansen; move the window and the collapsing; report what moves.

First differences or forward orthogonal deviations

`method="fd"` is the transformation of Seo and Shin (2016) and of `xthenreg`; `method="fod"` removes the unit effect by forward orthogonal deviations. With interior gaps, FOD discards fewer equations (an interior gap removes three FD equations and two FOD equations). Asymptotic theory exists for FD; for FOD the authors of the command report Monte Carlo evidence only (Nguyen and Lai 2026). On this balanced panel both give $\hat{\gamma} = 0.4194$ (AR(2) p-values 0.738 and 0.944). On an unbalanced panel, estimate both and put them side by side with `pyx.compare` (example 3 does exactly that); `diagnose()` also points it out when FD is used on an unbalanced panel.

Jump or kink

The default allows a discontinuity at γ. `kink=True` imposes continuity; `q` should then be a regressor, or the model is a one-sided hinge and a warning says so. If the function is continuous, the unrestricted estimator of γ converges at rate n^(1/4) to a non-normal limit, whereas the kink estimator is asymptotically normal (Gong and Seo 2026), so the choice affects inference, not only the fit. `continuity_test=True` tests the kink model against the jump model. On the Hansen panel the kink model gives $\hat{\gamma} = 0.032$ and a Hansen p-value of 0.000, against 0.4194 and 0.382 for the jump model.

Threshold search

The defaults are a uniform grid of 100 points between the 5th and 95th percentile of `q`. Uniform grids of 100, 199 and 397 points are nested, so `grid=199` or `grid=397` is the natural check that $\hat{\gamma}$ does not depend on the grid. `grid_type="quantile"` puts points where the data are; `refine` adds the observed values of `q` next to $\hat{\gamma}$; `trim` moves the ends; `min_regime` demands more observations in each regime. On the Hansen panel these choices keep $\hat{\gamma}$ between 0.414 and 0.421 (trim 0.20: 0.4144; quantile grid: 0.4210; observed support: 0.4137; refine 3: 0.4150).

Bootstrap

`n_boot=299` and `grid_ci=100` are the defaults. The smallest possible p-value is 1/(`n_boot` + 1), so a very small `n_boot` cannot show a small p-value. For exploring, `n_boot=99, grid_ci=40` is much faster; use the defaults, or more, for what you report. `boot_type="unit"` follows the algorithm with a proof (Gong and Seo 2026) but is much slower. Always fix a `seed` for results you publish (a seed is recorded anyway).

Variance

`vce="robust"` and `vce="windmeijer"` give the same estimates and thresholds; only the variance of the slopes and the AR statistics that include $\hat{\gamma}$ change. `center=False` reproduces `xtabond2`'s uncentered covariance (and its Hansen statistic).

Before you report

1. Keep `res.seed` and `res.settings_log()`, or `res.to_json()`.
2. Check the grid (`grid=199`) and one other instrument set.
3. Run `pyx.robustness()` and report how stable the threshold is.
4. Report the segments of the confidence set, not only its hull.
5. Report Hansen and AR(2) as diagnostics, with the number of instruments next to the number of units.

---

5. Calling `xthenreg`

```
res = pyx.xthenreg(data, y, x=(), *, q, endogenous=(), exogenous=(), inst=(),
                   id=None, time=None, where=None, static=False, kink=False,
                   grid_num=100, trim_rate=0.4, h_0=1.5, boot_reps=0, seed=None, draws=None,
                   engine="fast")
```

`xthenreg` implements the first-difference estimator of Seo, Kim and Kim (2019) for strongly balanced panels, with the sup-Wald linearity test of Seo and Shin (2016).

Python	Stata	Default	Notes
`y`, `x`	`depvar`, `indepvars`		In Stata the threshold variable is the **second word** of the variable list, `xthenreg y q x1`; here it is the separate argument `q`, so `xthenreg y q x1` is `xthenreg(df, "y", ["x1"], q="q")`.
`endogenous`	`endogenous(varlist)`	none	Excluded from `x`, as in Stata.
`exogenous`	`exogenous(varlist)`	none	Additional exogenous regressors. Accepted by the command; not described in its help file.
`inst`	`inst(varlist)`	none	Additional instruments.
`static`	`static`	`False`	No automatic lag of `y`.
`kink`	`kink`	`False`	Kink model.
`grid_num`	`grid_num(integer)`	`100`	Points of the grid.
`trim_rate`	`trim_rate(real)`	`0.4`	Total trimmed share of `q`.
`boot_reps`	`boost(integer)`	`0`	Replications of the bootstrap linearity test; `0` skips it.
`seed`	(set by `set seed`)	drawn	Seed of the bootstrap weights; recorded in `res.seed`.
`draws`	(none)	`None`	Standard normal weights `(boot_reps, N)` to use instead of the generator.
`engine`	(none)	`"fast"`	`"fast"` or `"literal"` moment computation; both give the same numbers, `"literal"` follows the Mata code line by line.
`h_0`	`h_0(real)`	`1.5`	Multiplier of the kernel bandwidth `h = h_0 · 1.06 · N^(-1/5) · sd(q)` of the covariance (jump model only; the kink model has no bandwidth). Checked against Stata with `h_0(3)` and `h_0(0.7)`.

```
sk = pyx.xthenreg(
    df, "invest", ["cashflow", "tobin_q"], q="debt", id="firm", time="year", boot_reps=49, seed=3
)
print(sk.names)
print(round(sk.gamma, 4), round(sk.scalars["sup_wald"], 2), sk.scalars["boots_p"])
print(sk.ci.round(3))  # e(CI): lower and upper limits of every coefficient, γ last
print(pyx.compare([sk, full()], model_names=["xthenreg", "xtdpthresh"]).to_text())
```

Coefficient names are those of the command. `Lag_y_b` is the slope of `L.y` below the threshold and `x_b` that of a regressor; `cons_d`, `Lag_y_d` and `x_d` are the changes above it (`_d` for δ; `cons_d` is the intercept shift); `r` is the threshold. With `kink=True` the names are `Lag_y_b`, `x_b`, `kink_slope` and `r`; with `static=True` there is no `Lag_y_*`. The table of the result shows them in the same three blocks (lower, change, upper) as `xtdpthresh`.

Inference. `xthenreg` reports an asymptotic interval for every coefficient, the threshold included (`res.ci`, `e(CI)`), not a bootstrap confidence set. Its linearity test is the sup-Wald statistic of Seo and Shin (2016) (`scalars["sup_wald"]`) with a bootstrap p-value `boots_p` that is `-1` when the test was not run (`boot_reps=0`) and can be exactly `0` when no replication reaches the sample statistic. `scalars` also holds `N` (units), `T`, `grid`, `trim`, `bs` (replications), `gamma`, `L` (number of moment conditions) and `bandwidth` (kernel bandwidth of the covariance).

What the result has. Estimates, `se`, `cov`, `ci`, `gamma`, `seed`, `scalars`, `summary()`, and all the table and export methods (`to_latex`, `to_html`, …, `compare`). Graphs, `diagnose()`, `settings_log()`, `predict`, `augment` and `ci_table()` belong to `xtdpthresh` results only; `pyx.plot` called on an `xthenreg` result raises a `SpecificationError` (key `plot_xthenreg`) that says so.

Against `xtdpthresh`. Same moment conditions and weight matrices under `method="fd"`, so the same estimates on the same grid; but the default trimming is 0.4 instead of 0.10, the threshold variable is given differently (see the table above), `xthenreg` needs a strongly balanced panel and has neither `fod`, `td`, `maxlag`, `collapse` nor `iv`, and strictly exogenous regressors are instrumented by their levels where `xtdpthresh` uses their transformed values.

---

6. Reading the output

The printed result

`print(res)` (or just `res` in a notebook) prints a report in the layout of `pyxtabond2` and `pyxthreg`, 80 columns wide: the header (sample, model, options), the threshold with its confidence interval, the threshold-effect test with its bootstrap critical values, the slopes of the three regimes, and the diagnostic tests:

```
                      PyXtdpthresh Estimation Results
================================================================================
Dep. Variable:                   invest    Threshold var.:                   debt
Model:            Dyn. panel threshold    Threshold effect:                 jump
Method:                       Two-Step    Transformation:                     FD
Date:                 Wed, 04 Nov 2026    Group Variable:                   firm
Time:                         10:12:31    Time variable:                    year
No. Observations:                6720    Lag Limits:                      (3, 8)
No. Groups:                       560    Collapse:                           Yes
Obs per group:                  12.00    Time effects:                        No
No. Instruments:                   25    Bootstrap B / seed:       299 / 12345
Covariance Type: Robust (joint, gamma)    Grid points:                      100
================================================================================

                     Threshold Estimates (Level = 95%)
================================================================================
Model                          Threshold            Lower CI            Upper CI
--------------------------------------------------------------------------------
Th-1                              0.4194              0.0122              0.5227
================================================================================

                  Threshold Effect Test (Bootstrap, B = 299)
================================================================================
Test                     Statistic     Prob      Crit10       Crit5        Crit1
--------------------------------------------------------------------------------
Linearity                   0.0087   0.0067      0.0052      0.0061       0.0085
================================================================================

                    Dynamic Panel Threshold Coefficients
================================================================================
                          coef  std err       z   P>|z|    [0.025    0.975]
--------------------------------------------------------------------------------
Lower regime  (debt <= 0.4194)
  L.invest              0.0433   0.2377   0.182  0.8556   -0.4227    0.5092
  cashflow             -0.0069   0.0752  -0.092  0.9269   -0.1543    0.1405
  tobin_q               0.0041   0.0069   0.583  0.5596   -0.0096    0.0177
  debt                 -0.1245   0.1006  -1.237  0.2162   -0.3217    0.0728
Change at the threshold
  _cons                -0.3922   0.1514  -2.590  0.0096   -0.6889   -0.0955 ***
  L.invest              1.5909   0.5956   2.671  0.0076    0.4235    2.7584 ***
  cashflow              0.3729   0.2524   1.477  0.1396   -0.1218    0.8675
  tobin_q               0.0107   0.0438   0.244  0.8073   -0.0752    0.0966
  debt                  0.3208   0.1717   1.868  0.0617   -0.0158    0.6573 *
Upper regime  (debt > 0.4194)
  L.invest              1.6342   0.5335   3.063  0.0022    0.5885    2.6799 ***
  cashflow              0.3660   0.2033   1.800  0.0718   -0.0325    0.7645 *
  tobin_q               0.0147   0.0426   0.346  0.7297   -0.0689    0.0983
  debt                  0.1963   0.1654   1.187  0.2353   -0.1279    0.5205
================================================================================
Significance: * p<0.10, ** p<0.05, *** p<0.01

                              Diagnostic Tests
================================================================================
Diagnostic Test                          Statistic                    P-value
--------------------------------------------------------------------------------
Arellano-Bond AR(1)                      z = -4.61            Pr > z = 0.000
```

```
Arellano-Bond AR(2)                     z = -0.33           Pr > z = 0.738
Hansen test of overid.              chi2(15) = 16.00        Pr > chi2 = 0.382
------------------------------------------------------------------------------
```

Reading it:

- **Header.** Two columns, as in `pyxtabond2`. `No. Observations` is the number of stacked equations (`e(N)`), `No. Groups` the units that contribute equations, `Lag Limits` the `maxlag` range (`.` is no upper limit), `Bootstrap B / seed` the number of draws and the seed (the one drawn when none was given, so the run can be repeated). The date and time are those of the estimation: printing the result again gives the same text.
- **Threshold Estimates.** The estimate and the confidence set of the threshold (grid bootstrap; its hull, see `res.ci_table()` for the segments) at the level of the estimation, `-` without bootstrap.
- **Threshold Effect Test.** The statistic and the bootstrap p-value of the linearity test (and of the continuity test when requested). `Crit10`, `Crit5` and `Crit1` are the bootstrap critical values at 10, 5 and 1 percent: the `k`-th largest draw with `k = floor(size x (B + 1))`, so a statistic above `Crit5` has a p-value of at most 0.05. A size with too few draws shows `-`. For `xthenreg` the statistic is the sup-Wald and the critical values come from the supremum of each replication.
- **Coefficients.** Lower regime, change at the threshold and upper regime (the upper slopes with their own standard error). The interval is the normal one at `res.summary(level=...)`; stars mark p $<$ 0.10, 0.05 and 0.01.
- **Diagnostic Tests.** Arellano-Bond AR(1) and AR(2) (first-difference residuals, joint with the threshold) and the Hansen test, as `z`/`chi2` and p-value. `xthenreg` has none of them and shows `-`.

`res.summary(lang="fr")` prints the same report in French. The numbers are those of `e()` and of the validated objects; the critical values are not Stata objects (they are derived from the bootstrap draws) and are shown for reading only.

Under `kink` the change block holds only the change in the slope of `q`, and the upper block only the slope of `q` above the threshold (`res.upper` is `None` when `q` is not a regressor).

Numbers as objects

```
print(res.params)  # Series indexed 'lower:cashflow', 'change:_cons', ... (e(b))
print(res.se)  # standard errors from the reported variance (sqrt of the diagonal of e(V))
print(res.cov.shape)  # e(V)
print(res.upper)  # upper-regime slopes and standard errors (e(b_upper), e(V_upper))
print(res.gamma, res.gamma_ci)
print(res.to_dataframe())  # estimate, se, z, p and the interval, indexed by (block, variable)
```

`res.gamma_ci` is the **hull** of the confidence set: the smallest and largest accepted candidate. When the set has several segments (`res.scalars["ci_nseg"] > 1`), the range between them contains rejected values, so look at `res.ci_table()` or `res.plot("ci")` and report the segments.

`res.scalars`

A dictionary with the names of Stata's `e()` (`nan` when a value does not exist for the fit):

Group	Names
Sample	`N` (equations), `N_units`, `N_clust`, `N_switch` (units whose regime changes within their equations at $\hat{\gamma}$)
Instruments	`N_iv`, `N_iv_common`, `N_iv_dep`, `N_iv_dep_near`, `iv_dep_res` (columns dropped as common to all units or numerically dependent, and the largest residual of a dropped one)
Estimator	`gamma`, `obj` (criterion), `estimator_twostep` (0 = one-step fallback), `W1_fallback`, `level`, `joint_vce`, `vce_applied`, `wind_same_threshold`, `gamma_bw`, `q_nvals_bw`, `kink_refined`
Grid	`q_lo`, `q_hi` (trimming bounds), `grid_admitted`
Hansen	`hansen`, `hansen_df`, `hansen_p`
Serial correlation	`ar1`, `ar1_p`, `ar2`, `ar2_p`, `ar_joint`, `ar1_cond`, `ar2_cond`; `ar1_b0`, `ar1_T1`, `ar1_TT`, `ar2_b0`, `ar2_T1`, `ar2_TT` are the pieces of the statistics, kept for checking against Stata
Confidence set	`gamma_lo`, `gamma_hi`, `ci_nseg`, `ci_empty`, `ci_incomplete`, `ci_unresolved`, `boundary_warn`
Tests	`pval_lin`, `boot_linearity_valid`, `pval_cont`, `boot_continuity_valid`, `citest_gamma`, `citest_D`, `citest_crit`, `citest_accept`, `citest_p`, `citest_status`, `citest_draws`, `rseed`

The tables behind the confidence set and the regimes:

```
print(res.ci_table().head())  # gamma, statistic, critical, accepted, n_valid, status
```

```
print(res.regime_table())  # observations, units, share and the mean of each variable by regime
```

In `ci_table()` the `status` is 1 for a resolved test, 2 for accepted with a zero statistic, 3 for a candidate that is not admissible, and 4 to 6 for unresolved ones. Statuses 3 to 6 are not rejections.

A reading order that works

1. **The sample.** `n_units`, `n_obs` and `n_instruments` (or `res.settings_log()`): were units dropped, and is the number of instruments small next to the number of units? Many instruments relative to units makes the Hansen test unreliable (Roodman 2009).
2. **The threshold.** $\hat{\gamma}$ is a grid point (a point between grid points under `kink` or `refine`). It has no standard error; inference rests on the confidence set. Check the number of segments, `boundary_warn` (a bound at the edge of the confidence-set grid: 1 lower, 2 upper, 3 both), `ci_incomplete` (no interval is reported if some candidates are unresolved) and `ci_empty`.
3. **Linearity.** The test of H_0: $\delta = 0$. A rejection says a threshold effect exists; it does not say where the threshold is. The p-value is $(1 + R)/(1 + B)$.
4. **Hansen J.** A diagnostic, with `N_iv - k - 1` degrees of freedom. Its chi-square reference presumes regular identification, which fails near a continuous threshold. A large p-value does not show that the instruments are valid. With `center=False` it equals the statistic of `xtabond2` for the same instruments.
5. **AR(1) and AR(2).** AR(1) is expected to reject. A rejection of AR(2) indicates serial correlation in the errors, which invalidates the lagged instruments: reconsider `maxlag` or the specification (the second example of the Stata help file shows this with `maxlag(2 10)`).
6. **The fallbacks.** `estimator_twostep == 0`, `W1_fallback`, `vce_applied == 0` mean the command fell back to a simpler computation; `diagnose()` and the printed notes say which.
7. **Discrete `q`.** In the jump model, the joint variance and the AR statistics with $\hat{\gamma}$ assume that `q` has a continuous density at the threshold. `q_nvals_bw` below 10 triggers a warning; the conditional statistics `ar1_cond` and `ar2_cond` do not use that assumption.

`diagnose()` and `settings_log()`

`res.diagnose()` lists, in plain language, what deserves attention, each finding with a recommendation and a level (warning or notice). It checks: the number of instruments against the number of units; a Hansen test that rejects or sits suspiciously close to one, and an AR(2) test that rejects; a threshold at an edge of the search grid; the confidence set (empty, incomplete, touching the edge of its grid, several segments, very wide, or so narrow that the grid is too coarse); a flat criterion profile or a distant rival threshold that fits almost as well; a small regime or a threshold variable with few distinct values; first differences on an unbalanced panel; the one-step and first-weight fallbacks; and the result of the continuity test when it was run. In English or French: `res.diagnose(lang="fr")`.

`res.settings_log()` is the journal of what the estimation *really* used: the sample (units with equations, dropped units), the model, the instruments kept and removed, the first-step weight, the grid (requested, evaluated, admitted, range, smallest regime allowed), the variance estimator, the bootstrap (type, draws, seed, points of the grid) and the package version. Keep it with the results.

```
print(res.diagnose())
print(res.settings_log())
```

Two diagnoses, read

Serial correlation. The same model with instruments from lag 2 (the third example of the Stata help file):

```
ar2 = full(maxlag=(2, 10))
print(ar2.diagnose())
```

```
Diagnostic report: 1 warning(s), 1 notice(s)

[warning] The AR(2) test rejects the absence of second-order serial correlation in the differenced residuals (p =
↪  0.013): lagged levels may not be valid instruments.
    Recommendation: Start the instruments one lag deeper (for example maxlag=(3, 8)) or add dynamics.
[notice] The confidence set of the threshold reaches the lower edge of the candidates: it is cut by the grid, not by the
↪  data.
    Recommendation: Widen the grid (lower trim) before reading the interval.
```

The warning is the one to act on: the instruments are suspect, so the estimate ($\hat{\gamma} = 0.0121$, near the lower end of the grid) should not be reported as it stands. The notice says the confidence set touches the edge of

the candidates: it is cut by the grid, so do not read its end as a bound.

Too many instruments. The same model on the first 79 firms, without `collapse` and with every lag:

```
small = pyx.xtdpthresh(
    df[df["firm"] < 80], "invest", **{**MODEL, **infer, "collapse": False, "maxlag": None}
)
print(small.n_units, small.n_instruments, small.scalars["estimator_twostep"])
print(small.diagnose())
```

```
79 377 0.0
Diagnostic report: 2 warning(s), 3 notice(s)

[warning] 377 instruments for 79 units: with more instruments than units the Hansen test loses its power and the weight
↪  matrix is poorly estimated.
    Recommendation: Reduce the instruments: collapse=True or limit the lags with maxlag=(2, 4).
[warning] The estimate stays at one step: the second-step weight could not be used.
    Recommendation: Reduce the instruments (collapse=True or limit the lags with maxlag=(2, 4)); the Hansen test is not
↪  available at one step.
[notice] The confidence set of the threshold reaches the lower and upper edge of the candidates: it is cut by the grid,
↪  not by the data.
    Recommendation: Widen the grid (lower trim) before reading the interval.
[notice] The confidence set covers 100 % of the search grid: the threshold is weakly identified.
    Recommendation: Treat the threshold with caution; more units, a longer panel or a more variable debt would help.
[notice] The criterion profile is flat: 100 % of the thresholds are within 3.8 of the minimum.
    Recommendation: The estimated threshold is not sharply determined; compare it with the confidence set and plot(fit,
↪  'profile').
```

Here the cause is upstream of everything else: with more instruments than units the second-step weight could not be used, the command falls back to one step (`estimator_twostep` is 0, and there is no Hansen test), and the confidence set is the whole grid. The five findings share one remedy, fewer instruments.

---

7. Working with a result

Predictions and residuals. `res.predict(kind, frame)` returns a Series for the estimation data; `kind` is `"residuals"`, `"xb"`, `"ar_residuals"` (the first-difference residuals used by the AR tests; they equal `"residuals"` under FD) or `"regime"` (1 when `q` > $\hat{\gamma}$, missing where `q` is). They are the series computed during the estimation, not recomputed. They are the counterparts of Stata's `predict, residuals | arresiduals | xb | regime`.

`res.augment(frame, kinds=…, prefix="", overwrite=False)` returns a copy of the frame with those series as columns (default `kinds`: residuals, xb, regime), named after the series with `prefix` in front. A column that already has the name is an error (`augment_exists`) unless `overwrite=True`.

```
aug = res.augment(df, kinds=("residuals", "regime"))
print(aug["regime"].mean())
both = res.augment(df, prefix="m1_", overwrite=True)
print([c for c in both.columns if c.startswith("m1_")])
```

Tables. `res.to_latex(path)`, `to_html`, `to_markdown`, `to_text`, `to_csv`, `to_word` and `to_excel` (the last two need `pip install "pyxtdpthresh[export]"`) write the academic table and return its text. `res.table(**options)` returns the table object, and `pyx.compare([a, b, …], model_names=[…])` puts several models side by side (`xthenreg` against `xtdpthresh`, FD against FOD, jump against kink). The options, shared by all of them:

Option	Effect
`statistic`	under each estimate: `"se"` (default), `"t"`, `"p"` or `"ci"` (interval at `level`)
`digits`, `star_levels`, `level`	decimals (3), significance levels (0.1, 0.05, 0.01), level of the intervals
`model_names`, `column_labels`	column names, and a group label above them
`covariate_labels`, `dep_var_label`	new names for variables (`{"cashflow": "Cash flow"}`; a `"block:variable"` key names one regime) and for the dependent variable
`keep`, `omit`, `order`	regular expressions selecting, dropping and ordering rows (against the name or `block:variable`)
`add_lines`, `notes`	extra lines under the coefficients (`[("Firm effects", ["Yes", "Yes"])]`) and notes
`title`, `label`, `font_size`, `single_row`	caption, LaTeX label, LaTeX size command, one line per estimate
`omit_stat`	diagnostic lines to drop (`"n_obs"`, `"hansen_p"`, …)

Option	Effect
`lang`	`"en"` or `"fr"` for labels and the decimal sign
`full_output=True`	(of `res.table` and `res.to_*`) one more line per scalar of `e()`

```
fod = pyx.xtdpthresh(df, "invest", method="fod", bootstrap=False, **MODEL)
kink = pyx.xtdpthresh(df, "invest", kink=True, bootstrap=False, **MODEL)
table = pyx.compare([fast, fod, kink], model_names=["FD jump", "FOD jump", "FD kink"])
print(table.to_text())

custom = res.table(
    statistic="ci",
    digits=2,
    keep=["cashflow", "debt"],
    covariate_labels={"cashflow": "Cash flow"},
    add_lines=[("Firm effects", ["Yes"])],
    title="Investment",
)
print(custom.to_text())
res.to_latex("table.tex", full_output=True)
```

Plots. `res.plot(kind, path=None, lang=None, **options)` returns a matplotlib `Figure` and writes it to `path` (`.png`, `.pdf` or `.svg`) when given. They are `xtdpthresh` results' methods (Section 5).

`kind`	Draws	Options
`profile`	the GMM criterion over the grid of thresholds	
`ci`	the grid-bootstrap statistic against its critical value, accepted set shaded	
`linearity`	the bootstrap distribution of the linearity test	
`coef`	slopes of both regimes and their change, with intervals	`variables`, `free_scale`
`regimes`	the distribution of `q` and the share in each regime	`bins` (30), `trim` (0.01 cut from each tail)
`effect`	the effect of one regressor according to the regime	`var`, `trim`
`switch`	the share of the upper regime by period, and the units that change regime	
`instruments`	instrument columns by period against the number of units	
`residuals`	transformed residuals by period and the AR(1) and AR(2) tests	

`ci` and `linearity` need the bootstrap. `robustness` takes a `RobustnessResult`; `coverage`, `power` and `distribution` take Monte Carlo results (one, a list, or a `{name: result}` mapping) through `pyx.plot(result, kind)`. Figures follow the style of the article and come in English or French (`lang="fr"`).

```
fig = res.plot("profile", path="profile.png")
res.plot("effect", var="cashflow", path="effect.png")
res.plot("regimes", bins=20, lang="fr", path="regimes_fr.png")
```

Robustness. `pyx.robustness(df, y, x, q=…, settings=…, product=False, **options)` re-estimates the model on a set of settings and gathers the threshold with its confidence set, the slopes, the linearity p-value and the diagnostics of each run. `settings` maps an option to the values to try (default: `grid` 50/100/200, `trim` 0.05/0.10/0.15, `collapse`, `method`, `kink`); the options of the call are the reference. By default one option changes at a time; `product=True` runs every combination. All runs share one bootstrap seed, so the settings, not the draws, explain the differences.

```
rob = pyx.robustness(df, "invest", n_boot=99, grid_ci=40, seed=1, **MODEL)
print(rob.table[["gamma", "gamma_lo", "gamma_hi", "ci_nseg", "in_reference_ci"]])
print(rob.stability())  # share of the other runs whose threshold stays in the reference set

own = pyx.robustness(
    df,
    "invest",
    n_boot=49,
    grid_ci=20,
    seed=1,
    settings={"trim": [0.05, 0.15], "method": ["fod"]},
    product=True,
    **MODEL,
)
print(own.table.index.tolist())  # reference, then the combinations
rob.plot(path="robustness.png")
print(rob.coefficients().head())  # every slope of every run, with its standard error
```

`rob.table` has one row per run; `stability(use="segments")` counts a run as stable when its threshold lies in an accepted segment of the reference set (`use="hull"` uses the hull). A run that fails is kept in the table with its message and does not stop the analysis; only a failure of the reference does. `rob.stargazer()` returns the table of all the runs side by side.

Records. `res.to_json(path)` writes a reproducible record of the fit with seven keys: `model`, `specification` (variables), `options` (every option, as used), `scalars`, `ci_grid` (the inversion table), `coefficients` and `environment` (versions of the package, Python, numpy, scipy, pandas and the platform). `res.fit` holds everything the estimator computed, as frozen containers.

```
import json

record = json.loads(res.to_json())
print(list(record), record["options"]["seed"])
```

---

8. Reproducibility, speed and memory

Seeds. Everything random in `xtdpthresh` comes from `seed`. The Stata generator is reproduced (`set seed`, `uniform()`, `invnormal()`), so a seeded bootstrap draws the same weights as Stata. A seed that is omitted is drawn and recorded in `res.seed`, `res.settings_log()` and `res.to_json()`.

Global options.

```
# not run by the tests
pyx.set_options(lang="fr")  # messages, reports, figures in French
pyx.set_options(threads=4)  # threads of the matrix products (default: half the logical cores)
pyx.set_options(progress=True)  # progress line during long bootstraps, on standard error
with pyx.option_context(lang="fr"):
    ...  # temporary change
```

`set_options` returns the previous options. Numba kernels run on one thread, which is faster than letting them compete with the matrix products.

First call. The Numba kernels are compiled the first time they are used and cached on disk. `pyx.warmup()` pays that cost ahead of time (a few seconds, once per environment).

Where the time goes. The bootstrap is 80–92 % of a full run. For a quick look at a specification, estimate with `bootstrap=False`; increase `n_boot` and `grid_ci` for the final run. With many instruments the cost is in one matrix product that already runs close to the speed of the machine; `collapse` and `maxlag` shrink it.

Memory. Memory grows with the number of equations times the number of raw instrument columns. A note is issued before a fit expected to need more than 2 GB, and a warning above 8 GB; `collapse`, `maxlag` and fewer draws reduce it.

---

9. Simulation and Monte Carlo

`pyx.simulate_panel(design, n, t, seed=…)` draws a panel from a named design and returns it in long form (`id`, `t`, `y`, `q`, …), ready for the estimators; `pyx.monte_carlo(design, n, t, …)` replicates the estimation and summarises bias, coverage, size and power.

Design	Source	What it is
`gong_seo`	Gong and Seo (2026), §5	`q` is an AR(1) correlated with the errors, predetermined
`gong_seo_endog`	Gong and Seo (2026)	the same with `q` weakly endogenous
`linear`	Gong and Seo (2026)	no threshold effect: the design for the **size** of the linearity test
`seo_shin_jump`, `seo_shin_cont`	Seo and Shin (2016), §6	self-exciting, `q = y(t-1)`; discontinuous (eq. 16) and continuous (eq. 17)
`seo_shin_jump_common`, `seo_shin_cont_common`	Seo and Shin (2016), §6.1	intercept fixed across regimes
`seo_shin_exogenous`, `seo_shin_exogenous_common`	Seo and Shin (2016), App. A.4	`q ~ U[-1, 1]`, independent of the errors

`simulate_panel(design, n, t, *, seed, burn=50, unbalanced=0.0, min_obs=3, **changes)` returns a frame with the columns `id`, `t` (1 to `t`), `y` and `q`; the panel is a function of `seed` alone. `burn` periods are simulated and discarded first, so that `q` and `y` are close to their stationary behaviour. `unbalanced` is the share of observations dropped at random (the remaining values do not change), and a unit left with fewer than `min_obs` observations is removed. `**changes` override any parameter of the design (`gamma`, `sigma`, `rho`, `rho_eu`, `beta_y`, `delta0`, …) and `jump=` sets the jump of the regression function at the threshold (`0` is a kink). `resolve_design(name, **changes)` returns the resulting `Design`: its `truth` holds the true coefficients (named like the estimates, plus `gamma`) and `arguments()` the call that estimates it (the roles of the variables and `collapse`).

`monte_carlo(design, n, t, *, reps=200, seed, workers, alpha=0.05, shifts=(0.10, 0.25, 0.50), simulation=None, **options)` replicates the estimation. `simulation` is a dictionary of generator parameters (`jump`, `unbalanced`, `burn`, any parameter of the design); `options` are `xtdpthresh` options (`n_boot`, `grid`, `grid_ci`, `level`, …) and override the design's arguments; `alpha` is the level of the linearity test; `shifts` are the `c` of the alternatives γ `+ c` whose rejection rate is reported. The result keeps one row per replication (`to_dataframe()`, with `status` and `error` for the replications that failed) and answers:

Method	Returns
`estimates()`	bias, RMSE and coverage of the threshold and of every slope, by parameter
`threshold_coverage()`	coverage of the true threshold by the confidence set (segments) and by its hull, and the share of incomplete sets
`linearity()`	rejection rate of the linearity test: the power with a threshold effect, the size under design `linear`
`power()`	rejection rate of γ `=` γ_0 `+ c`, `c = 0` included (the size of the threshold test)
`summary()`	a short text report

```
panel = pyx.simulate_panel("gong_seo", 200, 6, seed=3)
print(panel.head(3))

from pyxtdpthresh.extras.simulation import resolve_design

design = resolve_design("gong_seo", gamma=0.5, jump=0.2)
gaps = pyx.simulate_panel(design, 100, 6, seed=3, burn=20, unbalanced=0.15, min_obs=3)
print(len(gaps), gaps["id"].nunique(), design.truth["gamma"], round(design.jump, 2))
gap_fit = pyx.xtdpthresh(gaps, "y", id="id", time="t", bootstrap=False, **design.arguments())
print(round(gap_fit.gamma, 3), gap_fit.n_obs)

mc = pyx.monte_carlo("gong_seo", 100, 5, reps=4, seed=11, workers=1, n_boot=49, grid_ci=20)
print(mc.threshold_coverage())  # coverage of the true threshold by the set and by its hull
print(mc.linearity())  # rejection rate: the power here, the size under design "linear"
print(mc.power())  # rejection rate of gamma + c, c = 0 included

mc2 = pyx.monte_carlo(
    "gong_seo",
    100,
    5,
    reps=4,
    seed=11,
    workers=1,
    n_boot=49,
    grid_ci=20,
    alpha=0.10,
    shifts=(0.2, 0.4),
    simulation={"jump": 0.5, "unbalanced": 0.1},
)
print(mc2.power())
print(mc2.estimates().round(3))
```

The coverage and power figures of four replications mean nothing; the tables in `docs/montecarlo/` use hundreds. The calls are the same.

The experiment is a function of `seed` alone, whatever `workers` is. `workers=None` uses `min(cpu_count, reps)` processes of one thread each; `workers=1` runs in the calling process.

On Windows, where worker processes are started by spawning, a script that runs replications in several processes must be a real file with the usual guard:

```
# not run by the tests
import pyxtdpthresh as pyx
```

```
if __name__ == "__main__":
    mc = pyx.monte_carlo("gong_seo", 400, 6, reps=200, seed=1, workers=4)
    print(mc.summary())
```

The worker processes re-import the main file; without the guard each one would start its own experiment, and a script read from standard input (`python -`) has no file to re-import: the replications fail. `examples/04_monte_carlo.py` is a complete case.

The bundled results of larger experiments, and what they say about the agreement with the papers (coverage agrees; the power to reject shifted thresholds is lower than Table 2 of Gong and Seo, cause not established), are in `docs/montecarlo/`.

---

10. Errors and warnings

Exceptions all derive from `PyxdpthreshError`:

Class	Raised when
`SpecificationError` (also a `ValueError`)	the call breaks a rule of the Stata command; `.stata_code` is Stata's return code (198 for most option errors) and `.key` names the rule
`DataError` (also a `ValueError`)	the data cannot be used: repeated (unit, time) pairs, missing keys, no usable observation, an unbalanced panel for `xthenreg`, a point estimation that finds no valid solution
`NumericalError`	a computation fails numerically

All of them derive from `pyx.PyxdpthreshError` and are importable from the package (`pyx.SpecificationError`, `pyx.DataError`, `pyx.NumericalError`). `SpecificationError` and `DataError` are also `ValueErrors`.

```
try:
    pyx.xtdpthresh(df, "invest", grid=5, bootstrap=False, **MODEL)
except pyx.SpecificationError as error:
    print(error.stata_code, error.key)  # 198 opt_grid
    print(error)
```

The return codes match the command's. On 19 hostile inputs (minimal panels, a constant, binary or rare threshold variable, an empty regime, duplicated regressors, fewer units than instruments, a null outcome) the package returns the same code as Stata in all 19.

"No threshold admitted a valid GMM solve" (a `DataError`) means that at every candidate threshold the estimator could not be computed. The usual causes are too few units or periods, more instruments than the data can support, a threshold variable with too little variation, or a regime that is too small. Remedies: `collapse=True`, a tighter `maxlag`, a larger `trim`.

Warnings are `pyx.PyxdpthreshWarning` (a `UserWarning`), so the `warnings` module controls them, and `warn=False` silences most of them at the source. They cover: many instrument columns, memory, a kink with `q` not among the regressors, `refine` not applied after a one-step fallback, a Windmeijer correction that could not be applied, and the legacy `coef_boot`.

---

11. What differs from Stata

Stata	Python
`xtset firm year`	`id="firm", time="year"`, or a `MultiIndex`, or `from_stata`
`if` / `in`	`where=`; no `in`
`e(b)`, `e(V)`, `e(N)`, …	`res.params`, `res.cov`, `res.scalars["N"]`, …
`ereturn list, all` (hidden results)	`res.scalars` includes the hidden ones that are kept for checking
`predict, residuals | arresiduals | xb | regime`	`res.predict("residuals" | "ar_residuals" | "xb" | "regime", frame)`
`estimates store` / `esttab`	the result objects, `pyx.compare([...])` and the `to_*` methods
`set level`	`level=` of the call (95 by default)

Stata	Python
`rseed(#)`, not restored after the command	`seed=`; no global state is changed
`verbose`, `exportgmm`	accepted with no effect
`xthenreg … h_0()`	`xthenreg(…, h_0=)`
a missing value in `if` counts as true	counts as false
Mata matrices left in memory	frozen containers under `res.fit`

What the package adds: the result classes with tables, plots and a JSON record; `diagnose()` and `settings_log()`; `robustness()`; the simulation toolkit; bundled data; English and French messages. What it does not change: the estimators, the bootstrap draws, the tests and their conventions, which are checked against Stata 17 MP to four decimals.

A Methodological note

This appendix is the methodological note of the package. It states what the two estimators compute (Sections A.2–A.5), how the package computes it and why it is fast (Section A.4), how the bootstrap inference is organised (Section A.6), where it departs from the algorithm of Gong and Seo (Section A.7), what `xthenreg` adds (Section A.8), which conventions of the Stata commands are reproduced on purpose (Section A.9), what has been compared with Stata and how (Section A.10), and where every Stata function went (Section A.11).

Three kinds of statements are mixed and kept apart in the text. A *model* statement comes from the papers the commands implement (Seo and Shin 2016; Seo, Kim and Kim 2019; Gong and Seo 2026; Nguyen and Lai 2026). A *convention* is a behaviour of the Stata code that the package reproduces so that the numbers agree; Section A.9 lists them. An *implementation* statement says how the package obtains a result and is checked by tests, not by a theorem. Nothing here claims more theory than the papers prove: where a scheme is an approximation that the theory does not cover, the text says so.

A.1 Notation

Symbol	Meaning
$i = 1, \dots, n$	units (the clusters of the variance and of the bootstrap); n is `e(N_units)`
e	one transformed equation: a unit and a period. There are N of them, `e(N)`, stacked by unit then period
y_{it}, x_{it}, q_{it}	dependent variable, the k regressors (the lag of y first, in the dynamic model) and the threshold variable
γ	the threshold; Γ is the grid of candidates, G its size
$\mathcal{T}$	the transformation (first differences or forward orthogonal deviations) as a sparse matrix that maps the *level* observations to the N equations
$\theta = (\beta', \delta')'$	coefficients; $p = 2k + 1$ in the jump model, $p = k + 1$ in the kink model
$\mathbf{X}(\gamma)$	transformed regressors at γ: $\mathcal{T}\,[\,x,\ \mathbf{1}(q > \gamma),\ x\,\mathbf{1}(q > \gamma)\,]$ (jump) or $\mathcal{T}\,[\,x,\ (q-\gamma)\mathbf{1}(q > \gamma)\,]$ (kink)
$\mathbf{Z}$	the $N \times L$ instrument matrix; z_e is its row e
$D(\gamma)$, m_y	$D(\gamma) = \mathbf{Z}'\mathbf{X}(\gamma)/N$ $(L \times p)$ and $m_y = \mathbf{Z}'\mathcal{T}y/N$ (L)
$g_i(\theta, \gamma)$	moment contribution of unit i: $\sum_{e \in i} z_e\,(\mathcal{T}y_e - \mathbf{X}_e(\gamma)'\theta)$
$W_1,\ W_2$	weight matrices of the first and second step
$Q(\theta, \gamma)$	criterion $N\,\bar{g}'\,W\,\bar{g}$ with $\bar{g} = m_y - D(\gamma)\theta$

Stata counts the stacked equations N in `e(N)` and the units in `e(N_units)`; the note follows that. (The papers write n for the units and index the moments by period.)

A.2 Model and moment conditions

The commands fit the single-threshold dynamic panel model of Seo and Shin (2016),

$$y_{it} = x_{it}'\beta + (1, x_{it}')\,\delta\ \mathbf{1}(q_{it} > \gamma) + \eta_i + \varepsilon_{it}, \tag{1}$$

where x_{it} contains $y_{i,t-1}$ unless the model is static. The regression function jumps at γ by $\delta_1 + x'\delta_2$. With `kink`, continuity is imposed,

$$y_{it} = x_{it}'\beta + \delta_q\,(q_{it} - \gamma)\,\mathbf{1}(q_{it} > \gamma) + \eta_i + \varepsilon_{it}, \tag{2}$$

and q should be a regressor so that it has a slope below γ. The unit effect η_i is removed by a transformation.

First differences. The equation of unit i at period t exists when the unit has rows at the calendar periods t and $t-1$ (a gap in the dates gives no equation) and both are in the equation sample and complete in y, q and the regressors. It is $y_t - y_{t-1}$ on $x_t - x_{t-1}$, and touches two level observations with weights $+1$ and -1.

Forward orthogonal deviations. An equation row followed by $T_f \geq 1$ later equation rows of the same unit is $c\,(y_t - \overline{y}_{\text{later}})$ with $c = \sqrt{T_f/(T_f+1)}$ and the same for every regressor; the dates of the later rows do not matter, and the last row of a unit has no equation. In weights: $+c$ on the row, $-c/T_f$ on each later row.

Triplets. Both transformations are stored as triplets (equation, level observation, weight), i.e. as a sparse matrix $\mathcal{T}$ with two entries per row for first differences. The regime regressors depend on γ through the level observations only, so $\mathbf{X}(\gamma)$ is the transform of the level regressors $[x,\ \mathbf{1}(q > \gamma),\ x\,\mathbf{1}(q > \gamma)]$: the regressors at any threshold come from one sparse product, and the equations are never restacked. This is what makes the cumulative-sum algorithm of Section A.4 possible.

Instruments. The vector z_e of an equation at period t contains (Arellano and Bond 1991; Roodman 2009), each in its own block of columns per period unless `collapse`: a constant; under first differences the levels $y_{t-2}, y_{t-3}, \ldots$; the transformed strictly exogenous regressors; the lagged levels of the endogenous regressors from $t-2$ and of the predetermined regressors from $t-1$; and the external instruments dated t. Under forward orthogonal deviations the dates are one period later, since the transformed error involves later periods. `maxlag(lo hi)` bounds the lags, the minimum being 2 for y and the endogenous variables and 1 for the predetermined ones. With `collapse` the per-period blocks are replaced by one shared block (one column per lag instead of one per lag and period). An equation without an instrument other than the constant is dropped. The instrument matrix does not depend on γ.

Columns that carry no information are then removed, in the order of the Stata command: (1) columns that are multiples of the period constants (a variable common to all units in a period) are zeroed; (2) columns that are exactly zero are dropped; (3) exact duplicates are dropped, the first kept; (4) columns that are linear combinations of the earlier ones are dropped (squared relative residual below 10^{-13}). The counts are `e(N_iv_common)`, `e(N_iv_dep)` and `e(N_iv_dep_near)`.

Moment conditions. With $T(\cdot)$ the chosen transformation, the estimator uses $\mathrm{E}[z_e\,\mathcal{T}(\varepsilon)_e] = 0$. The sample moment is $\bar{g}(\theta,\gamma) = m_y - D(\gamma)\theta$ and the criterion is $Q(\theta,\gamma) = N\,\bar{g}'W\bar{g}$.

Time effects. With `td` the transformed period dummies are partialled out of the transformed y, regressors and instruments (Frisch–Waugh–Lovell). Under first differences the transformed dummy of a period is the same for every unit, so the projection is exact demeaning within the period; a period with one equation is demeaned to zero and its row is dropped. Under forward orthogonal deviations the dummies receive the same transform as the data and the system is projected on them with a generalised inverse (`invsym`); a row that the projection saturates is dropped. The projections M are symmetric and idempotent, so $(M\mathbf{Z})'\,d\mathbf{X} = \mathbf{Z}'\,M\,d\mathbf{X}$ and the regime moments are formed from the projected instruments and the unprojected level regressors.

A.3 Estimation

Closed form at a threshold. For a weight matrix W and a given γ the criterion is quadratic in θ, so

$$\hat{\theta}(\gamma) = (D'WD)^{-1}D'W\,m_y, \qquad \bar{g} = m_y - D\hat{\theta}, \qquad Q = N\,\bar{g}'W\bar{g}. \tag{3}$$

The inverse is Mata's `invsym`, or the gated `syminv` of `xtdpthresh` (Section A.9): a singular normal matrix is not an error but a candidate that is not admissible.

First step. θ_1 and γ_1 minimise Q over the grid with W_1. Under first differences W_1 is the MA(1) weight of Arellano and Bond (1991): the transformed errors of consecutive equations of a unit overlap in one term, so with $S_0 = \mathbf{Z}'\mathbf{Z}/n$ and $S_1 = n^{-1}\sum_i \sum_t z_{i,t-1}z'_{it}$ over consecutive equations of the same unit,

$$W_1 = (2S_0 - S_1 - S_1')^{-1}. \tag{4}$$

If that matrix cannot be inverted, $(\mathbf{Z}'\mathbf{Z})^{-1}$ is used, then the identity (`e(W1_fallback)` = 1 or 2). Under forward orthogonal deviations $W_1 = (\mathbf{Z}'\mathbf{Z}/N)^{-1}$.

Second step. With the first-step residuals r_1 at (θ_1,γ_1), the cluster covariance of the moments is

$$\Omega = \frac{1}{N}\sum_{i=1}^{n} g_i g_i' - \frac{1}{nN}\Big(\sum_i g_i\Big)\Big(\sum_i g_i\Big)', \tag{5}$$

the second term being dropped under `nocenter`. $W_2 = \Omega^{-1}$ is built *once*, from the first-step estimate, and then *held fixed* while the second-step criterion is evaluated at every other threshold; $\hat{\gamma}$ minimises that criterion over the grid, with ties going to the smaller threshold (relative tolerance 10^{-12}). This is what the Stata search does, and it is why the criterion of every candidate can be computed from sufficient statistics.

One-step fallback. If Ω cannot be inverted, no candidate can be solved with W_2, or the criterion does not vary over the admitted candidates of the second step, the first-step estimates are reported with the sandwich of W_1 (`e(estimator_twostep)` $= 0$) and without Hansen's J. If the first-step criterion itself is flat, the command stops with return code 498.

The grid. The effective support of q is the set of q values of the level observations that enter a stacked equation (periods t and $t-1$ under first differences); `gridsample(observed)` keeps the current-row values only. The trimming bounds are the type-7 quantiles at `trim`/2 and 1−`trim`/2. A uniform grid has equally spaced points between the bounds; a quantile grid places them on the empirical quantiles (duplicates collapsed). A candidate is admitted when both regimes have at least m observations, where the default floor is $m = \max\{2, \min(\#\{q \le q_{lo}\}, \#\{q > q_{hi}\})\}$ and `minregime` can only raise it, and when there are more instruments than parameters. At least 20 equations, 5 units and 20 usable values of q are required. The grids of 100, 199 and 397 uniform points are nested ($g \to 2g-1 \to 4g-3$), so a finer grid is a direct check.

Refinement. Under `kink` the criterion is continuous in γ, and $\hat{\gamma}$ minimises it between the two grid points next to the grid minimum: up to 8 rounds of 40 and then 10 trial points, shrinking the bracket until its spacing is below 10^{-6} of its width (`e(kink_refined)`). Under the jump model `refine(#)` adds the observed values of q between the neighbours of the minimiser, in at most # rounds, and re-minimises the second-step criterion with the same fixed W_2; the confidence set and the tests keep using the initial grid.

A.4 Cumulative sums: the criterion on a whole grid in one pass

Only the regime columns of $\mathbf{X}(\gamma)$ depend on γ. For instrument j and level regressor k the regime moment is

$$S_{jk}(\gamma) = \sum_{e} \mathbf{Z}_{ej} \sum_{(l,w)\in e} w\, W_{lk}\, \mathbf{1}(q_l > \gamma), \tag{6}$$

where the inner sum runs over the level observations l of equation e with their weights w (for the jump model $W_l = (1, x_l)$). Sort the grid, $\gamma_1 < \cdots < \gamma_G$, and let $m_l = \#\{g : \gamma_g < q_l\}$. Then $\mathbf{1}(q_l > \gamma_g) = 1$ exactly when $g \le m_l$, so each term $c = \mathbf{Z}_{ej}\, w\, W_{lk}$ contributes to the first m_l grid points and to no other. The kernel adds each contribution *once*, to the bucket m_l, and a reverse cumulative sum over the buckets gives every grid point:

$$A[m] = \sum_{\{p:\, m_p = m\}} c_p, \qquad S(\gamma_g) = \sum_{m \ge g} A[m]. \tag{7}$$

The cost is $O(P \cdot \mathrm{nnz} \cdot K + G\,L\,K)$, with P the number of pairs and nnz the number of non-zero entries of an instrument column, against $O(G\,P\,\mathrm{nnz}\,K)$ if each threshold were recomputed. The kernel runs over instruments (one Numba thread owns the slice $S[:, j, :]$), reads $\mathbf{Z}$ in compressed-column form so that the block-sparse layout costs nothing, and places a missing q in bucket 0, so that it is never above the threshold.

For the kink model the regime column is $(q-\gamma)\mathbf{1}(q>\gamma) = q\,\mathbf{1}(q>\gamma) - \gamma\,\mathbf{1}(q>\gamma)$, so the same pass with level columns $(1, q)$ gives $S_\kappa(\gamma) = S_q(\gamma) - \gamma\, S_1(\gamma)$. The full matrix is $D(\gamma_g) = [\,\mathbf{Z}'\mathbf{X}_{\mathrm{lin}}/N \;\big|\; S(\gamma_g)/N\,]$, the first block being the same at every threshold.

Solving the whole grid. Given $D(\gamma_g)$ for all g, the estimates (3) are batched matrix products (BLAS) and one batched `invsym`; this step is dominated by dense algebra and is plain NumPy. On the 565×15 panel the regime moments of 100 grid points take 4 ms against 278 ms for the straightforward reference implementation that is kept for the equivalence tests (block C of the project plan), and the two agree to 10^{-12}. At realistic sizes the remaining time is in the bootstrap (Section A.6), whose criterion is one matrix product that already runs close to the speed of the machine, which is why no further algebraic gain exists there: an idea of sharing the unrestricted fit across candidates was examined and rejected because the residual depends on the candidate.

Unit moments. $g_i = \sum_{e\in i} z_e r_e$ is a segmented sum over the CSR layout of $\mathbf{Z}$ (one kernel). Units without equations are zero rows and change nothing in (5).

A.5 Variances and specification tests

Robust sandwich. For a fixed weight A (efficient or not), with $D = \mathbf{Z}'\mathbf{X}/N$ and $B = (D'AD)^{-1}$,

$$\widehat{V} = \frac{1}{N}\, B\, D'A\, \Omega\, AD\, B, \tag{8}$$

with Ω rebuilt from the residuals at the reported estimate. It is valid for the one-step estimator too. This is the variance *conditional on* $\hat{\gamma}$.

Joint variance with the threshold. Following Seo and Shin (2016) the threshold is one more parameter. The Jacobian is augmented to $[\, D \mid \mathbf{Z}'x_\gamma/N \,]$, where x_γ is the derivative of the transformed residual with respect to γ, and the θ block of the sandwich of the augmented system is reported. In the jump model the sample moment is a step function of γ, so the derivative of its expectation is smoothed by a Gaussian kernel,

$$x_\gamma = -\,\mathcal{T}\Big[\varphi_h(q-\hat{\gamma})\,(\hat{\delta}_1 + x'\hat{\delta}_2)\Big], \qquad h = c\cdot 1.06\cdot s_q\cdot n^{-1/5}, \tag{9}$$

with $\varphi_h(u) = \varphi(u/h)/h$, s_q the standard deviation of q over all level observations, n the number of units with an equation and $c =$ `bwscale` (1.5 by default, as in `xthenreg`). In the kink model the derivative is analytic, $x_\gamma = -\hat{\delta}_q\,\mathcal{T}\,\mathbf{1}(q > \hat{\gamma})$. The variance conditional on $\hat{\gamma}$ is kept (`e(V_cond)`); if the joint variance cannot be computed, the conditional one is reported (`e(joint_vce)` $= 0$). The upper-regime slopes are $\hat{\beta} + \hat{\delta}_2$, with the variance of the linear combination.

Windmeijer correction. $W_2 = \Omega(\theta_1)^{-1}$ depends on the first-step estimate. Linearising the second-step estimate in θ_1 (Windmeijer 2005) adds $D_j = -B_2 G_2' W_2\,[\partial\Omega/\partial\theta_j]\,W_2\bar{g}_2$, with $B_2 = (G_2'W_2G_2)^{-1}$, $G_2 = D(\hat{\gamma})$, $\bar{g}_2$ the second-step moment residual and B_1, G_1 the first-step counterparts. The variance is

$$\widehat{V}_W = \frac{1}{N}\, M\, \Omega_1\, M', \qquad M = B_2G_2'W_2 + [D_1, \ldots, D_p]\; B_1G_1'W_1. \tag{10}$$

The derivative of Ω is never formed. With $h_{ij} = \sum_{e\in i} z_e x_{ej}$ and $s = \sum_i g_i$, only the product $[\partial\Omega/\partial\theta_j]\,v = -\frac{1}{N}\sum_i (h_{ij}\,g_i'v + g_i\,h_{ij}'v) + \frac{1}{nN}(h_j\,s'v + s\,h_j'v)$ with $v = W_2\bar{g}_2$ is needed (the last term comes from the centring), which costs two products per column instead of an $L \times L$ matrix. The correction is computed once at the final estimate; if it cannot be computed or is not positive semidefinite, the robust variance is reported (`e(vce_applied)` $= 0$). Windmeijer derived it for moments linear in the parameters; it need not improve the variance for the nonlinear moments of the threshold.

Hansen's J. $J = Q(\hat{\theta}, \hat{\gamma})$ of the second step, referred to χ^2 with $L - p - 1$ degrees of freedom (the threshold counts as one parameter). Nothing is reported after the one-step fallback or when the model is not over-identified.

Arellano–Bond serial correlation. On the *first-difference* residuals, whatever the estimation transformation (the `xtabond2` convention), the statistic of order m is $\mathrm{AR}(m) = b_0/\sqrt{T_1 + T_2 + T_3}$ (Arellano and Bond 1991, eq. 8), where, with r_{it} the residuals and $u_i = \sum_t r_{it}r_{i,t-m}$, $b_0 = \sum_i u_i$, $T_1 = \sum_i u_i^2$, $T_2 = -2\,g'(MAM')^{-1}MA\,s$ and $T_3 = g'\widehat{V}g$. Here $g = \sum_e x_e^* r_{e,t-m}$ collects the regressors against the lagged residual, $s = \mathbf{Z}'(r \circ u_{\text{unit}})$ the instrument moments weighted by the unit sums, $M = \mathbf{Z}'\mathbf{X}$ (raw sums) and $\widehat{V}$ the variance of the parameters. With the joint variance the threshold column x_γ is a parameter of g, M and $\widehat{V}$ (`e(ar_joint)` $= 1$); the statistics that treat $\hat{\gamma}$ as known are `e(ar1_cond)` and `e(ar2_cond)`. The p-value is $2\Phi(-|\mathrm{AR}|)$. At least five residual pairs, from at least five units, are needed.

A.6 Bootstrap inference

The wild bootstrap of the command. Every bootstrap component compares, on each sample, the criterion of a *restricted* model with the smallest criterion of the threshold models over the initial grid. A sample is $Y_b^* = F + r\,\eta_{i(e),b}$: a fitted part F and the residuals r of the restricted fit, multiplied by one weight η_{ib} per unit (so the correlation within a unit and the heteroskedasticity across units are kept), the same weights for every candidate. Regressors and instruments keep their sample values. The weights follow Mammen (1993): $-\phi$ with probability $(\sqrt{5}+1)/(2\sqrt{5})$ and $1/\phi$ otherwise, $\phi = (\sqrt{5}-1)/2$; mean 0, variance 1.

Why it is cheap. With a fixed weight W the criterion of a candidate g depends on the sample only through the moments $m^* = \mathbf{Z}'Y^*/N$:

$$\theta = C_g\, m^*, \quad C_g = (D_g' W D_g)^{-1} D_g' W, \qquad J_g(m^*) = N\, m^{*\prime}\, [M_g' W M_g]\, m^*, \quad M_g = I - D_g C_g. \tag{11}$$

The moments of all samples are $m_b^* = \mathbf{Z}'F/N + N^{-1}\sum_i \eta_{ib}\, g_i$, where $g_i = \sum_{e\in i} z_e r_e$ is the unit moment of the residuals. No $N \times B$ matrix is ever formed: for G candidates and B draws, all the criteria are a few batched products of the G matrices $Q_g = M_g' W M_g$ ($L \times L$) with the $L \times B$ matrix of moments, in chunks that bound the memory (8 million elements for the work array; the Q_g are kept when they fit in 40 million elements and the estimates are solved draw by draw otherwise). The bootstrap is 80–92 % of the time of a full run; it is evaluated this way for all draws and candidates at once.

Confidence set (grid bootstrap). Following Hansen (1999b) and Gong and Seo (2026, §4.1), the set collects the candidates γ_ℓ that a bootstrap test does not reject. With the weight of the reported estimator (W_2 after a two-step fit, W_1 after the fallback), for each candidate: (1) restricted fit $\hat\theta(\gamma_\ell)$ and residuals e; (2) statistic $D_n(\gamma_\ell) = Q(\hat\theta(\gamma_\ell), \gamma_\ell) - m_n$, where m_n is the minimum of $Q(\hat\theta(\gamma), \gamma)$ over $\Gamma \cup \{\gamma_\ell\}$ (so $D_n(\hat\gamma) = 0$); (3) draws $Y_b^* = \mathbf{X}(\gamma_\ell)\hat\theta(\gamma_\ell) + \eta_{ib} e$; (4) $D_{n,b}^*(\gamma_\ell) = J_\ell(m_b^*) - \min\{J_\ell(m_b^*), \min_{g\in\Gamma} J_g(m_b^*)\}$; (5) accept γ_ℓ iff $D_n(\gamma_\ell)$ does not exceed the order statistic of rank $k = \lceil p\,(B+1)\rceil$ of the $D_{n,b}^*$, with $p =$ `level`$/100$. That is the add-one rule: the bootstrap p-value $(1+R)/(1+B)$, $R = \#\{D^* \geq D\}$, exceeds $1-p$. If $k > B$ no draw can reject and every candidate is accepted. The set is the union of the accepted candidates; `e(ci_nseg)` counts its segments and `e(gamma_lo)`, `e(gamma_hi)` are the hull, which contains rejected values when there are several segments. A candidate whose restricted fit cannot be solved is not admissible (status 3); a draw that gives no valid statistic leaves the candidate unresolved (statuses 4–6), and then no set is reported (`e(ci_incomplete)` $= 1$). A statistic exactly zero is accepted mechanically (status 2).

Linearity test. $H_0 : \delta = 0$. The statistic is the one-step criterion of the linear model minus the smallest one-step criterion of the threshold model over the initial grid, both with W_1; samples are drawn from the linear fit; the p-value is $(1 + R)/(1 + B)$, and is missing if any draw is invalid. This GMM distance test differs from the sup-Wald test of `xthenreg`.

Test of a given threshold. `citest(#)` is one row of the grid bootstrap at $\gamma = \#$, with its own seed, so asking for it changes no other result; its p-value is $(1 + R)/(1 + B)$ and its decision is $D \leq$ crit.

Continuity test. Following Gong and Seo (2026, §3.2 and §4.3) with $\tilde\theta = \arg\min_{\Theta_c} \hat Q_n$ the continuity-restricted estimator, the statistic is the distance $T_n = N(Q(\tilde\theta) - Q(\hat\theta))$ between the kink and the jump criteria. The package computes it over the initial grid with the common weight, draws samples from the kink fit with the *jump* residuals reweighted by unit, and takes the p-value $(1+R)/(1+B)$ of $T_n^* = \min_{\text{kink}} J^* - \min_{\text{jump}} J^*$. It needs q as a contemporaneous regressor, otherwise the kink model is not nested in the jump model.

Coefficient bootstrap (legacy). `coef_boot` re-runs the whole search in each draw: samples $Y^* = \mathbf{X}(\hat\gamma)\hat\theta + r\eta$, the threshold searched again over the estimation grid with W_1, θ re-estimated there (two-step: W_2^* from the draw's residuals, threshold searched again). The intervals are $\hat\theta_j \pm q_{1-\alpha}(|\theta_j^* - \hat\theta_j|)$ (symmetric) or the percentiles $q_{\alpha/2}, q_{1-\alpha/2}$ of θ_j^*. Section A.7 says why this is not the coefficient bootstrap of Gong and Seo.

Unit resampling. `boot_type="unit"` follows the structure of Algorithm 1 (Section A.7). A resample is a vector of unit counts, so every quantity is a sum over units of per-unit cross products that do not depend on the draw: $\mathbf{Z}'\mathbf{X}(\gamma_g)$ of each grid point, $\mathbf{Z}'y$ and the unit moments; a draw costs a few small products and no pass over the N equations. Per draw: the first-step coefficients of every grid point with W_1 of the sample, the best one giving the residuals whose recentred cluster covariance defines $W_2^* = \Omega^{*-1}$; then the criterion at the candidate (restricted) and at every grid point (unrestricted) with W_2^*. A draw whose profile has no usable minimum is skipped, not replaced.

Random numbers. Stata 17 uses the 64-bit Mersenne Twister; `set seed s` starts it with `init_genrand64(s)`, and a draw is $u = ((x \gg 11)\,|\,1) \cdot 2^{-53}$, where x is the next 64-bit output (measured on 10 000 draws from five seeds: identical). A matrix is filled row by row; `uniform()`, `runiform()` and `invnormal(uniform())` consume one stream. The package reproduces the stream, so the same seed gives the same weights as the command. With `rseed(s)` each component restarts at $(s + \text{offset}) \bmod 2^{31}$:

confidence set 0, linearity 104 729, continuity 224 737, coefficient 350 377, `citest` 477 377. The weights of a component are drawn replication after replication, one uniform per contributing unit. Without a seed one random seed is drawn and the components share a stream in the order they run. The Python wrapper always records a seed (Section A.9).

A.7 Gong and Seo's Algorithm 1 and what the package does

This section is the reading of Gong and Seo (2026), Algorithm 1 and §§4.1–4.3 (Theorems 5–7), against the code. The proofs were not re-derived; the comparison is of the algorithms, and the statements about validity are the paper's.

The algorithm of the paper. For a parameter $\theta_0^* = (\beta_0^{*\prime}, \delta_0^{*\prime}, \gamma_0^*)'$ that generates the bootstrap data:

1. draw i^* i.i.d. uniformly from $\{1, \dots, n\}$ and resample the units *jointly*: $(x_{it}^*, x_{it-1}^*, z_{it}^*, \widehat{\Delta\epsilon}_{it}^*) = (x_{i^*t}, x_{i^*t-1}, z_{i^*t}, \widehat{\Delta\epsilon}_{i^*t})$, with the *unrestricted* residual $\widehat{\Delta\epsilon}_{it} = \Delta y_{it} - \Delta x_{it}'\hat\beta - \mathbf{1}_{it}(\hat\gamma)' X_{it}\hat\delta$;
2. generate $\Delta y_{it}^* = \Delta x_{it}^{*\prime}\beta_0^* + \mathbf{1}_{it}^*(\gamma_0^*)' X_{it}^*\delta_0^* + \widehat{\Delta\epsilon}_{it}^*$;
3. $g_{it}^*(\theta) = z_{it}^*(\Delta y_{it}^* - \Delta x_{it}^{*\prime}\beta - \mathbf{1}_{it}^*(\gamma)' X_{it}^*\delta)$;
4. recentre: $\bar g_n^*(\theta) = \frac{1}{n}\sum_i (g_i^*(\theta) - \bar g_n(\hat\theta))$, with $\bar g_n(\hat\theta)$ the sample moment of the unrestricted residuals;
5. $\hat\theta_{(1)}^* = \arg\min \bar g_n^{*\prime}\bar g_n^*$ (identity weight) and $W_n^* = \left(\frac{1}{n}\sum g_i^* g_i^{*\prime} - \bar g^* \bar g^{*\prime}\right)^{-1}$ at $\hat\theta_{(1)}^*$;
6. $\hat Q_n^*(\theta) = \bar g_n^*(\theta)' W_n^* \bar g_n^*(\theta)$ and the bootstrap statistic.

Three choices of θ_0^* give three procedures. The grid bootstrap of the threshold takes $\theta_0^* = (\hat\alpha(\gamma_\ell)', \gamma_\ell)'$ (the restricted estimate at the candidate) and $D_n^*(\gamma_\ell) = n(\min_\alpha \hat Q_n^*(\alpha, \gamma_\ell) - \min_\theta \hat Q_n^*(\theta))$; the set is $\{\gamma \in \Gamma_n : D_n(\gamma) \le \hat F^{*-1}(1-\tau; D_n^*(\gamma))\}$ (eq. 7), which is never empty because $D_n(\hat\gamma) = 0$. Theorem 5: at $\gamma = \gamma_0$, $D_n^* \to Z_0^2$ with $Z_0 = \max(0, Z_0^*)$ if the model is continuous and $\to \chi_1^2$ if it is not, so the set is valid whether or not the function is continuous. The coefficient bootstrap takes $\theta_0^* = w_n\hat\theta + (1-w_n)\tilde\theta$ with $w_n = \min\{T_n/(\hat C n^{1/4}), 1\}$ and the continuity-restricted estimator $\tilde\theta$ (eq. 8), so that the bootstrap Jacobian degenerates fast enough under continuity; the intervals are eq. (9) (percentile) or eq. (10) (symmetric), Theorem 6, pointwise only. The continuity test takes $\theta_0^* = \tilde\theta$ and $T_n^* = n(\min_{\Theta_c} \hat Q_n^* - \min_\Theta \hat Q_n^*)$ (Theorem 7).

Comparison.

Step	Gong and Seo, Algorithm 1	`boot_type="wild"` (default)
Resampling	units drawn with replacement; x, z and the residual move together	one Mammen weight per unit multiplies the residuals; x and z keep their sample values
Data-generating process	θ_0^* and *unrestricted* residuals	restricted fit at the candidate and *restricted* residuals; $\mathrm{E}^*[\eta] = 0$, so the null holds without recentring
Recentring	subtract $\bar g_n(\hat\theta)$	not needed (restricted residuals, mean-zero weights)
Weight in the bootstrap criterion	re-estimated in every draw from a first step with the identity	the sample weight, held fixed (W_2, or W_1 after the fallback)
Validity	Theorem 5 (continuous or not)	not covered by the theorem; coverage is examined by simulation (Nguyen and Lai 2026)

`boot_type="unit"` follows the structure instead: whole units resampled (as count vectors), restricted coefficients $\hat\theta(\gamma_\ell)$ with the *unrestricted* residuals at $(\hat\theta, \hat\gamma)$, moments recentred at the sample moment of those residuals, and W_2^* re-estimated in every draw. One difference is deliberate in the Stata command and copied: the first step of each draw uses the sample first-step weight W_1 where the paper uses the identity. It is available for `method="fd"` without `kink` or `td`, and is much slower. An unbalanced effective stack takes the command outside the common-T setting of the theory (Section A.11 lists the warning that Stata prints and the package does not).

The coefficient bootstrap is not the paper's. The legacy `coef_boot` generates data from $\mathbf{X}(\hat\gamma)\hat\theta$ with reweighted residuals, i.e. $\theta_0^* = \hat\theta$, which is the choice $w_n = 1$ of eq. (8). That is the right choice when the model is discontinuous; under continuity the paper shows that a bootstrap built on $\hat\theta$ is not valid, which

is why it proposes w_n. The command therefore calls it a legacy compatibility feature, `coef_boot="gs"` is refused, and the package says so in a note when it is used. The symmetric interval is eq. (10) with $\theta_0^* = \widehat{\theta}$.

What is open. The comparison above is a reading, not a proof of agreement. The power of the rejection of shifted thresholds in the package's Monte Carlo experiments on the designs of Gong and Seo is lower than their Table 2 while the coverage of the threshold agrees, and with the self-exciting designs of Seo and Shin (2016) the dispersion of the estimated threshold is larger than in their tables, although the extreme estimates are identical to Stata's, so they come from the estimator. The cause is not established.

A.8 `xthenreg`

Estimator. The balanced-panel estimator of Seo, Kim and Kim (2019) is the first-difference special case of Section A.3: equation periods $t = 2, \dots, T$, a block of instruments per period (constant, Δx_t of the plain regressors, the levels $y_1, \dots, y_{t-2}$, the levels of the endogenous regressors up to $t-2$, the external instruments), the coefficient order $[\beta \,|\, \delta_1 \,|\, \delta]$ for the jump and $[\beta \,|\, \kappa]$ for the kink, the first-step MA(1) weight built block by block ($W_1 = (2S_0 - S_1 - S_1')^{-1}$ with S_0 block diagonal, $(1/N)Z_p'Z_p$, and S_1 block upper diagonal, $(1/N)Z_{p-1}'Z_p$), and the second-step weight from the centred covariance of the unit moments. The grid is uniform between two `mm_quantile` bounds at `trim_rate`/2 and 1−`trim_rate`/2 over all the values of q (default `trim_rate` 0.4) and $\hat{\gamma}$ is the first minimum of the second-step criterion; both steps search the grid, the second step with the weight built at the first-step estimate.

Covariance. Following Seo and Shin (2016, p. 173), with $G = (G_b, G_d, G_r)$ the derivatives of the moments with respect to the linear coefficients, the regime coefficients and the threshold,

$$\widehat{V} = \frac{1}{N} \left(G' \, \Omega^{-1} \, G \right)^{-1}, \tag{12}$$

with the derivative in γ computed by the same Gaussian kernel as in (9), bandwidth $h = h_0 \cdot 1.06 \cdot N^{-1/5} \cdot \mathrm{sd}(q)$ (`h_0(1.5)`), in the jump model, and in closed form for the kink. The 95 % interval of every coefficient, the threshold included, is the estimate ± 1.959963984540054 se.

Linearity test. The Wald statistic of the regime coefficients is computed at every grid point, each time with its own two-step estimate and covariance, and the statistic is the supremum over the grid (`e(sup_wald)` in the package). The p-value is from the fast bootstrap of the paper: the step-two moments of each unit are multiplied by a standard normal weight (`invnormal(uniform())`), the one-step estimator is re-evaluated with the fixed step-two weight at every grid point, and the Wald form reuses the covariance matrices of the sample. The p-value is the *share* of bootstrap suprema above the sample supremum, not the add-one rule of `xtdpthresh`, so it can be exactly zero; a missing bootstrap statistic counts as exceeding.

Against `xtdpthresh`. Under `method(fd)` the two commands use the same moment conditions and weight matrices and, on the same grid, give the same estimates; the differences are in the trimming default, the placement of q, the exogenous-regressor instruments (levels in `xthenreg`, transformed values in `xtdpthresh`), the refinement of the kink, the absence of a standard error for $\hat{\gamma}$ and the GMM-distance linearity test of `xtdpthresh`. Where the two designs coincide (cases c1, c4 and c5 of the validation) the package checks that the two ports agree.

A.9 Conventions of the Stata commands that are reproduced

Each row is a behaviour of the Stata code that the package copies so that the numbers agree. The ones marked `STATA-QUIRK` in the source are the ones where the behaviour is surprising.

Convention	What is reproduced and why
`invsym`	Reads only the upper triangle of its argument; a sweep with the pivot on the largest remaining diagonal element; a pivot is dropped when it is at most 10^{-9} times the *original* diagonal element, at most 9×10^{-20}, or negative, and its row and column of the result are zero (a generalised inverse). Measured on Stata 17 MP. For a well-conditioned positive definite matrix no pivot can be dropped in any order, so a Cholesky factorisation replaces the sweep.
`cholinv`	Reads only the lower triangle; missing if the factorisation fails or the reciprocal condition number is below 10^{-20}.

Convention	What is reproduced and why
`syminv` gate	`xtdpthresh` admits an inverse only after Jacobi equilibration, a Cholesky factorisation, a condition number of at most 10^{12} and a check that $CC^{-1} \approx I$. A matrix that fails is a candidate that is not admissible. Validated on 23 matrices against the Mata source run unchanged.
Tolerances of ties	Equal criteria (relative 10^{-12} for profiles, 10^{-10} for the refinement check) go to the smaller threshold (`objtol`).
W_2 built once	The second-step weight is built at the first-step threshold and held fixed over the grid of the second step.
Equal splits, exact zero	Two candidates with the same split of the observed values of q give the same regressors; the statistic of the estimated threshold must be exactly zero, so the moments of the initial-grid entry with the same split are reused instead of summing the arrays in another order (a gap of the order of rounding would otherwise remain).
Order statistic	The critical value is the order statistic of rank $\lceil p(B+1) \rceil$, which is the add-one p-value rule; infinite when the rank exceeds B (no draw can reject).
Random stream	Mersenne Twister 64, $u = ((x \gg 11) \mid 1)2^{-53}$, row-major fill, component seeds (s+offset) mod 2^{31}, one draw per contributing unit (a unit without equation takes no draw, so adding such a unit leaves the draws unchanged).
Unit bootstrap draws	A draw whose profile has no usable minimum is skipped, not replaced (`STATA-QUIRK`); the missing ones stay missing in the result.
`dlt_len` in `xthenreg`	With a kink the Wald block that is tested has *two* entries (the last linear coefficient and the kink coefficient) instead of the kink coefficient alone (`STATA-QUIRK`); kept so that the statistic equals Stata's.
Signs of G	With a kink G_d has the positive sign (unlike the jump model) and G_r uses the indicator of the regime above the threshold, as in `cov_mat_con_kink` (`STATA-QUIRK`).
`exogenous()` of `xthenreg`	An `exogenous()` regressor enters with its levels up to t, two more lags than an `endogenous()` one (`STATA-QUIRK`); hence 87 moment conditions where `xtdpthresh` has 60 instruments for the same data.
`boots_p`	The share of bootstrap suprema above the sample supremum, a missing one counting as exceeding, as a missing value does in Mata; -1 when the test is not run.
h and s_q	s_q is the sample standard deviation over *all* the level observations and n the units with an equation; the 1.959963984540054 of the interval is the constant typed in the `xthenreg` source.
AR on first differences	The serial-correlation tests use first-difference residuals under both transformations.
Time-effect zeroing	A column that the projection reduces to rounding noise (below 10^{-10} of its size) is set to exactly zero, as Stata does.
`if`, missing values	A missing value in the condition counts as *true* in Stata; in the package it counts as false, and the rows with a missing variable are removed afterwards by the missing-value check, so the sample is the same.
Seed (the package adds)	When the bootstrap is on and no seed is given, the wrapper draws one in $[1, 2^{31}-1)$ and records it (`res.seed`, the settings log, the JSON). Stata draws from the current generator and does not report it.
Precision	All values are `float64`; Stata's operators on `float` variables are recomputed in double precision in the command (`_xdpt_tsdouble`), which a double-precision array makes unnecessary.

A.10 Validation journal

The numbers of every comparison, by block, by case and by object, are in Appendix B.

Rule. Every number the package reports must equal Stata 17 MP to four decimals: absolute difference at most 5×10^{-5} *or* relative difference at most 10^{-6}. The comparison is automated and runs without Stata.

Method. The real commands (`xthenreg`, `xtdpthresh` 0.9.38) are run in batch from PowerShell; a Mata routine exports each object at 15 significant digits; the results are frozen in `tests/golden/` (one file per component) with the versions of Stata and of the commands. The test suite compares the package with them case by case. The random numbers of Stata are reproduced, so a seeded bootstrap draws the very weights of the command and its p-values and sets are compared directly, not through their distribution. Each case is described once and drives both the Stata run and the Python run.

Object	Cases	Result
Random numbers	5 seeds	10 000 draws each, identical; 40 bootstrap suprema of each of 9 `xthenreg` cases equal to a relative 2×10^{-10}
`invsym`, `cholinv`, `syminv`	measured; 23 matrices	thresholds measured on Stata; `syminv` equal to the Mata source run unchanged
`xthenreg`: L, y, q, x, Z	12	equal to 10^{-12} running the Mata source as is (step 1 of `maincode_0719`)
`xthenreg`: $\hat{\theta}$, $\hat{\gamma}$, V, CI, $e()$	c1–c8 and variants	b within 3×10^{-11} worst case on c1–c6; `h_0(3)` and `h_0(0.7)` (c7, c8) through the public function; grid × trimming variants (12 jump, 3 kink, 4 on $T = 15$ with $L = 325$)

Object	Cases	Result
`xthenreg`: sup-Wald, p	9; 5	statistic to a relative 5×10^{-10}; the p-values of `boost(100)` equal on 5 cases (static and dynamic, jump and kink)
`xtdpthresh`: sample selection	36	`e(N)`, `e(N_stack)`, `e(N_units)` and the exact set of rows of `e(sample)` equal; return codes of the refusals equal
Instruments Z, X	28 + 30 + 15	default, `maxlag`, `collapse`, `iv()`, duplicates, dependence, common variable; 15 cases of forward deviations
One step, two steps	28	`e(b)`, `e(obj)`, `e(V_cond)`; the one-step fallback is reproduced
Grid and search	29; 5; 3	`e(grid_admitted)`, `e(q_lo)`, `e(q_hi)`; kink refinement (`e(kink_refined)`); `refine` (iterations and points added)
Whole commands, full panel	14	the examples of the help file and variants on 565 firms: `e(b)`, `e(gamma)`, `e(obj)`, `e(N)`, `e(N_iv)`
Unbalanced panels, FOD, td, history	5 + 5; 7 + 8; 4	interior gaps, missing values, short units, alternate years; `td` under FD and FOD; `history(panel|sample)`
Variances	45; 23	`e(V)`, `e(V_cond)`, `e(V_upper)`, `e(b_upper)`, jump and kink, FD and FOD, `td`, unbalanced; `vce(windmeijer)`; `bwscale` (0.8, 3), `nocenter`
Tests	45	`e(hansen)`, `e(hansen_df)`, `e(hansen_p)`, `e(ar1)`, `e(ar2)` and their p-values
`predict`	11	residuals, `xb`, AR residuals, regime, `if`
Bootstrap	62 Stata cases, 287 checks	`e(pval_lin)` (56), `e(ci_grid)`, `e(gamma_lo)`, `e(gamma_hi)`, `e(ci_nseg)`, `e(ci_segments)`; continuity (8); `citest` (8); coefficient bootstrap (10); `boottype(unit)`; refusals
Reporting	146 checks	the tables repeat the validated b, V, `ci_grid`, `e(b_upper)` exactly
Hostile inputs	19	minimal panels, a constant, binary or rare threshold, an empty regime, duplicated regressors, fewer units than instruments, a null outcome: the same return code as Stata on 19 of 19

Equivalence of the fast kernels. Each Numba kernel has a NumPy reference that is kept; the tests compare them to 10^{-12} (several seeds, ties between q and the grid, a missing q, degenerate grids, one against four threads bit for bit, and with the compiler switched off). The optimisations of the bootstrap were accepted only when the reference cases remained identical.

Suite. About 2 170 tests (unit, golden, integration), line and branch coverage about 91.5 % (the compiled bodies cannot be traced by the coverage tool); the code of the README and of this guide is executed by the tests.

What is not compared with Stata. The statistical behaviour of the package's own seeds (size and power of the linearity test, coverage of the confidence set over many panels) is measured by Monte Carlo and compared with the papers, not with Stata, and has the open points of Section A.7. Timings are measured on the same machine one run after the other, because the speed of a machine varies by up to a factor of two from one moment to the next. The statements of the guide about interpretation come from the Stata help files, the code and the execution of the package, and are not a second validation.

A.11 Where every Stata function went

The inventory is generated by `tools/ported_functions.py` from the `# Stata: function (file.ado:line)` comments of the source and the function ranges of `validation/stata_functions.json` (a manifest with the SHA-256 of the three `.ado` files); a test checks that every citation points inside a function. A function counts as *ported* when at least one comment points inside it. At the time of writing, 86 of the 101 functions of the three files are cited; the generated tables below list the cited ones and where they live. The 15 that are not cited fall in three groups.

Stata function	Status	Reason
`xdpt2_append_nested_level`, `xdpt2_nested_midpoints`, `xdpt2_profile_stability`	retired	Reachable only when `search_mode == "adaptive"`, which the command refuses since 0.9.26 (`search_mode`, `search_max` and `search_tol` are retired). The package evaluates the grid it is asked for; the nested grids $g \to 2g-1 \to 4g-3$ are reached by choosing `grid=199` or `397`.
`xdpt2_build_gamma_cache`	dead code	No caller in the command; the variant `xdpt2_build_gamma_cache_t` is the one in use and is ported (the cache of the candidates).
`xdpt2_tpl_build`, `xdpt2_tpl_dW`, `xdpt2_tpl_release`, `xdpt2_unit_cfg`	replaced	The Mata record of the stacked-equation template, its release and the settings read once per stack. The package keeps the transformation as the sparse matrix $\mathcal{T}$ (Section A.2) and the instrument settings as a dataclass, so none of this bookkeeping exists; the quantities they produce are the ones compared with Stata.

Stata function	Status	Reason
xdpt2_hasmiss	not needed	Tests for a missing value in a block; the package uses np.isfinite.
_xdpt_tsdouble	not needed	Recomputes the time-series operators in double precision (tsrevar keeps the type of the source); all values are float64.
xdpt2_rowsig2 (in xtdpthresh.ado and in xtdpthresh_p.ado), xdpt2_p_fill of xtdpthresh_p.ado	replaced	A checksum of the data and a keyed store that let predict detect a changed dataset or a cleared memory (return codes 459 and 498). The result object holds its data and predict(kind, frame) checks the keys of the frame it is given; the copy of xdpt2_p_fill in xtdpthresh.ado is ported.
xdpt2_is_strongly_balanced	message not ported	Used to print, when boottype(unit) runs on an unbalanced effective stack, "this is an experimental extension beyond the common-T theory", and to store a balance flag. The package does not print that warning; the estimates are unaffected.
xdpt2_near_common	message not ported	Names, in the error message of a rank failure, the variables that are common to all units only up to rounding. The package raises the same error with the same return code but without that hint.

The 19 functions that were not cited before this note was written are now cited where the package implements their role (for example xdpt2_gsum_by_unit by the unit-moment kernel, xdpt2_kink_joint_V and xdpt2_kink_joint_wind by the joint variance, xdpt2_fast_gmm_boot by the batched bootstrap criterion, FD_con, cov_con_b and cov_con_kink_b of xthenreg by the first differences and the Wald covariance of the linearity test).

Generated by tools/ported_functions.py inventory from the # Stata: comments of src/ and the function ranges of validation/stata_functions.json. A function is *ported* when at least one comment of the package points inside it.

A.11.1 xtdpthresh.ado — 63 of 76 functions referenced

Stata function	Lines	Ported in
_xdpt_coeftab	2319-2367	extras/reporting/summary.py
_xdpt_tsterm	2391-2466	data/operators.py
_xdpt_upper	2257-2318	models/xtdpthresh/coefficients.py
xdpt2_append_candidates	5275-5358	models/xtdpthresh/search.py
xdpt2_ar_full	8092-8260	models/xtdpthresh/diagnostics.py
xdpt2_ar_xg	5149-5190	models/xtdpthresh/stack.py
xdpt2_build_W_ma1	3808-3879	core/weights.py, models/xtdpthresh/stack.py
xdpt2_build_cluster_omega	4001-4011	models/xtdpthresh/estimator.py
xdpt2_build_cluster_omega_c	4012-4047	core/gmm.py
xdpt2_build_gamma_cache_t	4383-4619	core/grids.py, inference/bootstrap/objective.py, models/xtdpthresh/search.py, models/xtdpthresh/stack.py
xdpt2_build_units	2587-2689	models/xtdpthresh/sample.py
xdpt2_cache_w2	6647-6666	inference/bootstrap/objective.py
xdpt2_coef_bootstrap	6357-6633	models/xtdpthresh/bootstrap.py
xdpt2_component_seed	4796-4806	inference/bootstrap/draws.py, inference/bootstrap/rng.py
xdpt2_const_within	8261-8304	models/xtdpthresh/sample.py
xdpt2_continuity_test	7431-7767	models/xtdpthresh/bootstrap.py
xdpt2_crit_orderstat	4839-4888	inference/bootstrap/objective.py
xdpt2_demean_bytime	3142-3176	transforms/td.py
xdpt2_dense_uid	4740-4759	models/xtdpthresh/estimator.py
xdpt2_drop_cellconst	3304-3399	instruments/cleaning.py
xdpt2_fast_gmm_boot	4620-4667	inference/bootstrap/objective.py
xdpt2_fast_gmm_boot_w	6667-6696	inference/bootstrap/objective.py
xdpt2_fast_obj_split_list	4668-4739	inference/bootstrap/objective.py
xdpt2_find_t	2703-2720	core/calendar.py
xdpt2_gmm_sandwich	4048-4099	core/gmm.py
xdpt2_grid_bootstrap	6697-7430	models/xtdpthresh/bootstrap.py, models/xtdpthresh/unit_bootstrap.py
xdpt2_grid_search	5630-6356	models/xtdpthresh/estimator.py, models/xtdpthresh/search.py
xdpt2_gsum_by_unit	3957-4000	models/xtdpthresh/estimator.py
xdpt2_hansen_j	8048-8091	models/xtdpthresh/diagnostics.py
xdpt2_indep_cols	3177-3243	instruments/cleaning.py
xdpt2_jump_xg	5067-5120	models/xtdpthresh/stack.py
xdpt2_jump_xg_tp	5035-5066	models/xtdpthresh/stack.py

Stata function	Lines	Ported in
xdpt2_kink_apply_joint	5237-5256	models/xtdpthresh/variance.py
xdpt2_kink_joint_V	5191-5209	models/xtdpthresh/variance.py
xdpt2_kink_joint_wind	5210-5236	models/xtdpthresh/variance.py
xdpt2_kink_refine	5516-5629	models/xtdpthresh/search.py
xdpt2_kink_xg	4954-4986	models/xtdpthresh/stack.py
xdpt2_linearity_test	7768-8047	models/xtdpthresh/bootstrap.py
xdpt2_mammen_draw	4783-4795	inference/bootstrap/draws.py
xdpt2_n_switch	4274-4308	models/xtdpthresh/stack.py
xdpt2_nodup_cols	3244-3303	instruments/cleaning.py
xdpt2_objtol	2497-2515	core/linalg.py
xdpt2_p_fill	8349-8434	models/xtdpthresh/predict.py
xdpt2_profile_is_flat	5456-5469	models/xtdpthresh/search.py
xdpt2_profile_update	5386-5455	models/xtdpthresh/search.py
xdpt2_psd_ok	2554-2586	core/linalg.py
xdpt2_q_at_rows	2721-2743	models/xtdpthresh/search.py
xdpt2_q_nvals	5121-5148	models/xtdpthresh/variance.py
xdpt2_q_support	2744-2790	models/xtdpthresh/search.py
xdpt2_quantile	4807-4838	core/quantiles.py
xdpt2_quantile_grid	2467-2479	core/grids.py
xdpt2_rangen	2480-2496	core/quantiles.py
xdpt2_solve_gmm_1step	3880-3916	core/gmm.py
xdpt2_solve_gmm_1step_pre	3917-3956	core/gmm.py
xdpt2_stack_at_gamma	3452-3807	instruments/cleaning.py, models/xtdpthresh/stack.py, transforms/td.py
xdpt2_syminv	2516-2553	core/linalg.py
xdpt2_tpl_trans	4987-5034	models/xtdpthresh/stack.py
xdpt2_tpos	2690-2702	core/calendar.py
xdpt2_transform_unit	2801-3141	instruments/builder.py, models/xtdpthresh/stack.py, transforms/fd.py, transforms/fod.py
xdpt2_w2_solvable	6634-6646	inference/bootstrap/objective.py
xdpt2_windmeijer	4889-4953	models/xtdpthresh/variance.py
xtdpthresh	5-2256	models/xtdpthresh/bootstrap.py, models/xtdpthresh/coefficients.py, models/xtdpthresh/design.py, models/xtdpthresh/fit.py, models/xtdpthresh/options.py, models/xtdpthresh/sample.py
xtdpthresh_run	8435-11202	core/grids.py, instruments/builder.py, models/xtdpthresh/bootstrap.py, models/xtdpthresh/diagnostics.py, models/xtdpthresh/fit.py, models/xtdpthresh/search.py, models/xtdpthresh/variance.py

Not referenced by any comment (13): _xdpt_tsdouble, xdpt2_append_nested_level, xdpt2_build_gamma_cache, xdpt2_hasmiss, xdpt2_is_strongly_balanced, xdpt2_near_common, xdpt2_nested_midpoints, xdpt2_profile_stability, xdpt2_rowsig2, xdpt2_tpl_build, xdpt2_tpl_dW, xdpt2_tpl_release, xdpt2_unit_cfg

A.11.2 xtdpthresh_p.ado — 1 of 3 functions referenced

Stata function	Lines	Ported in
xtdpthresh_p	49-218	models/xtdpthresh/predict.py

Not referenced by any comment (2): xdpt2_p_fill, xdpt2_rowsig2

A.11.3 xthenreg.ado — 22 of 22 functions referenced

Stata function	Lines	Ported in
FD_con	885-895	models/xthenreg/moments.py
GMM_W_n_2_con	1293-1316	core/gmm.py, core/weights.py
GMM_W_n_con	1110-1174	core/weights.py
GMM_W_n_static	1349-1414	core/weights.py
GMM_cal	1175-1207	core/gmm.py, models/xthenreg/moments.py
GMM_cal_kink	1415-1453	models/xthenreg/moments.py
GMM_var_con	1014-1109	models/xthenreg/moments.py
Z_mat_IV_add	1454-1531	models/xthenreg/design.py

Stata function	Lines	Ported in
`Z_mat_con`	896-1013	`models/xthenreg/design.py`
`cov_con_b`	2034-2079	`models/xthenreg/linearity.py`
`cov_con_kink_b`	2080-2117	`models/xthenreg/linearity.py`
`cov_mat_con`	1532-1593	`models/xthenreg/covariance.py`
`cov_mat_con_kink`	1594-1635	`models/xthenreg/covariance.py`
`ep_cal`	1208-1249	`models/xthenreg/moments.py`
`ep_cal_kink`	1250-1292	`models/xthenreg/moments.py`
`grid_con`	870-884	`core/grids.py`
`index_non_ms`	1636-1682	`models/xthenreg/design.py`
`maincode_0719`	321-869	`models/xthenreg/design.py`, `models/xthenreg/estimate.py`, `models/xthenreg/estimator.py`, `models/xthenreg/linearity.py`
`moment_mat_con`	1317-1348	`models/xthenreg/design.py`
`sup_wald_cal`	1683-1919	`models/xthenreg/linearity.py`
`sup_wald_fb`	1920-2033	`models/xthenreg/linearity.py`
`xthenreg`	21-320	`models/xthenreg/estimate.py`

Not referenced by any comment (0): none

References

Arellano, M., and S. Bond (1991). Some tests of specification for panel data: Monte Carlo evidence and an application to employment equations. *Review of Economic Studies* 58: 277–297.

Arellano, M., and O. Bover (1995). Another look at the instrumental variable estimation of error-components models. *Journal of Econometrics* 68: 29–51.

Gong, W., and M. H. Seo (2026). Bootstraps for dynamic panel threshold models. *Journal of Econometrics* 253: 106153.

Hansen, B. E. (1999a). Threshold effects in non-dynamic panels: estimation, testing, and inference. *Journal of Econometrics* 93: 345–368.

Hansen, B. E. (1999b). The grid bootstrap and the autoregressive model. *Review of Economics and Statistics* 81: 594–607.

Mammen, E. (1993). Bootstrap and wild bootstrap for high dimensional linear models. *Annals of Statistics* 21: 255–285.

Nguyen, D. C., and N. D. Lai (2026). xtdpthresh: dynamic panel threshold regression for unbalanced panels, with endogenous regressors and continuity-robust inference. SSRN Working Paper 6619058.

Roodman, D. (2009). How to do xtabond2: an introduction to difference and system GMM in Stata. *Stata Journal* 9: 86–136.

Seo, M. H., S. Kim, and Y.-J. Kim (2019). Estimation of dynamic panel threshold model using Stata. *Stata Journal* 19: 685–697.

Seo, M. H., and Y. Shin (2016). Dynamic panels with threshold effect and endogeneity. *Journal of Econometrics* 195: 169–186.

Windmeijer, F. (2005). A finite sample correction for the variance of linear efficient two-step GMM estimators. *Journal of Econometrics* 126: 25–51.

B Validation against Stata 17

This appendix gives the largest deviation of the package (version 0.0.1) from Stata 17 by block, by case and by object, over every comparison of the golden tests and of the stress test. It was built on 04 November 2026 by `python -m validation.report`.

B.1 Rule and method

Every number the package reports must equal the one computed by Stata 17 to four decimals: the absolute difference is at most $5e-05$ *or* the relative difference is at most $1e-06$ (the relative branch keeps the rule meaningful for large values such as the objective function). The *margin* of a value is $\min(|\Delta|/5e-05,\ (|\Delta|/|x|)/1e-06)$; it is at most 1 when the value agrees, and the tables report the largest margin of each block.

The references are produced by the real commands (xthenreg (Seo, Kim and Kim 2019) and xtdpthresh 0.9.38) run in batch with Stata 17.0 MP; each object is exported at 15 significant digits and frozen in `tests/golden/`. The test suite compares the package with the frozen values, and the deviations in this report are recorded by those very comparisons (4\,764 comparisons of matrices or numbers in the golden tests, plus the objects of the help-file examples). The random numbers of Stata are reproduced exactly, so the seeded bootstraps draw the weights of the command and their p-values and confidence sets are compared directly.

B.2 Summary by block

Block	Cases	Objects	Values	Largest $\lvert\Delta\rvert$	Margin
Random numbers (Stata's Mersenne Twister)	5	1	10 500	1.3×10^{-15}	2.7e-11
Matrix routines (invsym cholinv syminv)	22	5	1 210	4.6×10^{5}	0.097
Quantiles and threshold grids	5	9	1 930	3.7×10^{-13}	3.7e-09
Data layer: xtset lags missing values	3	2	113 223	0	0
Time-series operators	19	19	4 838	5.6×10^{-17}	1.1e-12
xthenreg: data and instruments	12	14	44 849	2.3×10^{-13}	5.1e-10
xthenreg: sup-Wald statistic and linearity test	10	4	2 309	6.3×10^{-5}	0.00019
xthenreg: covariance and confidence intervals	24	4	2 324	1.4×10^{-12}	2.8e-08
xthenreg: complete result	32	6	2 916	4.1×10^{-11}	8.1e-07
xthenreg: bandwidth multiplier h_0() through the public function	2	3	260	3.7×10^{-13}	7.3e-09
xtdpthresh: instruments and fixed-threshold GMM	28	1	962 700	5.1×10^{-15}	1e-10
xtdpthresh: threshold search and final estimate	29	6	1 537	9.5×10^{-10}	1.6e-06
xtdpthresh: point estimates on the full panel	14	4	1 064	1.1×10^{-9}	2.1e-05
xtdpthresh: deviations time effects unbalanced panels history	41	7	1 508 791	7.6×10^{-9}	0.00011
xtdpthresh: variances and diagnostics	45	32	7 244	1.1×10^{-9}	2.1e-05
xtdpthresh: predict	12	6	66 718	3.3×10^{-13}	6.6e-09
xtdpthresh: bootstraps confidence set tests	56	29	14 538	8.9×10^{-8}	8.2e-05
xtdpthresh: unbalanced panel of the help file (no bootstrap)	2	4	184	6.6×10^{-14}	1.3e-09
xtdpthresh: hostile inputs	4	1	24	5.9×10^{-8}	0.0012
Simulated panels of the Seo-Shin design	4	1	20	4.5×10^{-9}	8.9e-05
xtdpthresh: the examples of the help file (default options)	8	72	503	2.1×10^{-7}	0.0014
xtdpthresh: 120 random scenarios (stress test)	97	76	5 286	2.9×10^{-5}	8.5e-05
All blocks	**474**		**2 752 968**	4.6×10^{5}	**0.097**

Cases are the distinct designs or samples of a block; *Objects* are the distinct names compared (for example `e(b)`); *Values* counts every number compared, a matrix counting for all its elements. The absolute deviation is large only where the values are (the matrix routines are tested on very large and very small matrices); the margin says whether the rule holds. Integer results (counts of equations, instruments, return codes of the refusals, sets of rows) are compared for exact equality and are not listed in the deviations: the tests fail on the first difference.

B.3 The examples of the help file

The eight commands of the help file of `xtdpthresh` were run as printed, with the default options of Stata (`boot(299)`, `gridci(100)`, `grid(100)`), on Hansen's investment panel without the five extreme firms. The tables of the values, Python against Stata, are in `docs/help_examples.md`; the largest deviation of each kind of object is below.

Example	Objects	Compared	Largest $\lvert\Delta\rvert$
Point estimates and diagnostics	coefficient	9	1.3×10^{-14}
	std. error	9	1.2×10^{-14}
	scalar	30	9.2×10^{-14}
	matrix	5	1.6×10^{-14}
Full inference and the test of Hansen's threshold	coefficient	9	1.3×10^{-14}
	std. error	9	1.2×10^{-14}
	scalar	45	2.3×10^{-11}
	matrix	7	4.9×10^{-9}
Instruments from lag 2	coefficient	9	1.8×10^{-9}
	std. error	9	5.9×10^{-9}
	scalar	45	1.0×10^{-9}
	matrix	7	2.1×10^{-7}
Kink model	coefficient	5	3.0×10^{-12}
	std. error	5	8.0×10^{-11}
	scalar	45	2.7×10^{-12}
	matrix	7	6.4×10^{-10}
Unbalanced panel, first differences	coefficient	9	3.3×10^{-14}
	std. error	9	6.7×10^{-13}
	scalar	45	1.6×10^{-13}
	matrix	7	3.3×10^{-10}
Unbalanced panel, forward orthogonal deviations	coefficient	9	3.8×10^{-10}
	std. error	9	7.5×10^{-10}
	scalar	45	9.2×10^{-11}
	matrix	7	1.7×10^{-8}
External instrument	coefficient	9	1.2×10^{-9}
	std. error	9	3.1×10^{-9}
	scalar	30	1.7×10^{-9}
	matrix	5	1.0×10^{-7}
Postestimation: residuals and regime	coefficient	9	1.3×10^{-14}
	std. error	9	1.2×10^{-14}
	scalar	30	9.2×10^{-14}
	matrix	5	1.6×10^{-14}
	predict	2	2.7×10^{-14}

B.4 The stress test

120 panels and calls were drawn at random (units, periods, share of missing periods and of missing values, a normal, rounded, heavy-tailed, outlying or four-valued threshold variable, the roles of the regressors, `method`, `kink`, `td`, `collapse`, `maxlag`, the grid and the bootstrap) and run in Stata and in the package on the same numbers (`validation/stress.py`, `validation/stress_cases.py`). The panels are read by Stata with `asdouble`: by default `import delimited` stores 32-bit floats, which moves a weakly identified estimate by 10^{-6}.

- 97 scenarios are estimated by both and agree on every object of the result (the objects are those of the help-file examples; see the summary table).
- 21 are refused by both with the same return code (498: no admissible threshold, an ill-conditioned instrument matrix…).
- 2 are degenerate and decided by rounding noise, and are listed in the test as expected to differ: with fewer units than instruments the criterion is exactly zero over part of the grid, so the extra candidate of the confidence set is the first of tied minima; with fewer equations than instrument columns the

dependence rule of the command keeps or drops a column on a pivot of the order of 10^{-13}, its own limit. The command documents this limit in its source.

Without Stata the same panels also check that the package never crashes (only its own exceptions, with a Stata return code), that every output works (summary, tables in five formats, JSON, plot, prediction), that a result does not change when the rows are shuffled, the units renamed or the time origin shifted, that the threshold follows an affine change of the threshold variable, that one or several BLAS threads give the same numbers, and that other containers of the same data (a `MultiIndex`, categorical or string identifiers, nullable or 32-bit numbers, accented names) give the same result while damaged inputs are refused with a message.

B.5 Largest deviation by object

The objects are named as the tests name them (`e(b)`, `e(V)`, `e(ci_grid)`…). A block appears here when it compares several named objects over several cases; the other blocks compare one kind of value, or one object per case, and are described by the table of their cases in the next section.

Matrix routines (invsym, cholinv, syminv)

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`cholinv`	14	384	4.6×10^{5}
`invsym`	21	439	1.2×10^{-1}
`psd`	22	22	0
`syminv`	13	343	4.6×10^{5}
`syminv_ok`	22	22	0

Quantiles and threshold grids

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`adm`	5	500	0
`minreg`	5	10	0
`mm`	5	140	4.6×10^{-14}
`qgrid`	5	130	7.8×10^{-14}
`qgrid1`	5	5	1.8×10^{-15}
`rangen`	5	500	5.0×10^{-14}
`rangen1`	5	5	1.8×10^{-15}
`rangen_x`	5	500	5.7×10^{-14}
`t7`	5	140	3.7×10^{-13}

Data layer: xtset, lags, missing values

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`balance`	3	3	0
`rows`	3	113 220	0

xthenreg: data and instruments

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`Z.bilinear`	12	12	1.1×10^{-13}
`Z.cols`	12	6 668	2.3×10^{-13}
`Z.rows`	12	1 140	1.1×10^{-13}
`moment`	12	453	0
`q.bilinear`	12	12	0
`q.cols`	12	105	0
`q.rows`	12	6 668	0
`scal`	12	60	0
`x.bilinear`	12	12	0

Object	Cases	Values	Largest $\lvert\Delta\rvert$
x.cols	12	105	0
x.rows	12	22 829	0
y.bilinear	12	12	0
y.cols	12	105	0
y.rows	12	6 668	0

xthenreg: sup-Wald statistic and linearity test

Object	Cases	Values	Largest $\lvert\Delta\rvert$
boot_sup	9	360	6.3×10^{-5}
cov_wald	9	1 900	1.4×10^{-6}
sup_wald	9	9	3.5×10^{-5}
value	1	40	2.8×10^{-14}

xthenreg: covariance and confidence intervals

Object	Cases	Values	Largest $\lvert\Delta\rvert$
CI	16	272	1.4×10^{-12}
V	16	1 232	1.6×10^{-13}
b	16	136	8.8×10^{-13}
value	8	684	6.9×10^{-18}

xthenreg: complete result

Object	Cases	Values	Largest $\lvert\Delta\rvert$
CI	32	518	2.4×10^{-11}
N	32	36	0
T	32	36	0
V	32	2 031	4.1×10^{-11}
b	32	259	6.4×10^{-12}
boots_p	32	36	0

xthenreg: bandwidth multiplier h_0() through the public function

Object	Cases	Values	Largest $\lvert\Delta\rvert$
CI	2	40	3.7×10^{-13}
V	2	200	5.1×10^{-14}
b	2	20	1.7×10^{-13}

xtdpthresh: threshold search and final estimate

Object	Cases	Values	Largest $\lvert\Delta\rvert$
e(V_cond)	29	1 208	5.2×10^{-13}
e(b)	29	184	1.7×10^{-12}
e(obj)	29	29	9.5×10^{-10}
first-stage threshold	29	29	3.3×10^{-16}
gamma	29	29	5.0×10^{-16}
q bounds	29	58	2.2×10^{-16}

xtdpthresh: point estimates on the full panel

Object	Cases	Values	Largest $\lvert\Delta\rvert$
e(V_cond)	14	925	1.6×10^{-12}
e(b)	14	111	2.5×10^{-12}
e(gamma)	14	14	5.0×10^{-16}
e(obj)	14	14	1.1×10^{-9}

xtdpthresh: deviations, time effects, unbalanced panels, history

Object	Cases	Values	Largest $\lvert\Delta\rvert$
X at the threshold	41	160 566	2.5×10^{-14}
e(V_cond)	41	1 855	1.6×10^{-10}
e(b)	41	546	1.9×10^{-10}
e(obj)	41	82	7.6×10^{-9}
first-stage threshold	41	41	4.4×10^{-16}
gamma	41	41	4.9×10^{-16}
value	41	1 345 660	1.5×10^{-13}

xtdpthresh: variances and diagnostics

Object	Cases	Values	Largest $\lvert\Delta\rvert$
e(N)	45	45	0
e(N_clust)	45	45	0
e(N_switch)	45	45	0
e(N_units)	45	45	0
e(V)	45	2 217	8.2×10^{-10}
e(V_cond)	45	2 217	2.4×10^{-10}
e(V_upper)	41	393	2.7×10^{-12}
e(ar1)	45	90	4.1×10^{-11}
e(ar1_T1)	45	90	1.9×10^{-10}
e(ar1_TT)	45	90	1.4×10^{-10}
e(ar1_b0)	45	90	2.2×10^{-11}
e(ar1_cond)	45	90	1.6×10^{-11}
e(ar1_p)	45	90	2.8×10^{-13}
e(ar2)	45	90	1.6×10^{-11}
e(ar2_T1)	45	90	9.1×10^{-12}
e(ar2_TT)	45	90	1.5×10^{-11}
e(ar2_b0)	45	90	2.9×10^{-12}
e(ar2_cond)	45	90	1.1×10^{-11}
e(ar2_p)	45	90	1.2×10^{-11}
e(ar_joint)	45	45	0
e(b)	45	309	2.9×10^{-11}
e(b_upper)	41	123	2.7×10^{-12}
e(estimator_twostep)	45	45	0
e(gamma)	45	90	4.4×10^{-16}
e(gamma_bw)	38	76	5.1×10^{-16}
e(hansen)	43	86	1.1×10^{-9}
e(hansen_df)	43	86	0
e(hansen_p)	43	86	5.7×10^{-11}
e(joint_vce)	45	45	0
e(obj)	45	45	1.1×10^{-9}
e(q_nvals_bw)	38	76	0
e(vce_applied)	45	45	0

xtdpthresh: predict

Object	Cases	Values	Largest $\lvert\Delta\rvert$
predict ar_residuals	11	13 098	3.3×10^{-13}
predict regime	11	16 973	0
predict residuals	11	13 144	3.3×10^{-13}

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`predict residuals if`	11	9 659	3.2×10^{-13}
`predict xb`	11	13 144	3.3×10^{-13}
`xb + residuals`	1	700	6.9×10^{-18}

xtdpthresh: bootstraps, confidence set, tests

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`boot_linearity_valid`	55	55	0
`continuity valid`	9	9	0
`e(b_bootci)`	10	134	7.8×10^{-13}
`e(boot_coef_B)`	10	10	0
`e(boot_coef_attempted)`	10	10	0
`e(boot_coef_failed)`	10	10	0
`e(boot_coef_twostep)`	10	10	0
`e(boot_grid_stage1)`	10	10	0
`e(boot_grid_stage2)`	10	10	0
`e(boundary_warn)`	56	56	0
`e(ci_empty)`	52	104	0
`e(ci_grid)`	56	13 002	8.9×10^{-8}
`e(ci_incomplete)`	56	112	0
`e(ci_nseg)`	52	104	0
`e(ci_segments)`	52	348	5.0×10^{-16}
`e(ci_unresolved)`	56	56	0
`e(citest_D)`	8	8	2.0×10^{-9}
`e(citest_accept)`	8	8	0
`e(citest_crit)`	8	8	5.0×10^{-9}
`e(citest_draws)`	9	9	0
`e(citest_gamma)`	9	9	0
`e(citest_p)`	8	8	0
`e(citest_status)`	9	9	0
`e(gamma)`	56	56	4.4×10^{-16}
`e(gamma_hi)`	52	104	5.0×10^{-16}
`e(gamma_lo)`	52	104	2.8×10^{-16}
`e(pval_cont)`	9	9	0
`e(pval_lin)`	55	110	0
`e(rseed)`	56	56	0

xtdpthresh: unbalanced panel of the help file (no bootstrap)

Object	Cases	Values	Largest $\lvert\Delta\rvert$
`e(V_cond)`	2	162	1.9×10^{-15}
`e(b)`	2	18	6.7×10^{-15}
`e(gamma)`	2	2	3.9×10^{-16}
`e(obj)`	2	2	6.6×10^{-14}

B.6 Largest deviation by case

Random numbers (Stata's Mersenne Twister)

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
`0`	1	2 100	`value`	1.3×10^{-15}
`1`	1	2 100	`value`	1.3×10^{-15}
`123`	1	2 100	`value`	1.3×10^{-15}
`12345`	1	2 100	`value`	1.3×10^{-15}
`2147483647`	1	2 100	`value`	1.1×10^{-15}

Matrix routines (invsym, cholinv, syminv)

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
asym	5	29	syminv	4.4×10^{-15}
cond10	4	52	cholinv	4.6×10^{2}
cond11	4	52	cholinv	2.8×10^{3}
cond6	5	77	syminv	6.9×10^{-6}
cond8	5	77	cholinv	3.3×10^{-2}
diag_wide	5	14	invsym	0
diag_zero	3	11	invsym	0
gram	5	77	invsym	4.3×10^{-17}
gram_sing	3	18	invsym	2.8×10^{-17}
huge	5	50	syminv	4.0×10^{-27}
indef	3	6	invsym	0
indef3	3	11	invsym	3.3×10^{-16}
mixed	4	20	syminv	1.6×10^{1}
nan	2	2	syminv_ok	0
negdef	3	6	invsym	0
ones	3	6	invsym	0
spd1	5	5	invsym	4.4×10^{-16}
spd12	5	434	invsym	3.9×10^{-14}
spd2	5	14	invsym	3.1×10^{-15}
spd4	5	50	syminv	5.1×10^{-15}
spd7	5	149	invsym	7.7×10^{-12}
tiny	5	50	cholinv	4.6×10^{5}

Quantiles and threshold grids

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
n300	9	390	qgrid	7.8×10^{-14}
skew1000	9	388	rangen_x	5.3×10^{-14}
small25	9	392	rangen	7.1×10^{-15}
ties10	9	370	rangen	6.2×10^{-15}
uni4520	9	390	t7	3.7×10^{-13}

Data layer: xtset, lags, missing values

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
full	2	40 681	balance	0
unbal	2	35 593	balance	0
weak	2	36 949	balance	0

Time-series operators

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
D.cashflow	1	300	D.cashflow	5.6×10^{-17}
D2.cashflow	1	222	D2.cashflow	5.6×10^{-17}
DF.cashflow	1	300	DF.cashflow	5.6×10^{-17}
DL.cashflow	1	222	DL.cashflow	5.6×10^{-17}
F.cashflow	1	300	F.cashflow	0
F2.cashflow	1	278	F2.cashflow	0
FD.cashflow	1	300	FD.cashflow	5.6×10^{-17}
L.D2.cashflow	1	153	L.D2.cashflow	5.6×10^{-17}
L.cashflow	1	300	L.cashflow	0
L2.cashflow	1	278	L2.cashflow	0
L2D.cashflow	1	205	L2D.cashflow	5.6×10^{-17}
L3.cashflow	1	257	L3.cashflow	0
LD.cashflow	1	222	LD.cashflow	5.6×10^{-17}

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
`LF.cashflow`	1	386	`LF.cashflow`	0
`LS2.cashflow`	1	205	`LS2.cashflow`	5.6×10^{-17}
`S2.cashflow`	1	278	`S2.cashflow`	5.6×10^{-17}
`S2D.cashflow`	1	153	`S2D.cashflow`	5.6×10^{-17}
`S3.cashflow`	1	257	`S3.cashflow`	5.6×10^{-17}
`SS.cashflow`	1	222	`SS.cashflow`	5.6×10^{-17}

xthenreg: data and instruments

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
`c1`	14	4 113	`scal`	0
`c2`	14	4 113	`scal`	0
`c3`	14	4 093	`Z.cols`	1.4×10^{-14}
`c4`	14	4 124	`Z.cols`	2.3×10^{-13}
`c5`	14	3 486	`Z.cols`	1.4×10^{-14}
`c6`	14	4 110	`Z.cols`	7.1×10^{-15}
`f1`	14	3 708	`Z.cols`	2.8×10^{-14}
`f2`	14	3 866	`Z.cols`	1.7×10^{-13}
`m1`	14	3 169	`Z.cols`	7.1×10^{-15}
`m2`	14	3 150	`Z.cols`	7.1×10^{-15}
`w1`	14	3 465	`Z.cols`	2.8×10^{-14}
`w2`	14	3 452	`Z.cols`	1.1×10^{-13}

xthenreg: sup-Wald statistic and linearity test

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
`injected_draws_reproduce_the_same_statistic_as_the_generator`	1	40	`value`	2.8×10^{-14}
`l1`	3	541	`cov_wald`	1.8×10^{-9}
`l2`	3	121	`sup_wald`	2.3×10^{-7}
`l3`	3	361	`cov_wald`	1.4×10^{-6}
`l4`	3	121	`sup_wald`	2.6×10^{-11}
`l5`	3	541	`cov_wald`	1.0×10^{-9}
`p1`	3	121	`boot_sup`	2.9×10^{-12}
`p2`	3	221	`boot_sup`	2.5×10^{-12}
`p3`	3	121	`boot_sup`	6.3×10^{-5}
`p4`	3	121	`sup_wald`	9.8×10^{-8}

xthenreg: covariance and confidence intervals

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
`c1`	1	110	`value`	6.9×10^{-18}
`c1-fast`	3	130	`CI`	3.1×10^{-13}
`c1-literal`	3	130	`CI`	1.3×10^{-13}
`c2`	1	42	`value`	2.7×10^{-20}
`c2-fast`	3	54	`CI`	2.2×10^{-14}
`c2-literal`	3	54	`CI`	1.6×10^{-14}
`c3`	1	72	`value`	1.1×10^{-19}
`c3-fast`	3	88	`CI`	1.4×10^{-12}
`c3-literal`	3	88	`CI`	8.9×10^{-13}
`c4`	1	110	`value`	4.3×10^{-19}
`c4-fast`	3	130	`CI`	4.3×10^{-14}
`c4-literal`	3	130	`b`	3.4×10^{-14}
`c5`	1	20	`value`	2.2×10^{-19}
`c5-fast`	3	28	`CI`	7.1×10^{-15}
`c5-literal`	3	28	`CI`	5.2×10^{-15}
`c6`	1	110	`value`	5.4×10^{-20}

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
c6-fast	3	130	CI	5.9×10^{-13}
c6-literal	3	130	CI	3.5×10^{-13}
c7	1	110	value	6.9×10^{-18}
c7-fast	3	130	CI	3.7×10^{-13}
c7-literal	3	130	CI	1.6×10^{-13}
c8	1	110	value	2.2×10^{-19}
c8-fast	3	130	CI	4.3×10^{-14}
c8-literal	3	130	CI	3.4×10^{-14}

xthenreg: complete result

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
c1	6	266	CI	3.1×10^{-13}
c2	6	114	CI	2.2×10^{-14}
c3	6	182	CI	1.4×10^{-12}
c4	6	133	CI	4.3×10^{-14}
c5	6	62	CI	7.1×10^{-15}
c6	6	133	CI	5.9×10^{-13}
c7	6	133	CI	3.7×10^{-13}
c8	6	133	CI	4.3×10^{-14}
gb1	6	31	CI	1.1×10^{-14}
gb2	6	57	CI	1.2×10^{-14}
gb3	6	31	V	4.1×10^{-11}
gb4	6	21	CI	8.7×10^{-14}
gb5	6	31	V	2.7×10^{-11}
gk1	6	43	CI	7.8×10^{-15}
gk2	6	31	CI	4.9×10^{-15}
gk3	6	43	CI	1.1×10^{-14}
gt1	6	133	CI	6.7×10^{-12}
gt2	6	57	CI	1.2×10^{-12}
gt3	6	57	CI	5.7×10^{-14}
gt4	6	133	CI	9.3×10^{-13}
gv01	6	91	CI	1.2×10^{-13}
gv02	6	91	CI	1.4×10^{-14}
gv03	6	91	CI	1.9×10^{-14}
gv04	6	91	CI	8.5×10^{-14}
gv05	6	91	CI	2.8×10^{-14}
gv06	6	91	CI	2.1×10^{-14}
gv07	6	91	CI	1.4×10^{-13}
gv08	6	91	CI	7.3×10^{-15}
gv09	6	91	CI	2.6×10^{-14}
gv10	6	91	CI	1.7×10^{-13}
gv11	6	91	CI	2.1×10^{-14}
gv12	6	91	CI	3.7×10^{-14}

xthenreg: bandwidth multiplier h_0() through the public function

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
c7	3	130	CI	3.7×10^{-13}
c8	3	130	CI	4.3×10^{-14}

xtdpthresh: instruments and fixed-threshold GMM

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
collapse	1	11 200	value	1.9×10^{-16}
common_variable	1	29 400	value	1.9×10^{-16}
common_variable_collapse	1	7 000	value	1.9×10^{-16}
dependent_iv	1	49 000	value	1.9×10^{-16}

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
duplicate_endog_lag	1	49 000	value	1.9×10^{-16}
duplicate_predetermined_ylag	1	17 500	value	1.7×10^{-16}
endog	1	49 000	value	1.9×10^{-16}
endog_predet	1	68 600	value	0
exog_only	1	29 400	value	1.9×10^{-16}
exogenous_option	1	34 300	value	6.7×10^{-16}
iv_one	1	34 300	value	4.4×10^{-16}
iv_two	1	58 800	value	5.1×10^{-15}
iv_two_collapse	1	50 400	value	5.1×10^{-15}
iv_with_collapse	1	7 000	value	4.4×10^{-16}
kink	1	49 000	value	1.9×10^{-16}
maxlag_1_2	1	24 500	value	0
maxlag_2_5	1	40 600	value	1.9×10^{-16}
maxlag_4_4	1	16 800	value	1.9×10^{-16}
maxlag_collapse	1	7 000	value	1.9×10^{-16}
nocenter	1	49 000	value	1.9×10^{-16}
predet	1	53 900	value	1.9×10^{-16}
static	1	14 700	value	8.9×10^{-16}
static_endog	1	35 200	value	1.9×10^{-16}
unbalanced_alternate_years	1	34 300	value	1.9×10^{-16}
unbalanced_gaps	1	35 560	value	1.9×10^{-16}
unbalanced_missing_endog	1	42 000	value	1.9×10^{-16}
unbalanced_missing_x	1	26 040	value	1.1×10^{-16}
unbalanced_short_units	1	39 200	value	1.9×10^{-16}

xtdpthresh: threshold search and final estimate

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
alternate_years_grid10	6	61	e(V_cond)	3.4×10^{-14}
alternate_years_kink	6	25	e(b)	3.8×10^{-15}
collapse	6	61	e(V_cond)	5.2×10^{-13}
default	6	61	e(obj)	1.5×10^{-11}
exog_only	6	35	e(obj)	2.1×10^{-12}
grid10	6	61	e(obj)	6.3×10^{-11}
grid200	6	61	e(obj)	1.1×10^{-10}
iv	6	61	e(obj)	4.0×10^{-11}
kink	6	25	e(obj)	3.8×10^{-10}
kink_grid10	6	25	e(obj)	2.4×10^{-10}
kink_trim20	6	25	e(obj)	7.7×10^{-12}
min_regime	6	61	e(obj)	6.6×10^{-11}
nocenter	6	61	e(obj)	8.7×10^{-12}
observed_support	6	61	e(obj)	4.2×10^{-11}
quantile	6	61	e(obj)	1.0×10^{-10}
quantile_10	6	61	e(obj)	4.3×10^{-11}
refine1	6	61	e(obj)	2.4×10^{-12}
refine3	6	61	e(obj)	2.4×10^{-12}
refine_quantile	6	61	e(obj)	7.2×10^{-11}
static	6	35	e(obj)	2.0×10^{-13}
trim05	6	61	e(obj)	1.2×10^{-10}
trim20	6	61	e(obj)	7.7×10^{-11}
unbalanced_alternate_years	6	61	e(b)	1.9×10^{-14}
unbalanced_gaps	6	61	e(obj)	1.1×10^{-11}
unbalanced_kink	6	25	e(obj)	9.5×10^{-10}
unbalanced_missing_endog	6	61	e(obj)	3.8×10^{-11}
unbalanced_missing_x	6	61	e(obj)	1.6×10^{-11}
unbalanced_refine	6	61	e(obj)	1.8×10^{-11}
unbalanced_short_units	6	61	e(obj)	4.7×10^{-10}

xtdpthresh: point estimates on the full panel

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
default_instruments	4	58	e(obj)	1.6×10^{-10}
help_grid_200	4	92	e(obj)	3.3×10^{-13}
help_iv_peer	4	92	e(obj)	6.0×10^{-13}
help_kink	4	32	e(obj)	1.1×10^{-12}
help_maxlag_2_10	4	92	e(obj)	1.1×10^{-9}
help_maxlag_3_8	4	92	e(obj)	2.9×10^{-13}
help_quantile_grid	4	92	e(obj)	1.4×10^{-13}
help_refine	4	92	e(obj)	2.7×10^{-13}
help_unbalanced_alternate	4	92	e(obj)	1.1×10^{-12}
help_unbalanced_gaps	4	92	e(obj)	1.8×10^{-13}
help_unbalanced_short	4	92	e(obj)	7.3×10^{-14}
kink_default_instruments	4	22	e(obj)	4.5×10^{-10}
mixed_roles	4	92	e(b)	1.9×10^{-14}
static_default	4	32	e(obj)	5.0×10^{-13}

xtdpthresh: deviations, time effects, unbalanced panels, history

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
fd_age_without_td	7	55 403	e(b)	1.9×10^{-10}
fd_history_panel	7	33 567	e(obj)	4.7×10^{-12}
fd_history_sample	7	14 067	e(obj)	3.6×10^{-13}
fd_iv_with_gaps	7	42 739	e(obj)	1.7×10^{-10}
fd_td_alternate_years	7	34 367	e(obj)	2.0×10^{-10}
fd_td_collapse	7	15 467	e(obj)	6.9×10^{-15}
fd_td_endog	7	49 067	e(obj)	1.6×10^{-11}
fd_td_exog_only	7	28 039	e(obj)	1.8×10^{-13}
fd_td_gaps	7	35 627	e(obj)	1.2×10^{-11}
fd_td_kink	7	46 928	e(obj)	4.4×10^{-11}
fd_td_short_units	7	39 267	e(obj)	6.4×10^{-11}
fd_td_single_period	7	33 067	e(obj)	1.1×10^{-11}
fd_td_sparse_periods	7	14 036	e(b)	9.0×10^{-15}
fod_alternate_years	7	37 797	e(b)	8.2×10^{-15}
fod_collapse	7	16 167	e(V_cond)	3.1×10^{-14}
fod_endog	7	53 967	e(obj)	3.0×10^{-11}
fod_exog_only	7	32 939	e(obj)	2.2×10^{-12}
fod_gaps	7	42 725	e(obj)	1.8×10^{-11}
fod_history_panel	7	33 567	e(obj)	4.9×10^{-11}
fod_history_sample	7	14 067	e(obj)	3.1×10^{-13}
fod_iv	7	63 767	e(obj)	3.1×10^{-10}
fod_iv_with_gaps	7	46 603	e(obj)	5.2×10^{-11}
fod_kink	7	51 828	e(obj)	4.5×10^{-11}
fod_maxlag	7	45 567	e(obj)	9.2×10^{-12}
fod_missing_endog	7	50 117	e(obj)	2.0×10^{-11}
fod_missing_x	7	38 569	e(obj)	6.8×10^{-11}
fod_nocenter	7	53 967	e(obj)	4.0×10^{-12}
fod_predet	7	58 867	e(obj)	1.2×10^{-11}
fod_refine	7	53 967	e(obj)	1.5×10^{-11}
fod_short_units	7	43 187	e(obj)	6.4×10^{-11}
fod_sparse_periods	7	26 701	e(V_cond)	4.7×10^{-14}
fod_static	7	18 239	e(obj)	6.4×10^{-14}
fod_td_alternate_years	7	34 367	e(obj)	4.1×10^{-11}
fod_td_collapse	7	15 467	e(obj)	3.0×10^{-14}
fod_td_endog	7	49 067	e(obj)	6.9×10^{-12}
fod_td_exog_only	7	28 039	e(obj)	6.4×10^{-13}
fod_td_gaps	7	42 725	e(obj)	4.2×10^{-12}
fod_td_short_units	7	39 267	e(obj)	1.6×10^{-11}
fod_td_single_period	7	36 728	e(obj)	7.6×10^{-9}
fod_td_sparse_periods	7	25 543	e(b)	2.7×10^{-13}
fod_td_static	7	13 339	e(obj)	2.6×10^{-14}

xtdpthresh: variances and diagnostics

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
fd_alternate_wind	29	156	e(ar2_cond)	3.4×10^{-14}
fd_bw_large	32	162	e(hansen)	1.7×10^{-11}
fd_bw_small	32	162	e(hansen)	1.7×10^{-11}
fd_collapse_wind	32	162	e(V)	6.1×10^{-14}
fd_discrete_q	32	162	e(hansen)	4.5×10^{-10}
fd_exog_only	32	106	e(hansen)	1.6×10^{-12}
fd_exog_only_wind	32	106	e(hansen)	1.6×10^{-12}
fd_gaps	32	162	e(hansen)	1.6×10^{-11}
fd_gaps_wind	32	162	e(hansen)	1.6×10^{-11}
fd_iv_wind	32	162	e(hansen)	1.3×10^{-10}
fd_jump	32	162	e(hansen)	1.7×10^{-11}
fd_jump_q_regressor	32	236	e(hansen)	1.1×10^{-9}
fd_jump_wind	32	162	e(hansen)	1.7×10^{-11}
fd_kink	28	77	e(hansen)	3.7×10^{-10}
fd_kink_q_regressor	30	98	e(hansen)	6.8×10^{-10}
fd_kink_wind	28	77	e(hansen)	3.7×10^{-10}
fd_nocenter	32	162	e(hansen)	1.8×10^{-12}
fd_nocenter_wind	32	162	e(hansen)	1.8×10^{-12}
fd_predet	32	162	e(hansen)	2.2×10^{-10}
fd_refine_wind	32	162	e(hansen)	5.1×10^{-12}
fd_short_wind	32	162	e(hansen)	4.6×10^{-10}
fd_static	32	106	e(hansen)	2.0×10^{-13}
fd_td	32	162	e(hansen)	8.0×10^{-12}
fd_td_wind	32	162	e(hansen)	8.0×10^{-12}
fod_alternate_wind	29	156	e(ar1_cond)	9.8×10^{-15}
fod_bw_small	32	162	e(hansen)	1.0×10^{-11}
fod_gaps_wind	32	162	e(hansen)	4.5×10^{-12}
fod_jump	32	162	e(hansen)	1.0×10^{-11}
fod_jump_wind	32	162	e(hansen)	1.0×10^{-11}
fod_kink	28	77	e(hansen)	2.7×10^{-11}
fod_kink_wind	28	77	e(hansen)	2.7×10^{-11}
fod_nocenter_wind	32	162	e(hansen)	7.4×10^{-13}
fod_short	32	162	e(hansen)	1.4×10^{-10}
fod_td	32	162	e(hansen)	2.6×10^{-12}
fod_td_wind	32	162	e(hansen)	2.6×10^{-12}
full_help	32	236	e(ar1_TT)	3.7×10^{-12}
full_help_fod	32	236	e(ar1_TT)	7.3×10^{-13}
full_help_fod_wind	32	236	e(ar1_T1)	5.9×10^{-13}
full_help_kink	30	98	e(hansen)	1.1×10^{-12}
full_help_kink_wind	30	98	e(hansen)	1.1×10^{-12}
full_help_wind	32	236	e(ar2_TT)	3.9×10^{-13}
full_maxlag_2_10_wind	32	236	e(hansen)	1.1×10^{-9}
full_unbalanced_alternate_wind	32	236	e(ar1_T1)	2.6×10^{-12}
full_unbalanced_gaps_wind	32	236	e(ar1_b0)	2.5×10^{-13}
full_unbalanced_short	32	236	e(ar1_TT)	8.7×10^{-13}

xtdpthresh: predict

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
fd_jump	5	3 500	predict residuals	4.3×10^{-14}
fd_kink	5	3 500	predict residuals	2.5×10^{-13}
fd_short_wind	5	2 830	predict residuals	3.3×10^{-13}
fd_td	5	3 500	predict residuals	1.3×10^{-14}
fod_alternate_wind	5	2 530	predict ar_residuals	1.7×10^{-15}
fod_gaps_wind	5	2 743	predict ar_residuals	1.8×10^{-14}
fod_jump	5	3 400	predict ar_residuals	2.7×10^{-14}
fod_kink	5	3 400	predict ar_residuals	8.8×10^{-15}
fod_short	5	2 750	predict ar_residuals	2.8×10^{-13}
fod_td	5	3 400	predict ar_residuals	2.3×10^{-14}

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
full_help_fod	5	34 465	predict ar_residuals	8.6×10^{-14}
xb_plus_residuals_is_the_transformed_dependent_variable	1	700	xb + residuals	6.9×10^{-18}

xtdpthresh: bootstraps, confidence set, tests

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
citest_fod	20	272	e(ci_grid)	2.8×10^{-9}
citest_inside	20	270	e(ci_grid)	1.1×10^{-9}
citest_kink	20	270	e(ci_grid)	4.3×10^{-10}
citest_one_step	20	270	e(ci_grid)	5.6×10^{-16}
citest_outside	20	270	e(citest_crit)	1.4×10^{-9}
citest_unseeded_subset	20	270	e(ci_grid)	1.1×10^{-9}
citest_with_tests	22	278	e(ci_grid)	9.9×10^{-9}
coef_onestep	20	283	e(ci_grid)	1.1×10^{-9}
coef_onestep_percentile	20	283	e(ci_grid)	1.1×10^{-9}
coef_onestep_td	20	287	e(ci_grid)	3.1×10^{-10}
coef_twostep	20	283	e(ci_grid)	1.1×10^{-9}
coef_twostep_collapse	20	287	e(ci_grid)	1.6×10^{-12}
coef_twostep_fod	20	285	e(ci_grid)	2.8×10^{-9}
coef_twostep_kink	20	277	e(ci_grid)	4.3×10^{-10}
coef_twostep_one_step_fit	20	283	e(b_bootci)	3.1×10^{-14}
coef_twostep_percentile	20	283	e(ci_grid)	1.1×10^{-9}
coef_twostep_refine	20	293	e(ci_grid)	1.5×10^{-9}
cont_fd	15	271	e(ci_grid)	9.9×10^{-9}
cont_fd_collapse	15	255	e(ci_grid)	8.2×10^{-10}
cont_fd_gaps	15	269	e(ci_grid)	3.8×10^{-8}
cont_fd_one_step	15	259	e(ci_grid)	3.1×10^{-15}
cont_fd_static	15	269	e(ci_grid)	1.3×10^{-10}
cont_fd_td	15	265	e(ci_grid)	2.9×10^{-9}
cont_fod	15	267	e(ci_grid)	4.0×10^{-9}
fd_collapse	13	267	e(ci_grid)	1.6×10^{-12}
fd_exog_only	13	259	e(ci_grid)	2.0×10^{-11}
fd_gaps	13	267	e(ci_grid)	2.4×10^{-9}
fd_iv	13	265	e(ci_grid)	4.0×10^{-8}
fd_jump	13	263	e(ci_grid)	1.1×10^{-9}
fd_kink	13	263	e(ci_grid)	4.3×10^{-10}
fd_level_90	13	263	e(ci_grid)	1.5×10^{-9}
fd_no_linearity_test	11	260	e(ci_grid)	1.1×10^{-9}
fd_one_step	13	263	e(ci_grid)	5.6×10^{-16}
fd_quantile_grid	13	267	e(ci_grid)	2.9×10^{-9}
fd_refine	13	273	e(ci_grid)	1.5×10^{-9}
fd_short	13	267	e(ci_grid)	8.9×10^{-8}
fd_small_boot	13	271	e(ci_grid)	9.3×10^{-10}
fd_static	13	273	e(ci_grid)	1.6×10^{-12}
fd_td	13	267	e(ci_grid)	3.1×10^{-10}
fod_jump	13	265	e(ci_grid)	2.8×10^{-9}
fod_kink	13	261	e(ci_grid)	2.0×10^{-10}
fod_one_step	13	269	e(ci_grid)	1.2×10^{-15}
fod_td	13	267	e(ci_grid)	2.9×10^{-10}
full_citest	20	266	e(ci_grid)	2.4×10^{-10}
full_cont	15	261	e(ci_grid)	2.4×10^{-10}
full_help	13	259	e(ci_grid)	2.4×10^{-10}
full_help_fod	13	265	e(ci_grid)	1.1×10^{-9}
full_help_kink	13	257	e(ci_grid)	1.6×10^{-11}
unit_citest	11	138	e(ci_grid)	5.0×10^{-16}
unit_collapse	13	263	e(ci_grid)	6.3×10^{-12}
unit_exog_only	13	257	e(ci_grid)	1.9×10^{-9}
unit_fd	8	135	e(ci_grid)	5.0×10^{-16}
unit_gaps	8	130	e(ci_grid)	4.4×10^{-16}
unit_nocenter	13	263	e(ci_grid)	3.7×10^{-12}

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
unit_quantile_grid	8	134	e(ci_grid)	4.7×10^{-16}
unit_static	13	261	e(ci_grid)	8.0×10^{-12}

xtdpthresh: unbalanced panel of the help file (no bootstrap)

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
fd	4	92	e(obj)	6.6×10^{-14}
fod	4	92	e(obj)	5.0×10^{-14}

xtdpthresh: hostile inputs

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
iv_equals_x	1	5	value	4.8×10^{-8}
q_four_values	1	7	value	4.9×10^{-15}
units_fewer_than_iv	1	5	value	9.2×10^{-9}
units_fewer_than_iv_boot	1	7	value	5.9×10^{-8}

Simulated panels of the Seo-Shin design

Case	Objects	Values	Worst object	$\lvert\Delta\rvert$
1	1	5	value	3.4×10^{-9}
2	1	5	value	4.5×10^{-9}
3	1	5	value	4.5×10^{-9}
4	1	5	value	4.3×10^{-9}

B.7 Comparisons under a looser tolerance

The rule of Section 1 holds for every other value of this report. 4 comparison(s) are made under a looser relative tolerance, declared in the test and not hidden in the tables above: the input is so ill-conditioned that the error of any implementation is of the order of the condition number times the machine precision (a matrix of condition number 10^{13} gives about 10^{-3}), so two correct inverses legitimately differ in the fourth digit.

Case	Object	Relative tolerance	$\lvert\Delta\rvert$
cond13	invsym	1.0×10^{-3}	6.0×10^{1}
cond13	cholinv	1.0×10^{-3}	2.3×10^{8}
cond13	syminv_ok	1.0×10^{-3}	0
cond13	psd	1.0×10^{-3}	0

B.8 What this report does not cover

- The statistical behaviour of the package's own seeds (size and power of the tests, coverage of the confidence set over many panels) is measured by Monte Carlo and compared with the papers, not with Stata.
- The tables written by the package (LaTeX, HTML, Word, Excel) repeat the validated numbers; the tests check the printed digits, not the typography. They are not in the deviations above.
- Timings are measured on the same machine one run after the other and are not a validation.
- The messages that Stata prints and the package does not (the warning of an unbalanced `boottype(unit)` and the hint on near-common variables in a rank failure) are not compared; they do not change any number.

B.9 Environment

Reference	Stata 17.0 MP; xthenreg (Seo, Kim and Kim 2019) and xtdpthresh 0.9.38
Package	pyxtdpthresh 0.0.1
Python	3.12.10; NumPy 2.5.4; Numba 0.68.1
Platform	Windows 11
Report built	04 November 2026