Read Me for Replication Package "Strategic or Confused Firms? Evidence from "Missing" Transactions in Uganda"

Miguel Almunia, Jonas Hjort, Justine Knebelmann, and Lin Tian

Contents

Overview 2
Data Availability and Provenance Statements 2
Data Dictionary 2
Computational Requirements 4
Code Description 4

Overview

This document provides replication instructions for the analysis in Almunia et al. (2021). We provide a one-click code that allows to generate all the tables and figures in the paper. The estimated program runtime largely depends on the machine's hardware specifications, but can be expected to last between 7 and 14 days.

Data Availability and Provenance Statements

The analysis relies on administrative data from the Uganda Revenue Authority (URA). More precisely, we use data from Ugandan firms' monthly VAT declarations between 2013 and 2016. The VAT returns are submitted electronically by the firms to the URA. The data are confidential and proprietary to the URA.

The data was accessed within the context of a Memorandum of Understanding signed by the URA and the researchers. For other researchers interested in accessing the same data, the point of contact would be the Manager of Research and Revenue Modelling, Ms. Tina Kaidu (tkaidu@ura.go.ug).

Summary of Availability

☐ All data **are** publicly available.
☐ Some data **cannot be made** publicly available.
☒ **No data can be made** publicly available.

Data Dictionary

We list below the different VAT datasets. We describe the variables of each dataset used for the analysis.[1] The replication codes described in the following section enable to produce all results, estimations and figures, starting from the raw URA data. The source of all the raw data is the Uganda Revenue Authority.

- **mvrtot_stripped_to_july_2017**: This dataset includes the Monthly VAT returns that firms submit electronically to the URA.
  - sellertin: TIN of the firm filing the return (masked)
  - rtn_no: return number
  - fdate_year: fiscal year (YY = JulYY - JunYY+1)
  - fdate_month: fiscal month (Jul=1 - Jun=12)
  - offset_claim: offsets claimed from previous period
  - tot_final_tax_a_b: total sales to final consumers
  - tot_final_tax_a_b: total VAT charged on sales to final consumers
  - tot_output: total VAT on sales
  - tot_input: total VAT paid on inputs
  - vat_payable: tot_output minus tot_input
  - vat_due: vat_payable minus offets
- **vs1_vat_to_july_2017**: This dataset includes the Schedule 1 Annex to the VAT return that firms submit electronically to the URA. This Schedule reports the sales to other VAT firms at the transaction level.

[1] For the sake of concision, we do not describe the variables of the datasets that are not used in the main part of the analysis. However they are all included and labeled in the datasets. We remain available for further detail when necessary.

  - sellertin: TIN of the seller, the firm filing the return (masked)
  - buyertin: TIN of the buyer (masked)
  - amount: amount of transaction excluding VAT
  - vat_charged: VAT charged
  - idate_year: invoice year
  - idate_month: invoice month
- **vs24_vat_to_july_2017**: This dataset includes the Schedules 2 and 4 Annexes to the VAT return that firms submit electronically to the URA. These Schedules report purchases from other VAT firms at the transaction level.
  - sellertin: TIN of the seller (masked)
  - buyertin: TIN of the buyer, the firm filing the return (masked)
  - amount: amount of transaction excluding VAT
  - vat_paid: VAT paid
  - idate_year: invoice year
  - idate_month: invoice month
- **akm_dataset_connect_edit_round**: This dataset is the main sample we use to run the fixed-effect analysis. Observations are at the pair-month level, and the dataset is created based on the Schedules data (see do files for full detail).
  - sellertin: TIN of the seller (masked) (string)
  - buyertin: TIN of the buyer (masked) (string)
  - d_vat: amount reported by the buyer minus amount reported by the seller
  - period: transaction date
  - seller: seller identifier (numeric)
  - buyer: buyer identifier (numeric)
- **akm_dataset_connect_edit_control_round**: This dataset is the sample we use to run the robustness check version of the fixed-effect analysis, with controls
  - sellertin: TIN of the seller (masked) (string)
  - buyertin: TIN of the buyer (masked) (string)
  - d_vat: amount reported by the buyer minus amount reported by the seller
  - same_area: dummy variable for whether two firms are located in the same sub-county
  - io_share: share of products from the seller's sector that are sold to the buyer's sector
  - period: transaction date
  - seller: seller identifier (numeric)
  - buyer: buyer identifier (numeric)
- **Customs_VS3_Comparison_edit**: This dataset is the used for the VAT Schedule 3 versus Customs comparison. It is created by merging VAT Schedule 3, an Annex to the VAT return that lists imports, and Customs data.
  - tin: TIN of the importer, the firm filing the return (masked)

  - diff: difference between amount reported in VAT Schedule 3 and Customs
  - vat_vs3_2: VAT paid as declared in VAT Schedule 3
  - vat_cust2: VAT paid as declared in the customs import data
  - acty_c: firm sector code
  - sales_q: firm yearly turnover quintile
  - : s_1_-s_24_: variables indicating the share of imports corresponding to each HS good code category at the firm-week level
  - imonthyear: transaction month-year

Computational Requirements

Software Requirements

- Stata (code was last run with version 16)
  - `rscript` (available here)
  - `datacheck` (available at SSC)
  - `distinct` (available at SSC)
  - `astile` (available at SSC)
  - `parmest` (available at SSC)
  - `lassopack` (available at SSC)
  - The program "`packages_stata.do`" will install all dependencies locally
- R (code was last run with version 4.0.3)
  - `tidyverse` (available at CRAN)
  - `reshape2` (available at CRAN)
  - `foreign` (available at CRAN)
  - `lfe` (available at CRAN)
  - `haven` (available at CRAN)
  - `magrittr` (available at CRAN)
  - `dplyr` (available at CRAN)
  - `whoami` (available at CRAN)
  - `rsample` (available at CRAN)
  - `tidyr` (available at CRAN)
  - `broom` (available at CRAN)
  - `bbw` (available at CRAN)
  - Tthe program "`packages_R.R`" will check if the relevant dependencies are already installed and, if not, will install them locally

Memory and Runtime Requirements

The code was last run on a **20-core Intel-based desktop computer with 64 GB of RAM and Windows version 10 (x64)**. The computation took around one week. We estimate that the time needed to reproduce the analyses on a standard desktop machine would range between 7 to 14 days. You will need a minimum of **900 GB** of local storage available to reproduce the analysis.

Code Description

The Tables and Figures in the paper and in the online appendix are constructed using Stata, R, and Python. The replication files are organized in two main folders:

- `\code`: contains all the Stata dofiles and R scripts and stores the corresponding log files
- `\data`: stores all the datasets

The dofile `_Master.do` allows to generate all Figures and Tables. It is a one-click code that automatically defines the relevant file paths and creates subfolders to store tables and figures, respectively. The `_Master.do` also automatically calls R within the current Stata session, meaning that it is able to run both all the dofiles and R scripts at once. All Tables and Figures will be stored in the folder `\output`. Once the code has been executed, the replication file can be generated by compiling the tex file `output\build\AlmuniaEtAlvat_replicationfile.tex`.

We provide below a description of each part of the code, run in the following order:

- `packages_stata.do` will install the Stata packages needed for the analysis
- `transformations.do` will create programs for several transformations used in the analysis (inverse hyperbolic sine, conversion to millions and billions, standardization)
- `winsorize.do` will create a program for winsorizing
- `a0. data_import_DT.do` imports raw VAT data (monthly VAT summary declarations and transaction-level details from the VAT Schedules)
- `a1. data_cleaning_DT.do` cleans raw VAT data
- `a2. data_combine_DT.do` combines information from the monthly VAT summary declarations and the VAT Schedules
- `a3. data_stripped_DT.do` restricts the datasets to the variables needed for the analysis and merges the monthly VAT summary declarations with the VAT Schedules
- `a4. import_customs.do` imports raw customs data
- `a5. cleaning_customs.do` cleans raw customs data
- `_Master_geography.do` runs the geocoding exercise, that aims to assign geo-coordinates to firms and compute distance to URA office. The relevant dofiles can be found in the subfolder `\code\Geography\`:
  - `a. creation` creates the necessary datasets for the analysis
  - `b. clean` cleans the address data and extracts district, subcounty and address
  - `geocoding.py` conducts geocoding, i.e. derive geographic coordinates from using geographic data
  - `d. analysis` compiles the various geographic data files, cleans them, and produces geo-coordinates to work with
  - `geo_analysis.py` selects the closest URA office for each firm and closest URA office to the centroid of each subcounty a firm is located in
- `b0. data_prep_AKM.do` prepares the VAT dataset for the fixed-effect analysis
- `b0. data_construct_AKM.do` constructs the dataset for the fixed-effect analysis
- `b0. data_final_AKM.do` finalizes the datasets for the fixed-effect analysis (rounding, restricting to largest connected set, merging in other relevant covariates, creating robustness samples)
- `weights.do` computes the weights for the fixed-effect analysis
- `packages_R.R` installs the relevant R packages needed to run the analysis and sets the paths in R. This script is run every time there is a call to R within Stata, i.e. every time we run the command `rscript` in `_Master.do`.
- `c0. AKM_bootstrap_resampling.R` creates the 100 bootstrap samples for the computation of the variance of Q
- `c1. AKM_bootstrap_lcs.R` defines the largest connected set for each bootstrap sample
- `c2. AKM_bootstrap_100_iterations.R` runs the fixed-effect estimation on the 100 bootstrap samples
- `c3. AKM_benchmark_model.R` runs the fixed-effect estimation on the benchmark sample
- `_Master_AKM_robustness.R` runs the fixed-effect estimation on each of the robustness samples (scripts c4. to c9.)
  - `c4. AKM_control.R` runs the fixed-effect estimation on the sample with controls
  - `c5. AKM_raw.R` runs the fixed-effect estimation on the raw sample with no rounding nor pure timing corrections
  - `c6. AKM_panel.R` runs the fixed-effect estimation on the "panel" sample
  - `c7. AKM_round_5.R` runs the fixed-effect estimation on the sample corrected for rounding at 5% of transaction value
  - `c8. AKM_years_new_round_TWFE.R` runs the year-by-year fixed-effect estimation
  - `c9. AKM_collect_results_bs.R` retrieves the results of the 100 bootstrap iterations
- `e2. AKM_recover_robustness_checks.do` imports and reads the results of the robustness checks

estimations into Stata

- `e3. prepare_customs.do` prepares customs data for analysis
- `f4. format_vs_data.do` creates codeboook of 4-digit HS good codes and descriptions from EAC codebook and customs data
- `f2. goodcodes_clean.do` imports VAT Schedules data with good codes associated to the text descriptions of goods
  - The good codes are not included in the original VAT Schedules. We generate them using a machine learning algorithm, trained on good descriptions and harmonized HS good codes in the customs data. The algorithm is then applied to the text descriptions from the VAT Schedules.
- `f3. upstreamness.do` uses good codes to build upstreamness measures
- `f4. format_vs_data.do` cleans the VAT schedules datasets with the good codes so they have the same format as the datasets from the main analysis
- `g. _master_sim.R` generates the data for the simulation exercise
- `g5.` to `g8.` run the AKM estimation on the simulated data and retrieve the results
- `Figure1_heatmap_hist.do` generates Figure 1
- `FigureE1_switchers.do` generates Figure E.1
- `FigureE2_switchers.do` generates Figure E.2
- `FigureF1_Qs_firm_size.do` generates Figure F.1
- `FigureF2_Qs_distribution.do` generates Figure F.2
- `Table1_firmtype_class.do` generates Table 1
- `Table2_final_consumers.do` generates Table 2
- `Table3_revenue_consequences.do` generates Table 3
- `Table4_customs.do` generates Table 4
- `TableB1_comparison_types.do` generates Table B1
- `TableB2_transition_matrix.do` generates Table B2
- `TableB3_firmtype_class_noreplacing.do` generates Table B3
- `TableB4_ctrl_panel.do` generates Table B4
- `TableB5_raw_round5.do` generates Table B5
- `TableC1_alt_rev_consequences.do` generates Table C1
- `TableG1_vat_shares.do` generates Table G1
- `TableG2_extensive_descr.do` generates Table G2
- `TableG3_type_and_VAT.do` generates Table G3
- `TableG4_seem_anomalous.do` generates Table G4
- `TableG5_customs.do` generates Table G5