Georgia firm merge

November 10, 2021

A brief explainer of what we're doing here:

We have data on ~400k bids for gov't auctions in Georgia, and (in a separate dataset) information about the firms represented in that dataset. In particular, we have information about the shareholders of those firms at a variety of dates, and we are interested in seeing whether common ownership leads to cartel-like behavior. This code merges the two datasets and, for each bid, adds information about the firm's executive and major shareholders.

This would've been done in Stata but there's a lot of messing around with strings (which is NOT FUN in Stata) and being able to hold two datasets in memory simultaneously is *very* convenient for the final step.

[93]:
```
import pandas as pd
import numpy as np
from tqdm.notebook import tqdm
import subprocess
```

[2]:
```
dat = pd.read_stata("../01_data/GeorgianFirmsMerged.dta")
```

Preliminary transformations—if an observation does not have a time associated with it, we assume the observation is as of the firm's registration date

If we don't have an observation or registration date, set it to dummy value 1/1/1970. We can't groupby over NaN values, and this doesn't affect results in any way.

[3]:
```
dat.loc[dat.date.isnull(), 'date'] = dat.loc[dat.date.isnull(),␣
↪'registration_date']
dat.loc[dat.date.isnull(), ['date', 'registration_date']] = pd.to_datetime("1/1/
↪1970")
dat['former'] = dat.role.str.contains('(Former)', regex=False)
dat = dat.drop_duplicates()
```

We then best-guess who is the chief executive of a firm at a given time.

[35]:
```
def identify_director(dat):
    # These are the (most common) titles for executives
    director_roles = [
        'Direktori',
        'Direktori, Partniori',
        'Partniori, Direktori',
```

```
        'Generaluri Direktori',
        'Partniori, Generaluri Direktori',
        'Generaluri Direktori, Partniori',
        'Khelmdzghvaneli',
        'Partniori, Khelmdzghvaneli',
        'Khelmdzghvaneli, Partniori',
        'Aghmasrulebeli Direktori'
    ]

    # If a firm has one person and is marked as 'Individual Entrepeneur', the␣
↪director name is the same as the
    # firm's name
    # Ind'l entrepeneur
    if (dat.iloc[0]['legal_form'] == "Individual Entrepreneur") or (len(dat) ==␣
↪1 and not dat.iloc[0]['legal_form']):
        return np.array([dat.iloc[0]['firm_name'], '---'])


    # If all personnel are marked as 'former', we assume the firm is no longer␣
↪in operation
    if dat.former.to_numpy().all(): #defunct
        return np.array(['DEFUNCT', '---'])


    # Find all rows where the job title is in the list above
    director = dat[dat.role.isin(director_roles)]

    # If there's exactly one, we're in luck
    if len(director) == 1: # identified
        return director[['person', 'personal_num']].to_numpy().flatten()

    # Otherwise, we assume the person with the highest share who is not marked␣
↪'former' is the executive
    elif len(director) == 0: # underitentified
#         print(data)
        dir_guess = dat[dat.former == False]\
                    .sort_values('share', ascending = False)\
                    .head(1)[['person', 'personal_num']]\
                    .to_numpy()\
                    .flatten()
        return dir_guess

    elif len(director) > 1: # overidentified
        dir_guess = dat[dat.former == False]\
                    .sort_values('share', ascending = False)\
                    .head(1)[['person', 'personal_num']]\
                    .to_numpy()\
```

```
                    .flatten()
        return dir_guess

    return None

# For each firm/date pair, find the controlling interest
grped = dat.groupby(['firmid', 'date'])
print(len(grped))

for name, data in tqdm(grped):
    result = identify_director(data)

    dat.loc[data.index , 'director_name'] = result[0]
    dat.loc[data.index , 'director_num'] = result[1]
```

```
117594
```

```
HBox(children=(FloatProgress(value=0.0, max=117594.0), HTML(value='')))
```

We also want to include the five largest shareholders for each firm/date combination. This adds a shareholder rank for each firm/date combo and drops all but the largest 5.

[38]:
```
grouped = dat.sort_values('share', ascending = False)\
             .groupby(['firmid', 'date'])\
             .head(5)

grouped['grp_id'] = grouped.groupby(['firmid', 'date']).cumcount()
```

Then pivot from long to wide format (we'll clean up the MultiIndex in the next step

[39]:
```
widedat = pd.pivot(grouped,
                        index = ['firmid', 'date'],
                        columns = ['grp_id'],
                        values = ['personal_num', 'person', 'role', 'share',␣
 ↪'num_role']
                       )
```

[39]:

		personal_num					person	
grp_id		0	1	2	3	4	0	1
firmid	date							
01-10-043308	1970-01-01	NaN	NaN	NaN	NaN		NaN	
086 001 3021	1970-01-01	NaN	NaN	NaN	NaN		NaN	
10001000480	2008-06-23	NaN	NaN	NaN	NaN		NaN	
10001000667	1970-01-01	NaN	NaN	NaN	NaN		NaN	
10001000742	2011-01-03	NaN	NaN	NaN	NaN		NaN	
…	…	…	…	…	…	…	…	…

```
SI 40312623           1970-01-01               NaN  NaN  NaN  NaN          NaN
TVA FR 54 349 505 370 1970-01-01               NaN  NaN  NaN  NaN          NaN
U24230MH1971PLC015134 1970-01-01               NaN  NaN  NaN  NaN          NaN
Ugoplast              1970-01-01               NaN  NaN  NaN  NaN          NaN
dotecon               1970-01-01               NaN  NaN  NaN  NaN          NaN

                                                 … share                    \
grp_id                                 2    3    4  …     0    1    2    3
firmid                date                         …
01-10-043308          1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
086 001 3021          1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
10001000480           2008-06-23     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
10001000667           1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
10001000742           2011-01-03     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
…                                    …    …    …    …   …    …    …    …
SI 40312623           1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
TVA FR 54 349 505 370 1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
U24230MH1971PLC015134 1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
Ugoplast              1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN
dotecon               1970-01-01     NaN  NaN  NaN  …   NaN  NaN  NaN  NaN

                                   num_role
grp_id                                 4         0    1    2    3    4
firmid                date
01-10-043308          1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
086 001 3021          1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
10001000480           2008-06-23     NaN       0.0  NaN  NaN  NaN  NaN
10001000667           1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
10001000742           2011-01-03     NaN       0.0  NaN  NaN  NaN  NaN
…                                    …         …    …    …    …    …
SI 40312623           1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
TVA FR 54 349 505 370 1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
U24230MH1971PLC015134 1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
Ugoplast              1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN
dotecon               1970-01-01     NaN       NaN  NaN  NaN  NaN  NaN

[117594 rows x 25 columns]
```

Flatten the column indices, relabeling them to be e.g. share_2

[40]:
```
widedat.columns = [f"{a[0]}_{a[1]+1}" for a in widedat.columns.to_flat_index()]
```

Rearrange them so they go e.g. person_1 person_num_1 share_1 … person_n person_num_n share_n

[41]:
```
col1 = ['person', 'personal_num', 'role', 'share', 'num_role']
col2 = [str(a) for a in range(1,6)]
```

```
colorder = ['_'.join([a,b]) for b in col2 for a in col1 ]
```

Reorder according to the above, and flatten the row index

[42]:
```
widedat = widedat.reindex(columns = colorder).reset_index()
```

Replace placeholder values with the proper NaN missing value for Pandas. Also prepare the original dataset for merging

[74]:
```
dat.loc[((dat.director_num == '---') | dat.director_num.isna() | dat.
 ↪director_num == None) & (dat.legal_form != 'Individual Entrepeneur') ,␣
 ↪'director_num' ] = np.NaN
firmdat = dat[['firmid', 'firm_name', 'legal_form' , 'registration_date',␣
 ↪'address' , 'email', 'date', 'director_name', 'director_num']].
 ↪drop_duplicates()
```

Merge the two together. We now have a dataset that, for each firm/date pair, has one observation with:

- Firm information, like name, ID, registration date, CEO, etc.
- For the top 5 shareholders
  - Name
  - ID number
  - Share
- Director can be (and almost always is) one of the listed shareholders

[75]:
```
mergedfirmdat = firmdat.merge(widedat, on=['firmid','date'], how='right')
```

Save the thing (as Stata format

[ ]:
```
fdir = '~/Documents/Freiburg/ZEW/georgia/01_data/'
fname = 'georgiamergedfirms.dta'
mergedfirmdat.where(mergedfirmdat.isna(), mergedfirmdat.astype(str)).
 ↪to_stata(fdir + fname, version = 119, write_index=False)
```

And compress it. This little script loads the .dta, converts all strings to strL, and uses Stata's compress to optimize. In this case, we go from ~400MB to ~40MB, although this is an extreme result

[113]:
```
a = subprocess.call(['stata-mp' ,'do', 'stata-compress.do', f'{fdir}' ,␣
 ↪f'{fname}'])
```

Now read in the list of bids, both winning and losing

[114]:
```
procurement_dat = pd.read_stata('../01_data/Georgia_backlog_merged.dta')
```

Going back to our previous dataset, sort by date and group by firm

[116]:
```
grouped_firm_dat = mergedfirmdat.sort_values('date', ascending = False).
 ↪groupby('firmid')
```

This is where the magic happens. For each bid observation:

- We find the group of observations corresponding to that firm
- We drop all that are newer
- Because we sorted, the first remaining observation is the most recent. We get its ID and set it as the 'foreignkey'

[149]:
```
missing_firms = 0

procurement_dat['foreignkey'] = -1
for i in tqdm(procurement_dat.index):
    proc = procurement_dat.loc[i]

    try:
        firmgroup = grouped_firm_dat.get_group(proc['firmid'])
        firmobs = firmgroup[firmgroup.date <= proc.announcement_date].head(1)
    except KeyError:
        missing_firms += 1
        procurement_dat.loc[i, 'foreignkey'] = -1

    try:
        procurement_dat.loc[i, 'foreignkey'] = firmobs.index.item()
    except ValueError:
        missing_firms += 1
        procurement_dat.loc[i, 'foreignkey'] = -1
```

```
HBox(children=(FloatProgress(value=0.0, max=384492.0), HTML(value='')))
```

We then merge using the previously calculated 'foreignkey'

[151]:
```
georgiawithfirmsdat = pd.merge(procurement_dat, mergedfirmdat, left_on =␣
↪'foreignkey', right_index = True, how='left')
```

And save this (now massive) dataset.

[153]:
```
fdir = '~/Documents/Freiburg/ZEW/georgia/01_data/'
fname = 'georgia_withfirms.dta'
georgiawithfirmsdat.where(georgiawithfirmsdat.isna(), georgiawithfirmsdat.
↪astype(str)).to_stata(fdir + fname, version = 119, write_index=False)
a = subprocess.call(['stata-mp' ,'do', 'stata-compress.do', f'{fdir}' ,␣
↪f'{fname}'])
```

[ ]:
```

```